Préface

L'an passé, ce qui nous a immédiatement décidé à nous replonger encore une fois dans l'organisation - pourtant éprouvante - de SSTIC, ce fut tout simplement l'ambiance. La chose qui nous a le plus marqué est venue des multiples échanges qui ont eu lieu entre les diverses personnes présentes. Industriels, étatiques, universitaires, tout le monde était réuni avec une seule et même préoccupation : apprendre.

Il en est, si, si, je vous assure, qui croient toujours qu'un firewall les protégera de tout ou presque. En fait, le "presque", ce sont les virus, c'est pour ça qu'on a inventé l'antivirus. On a donc fait des progrès : en combinant pare-feu et antivirus, cette fois, on devrait être tranquille. Et pour peu que l'antivirus soit à jour, c'est la panacée ... sauf que souvent le système hôte n'a pas été patché depuis son installation.

Ces produits, comme beaucoup d'autres en sécurité informatique, ont une caractéristique comme : ils sont réactifs, ils n'agissent qu'*a posteriori*.

Cette tendance réactive se retrouve également dans les comportements, mentalités et réactions. Tout le monde s'accorde à dire que la veille et l'anticipation sont primordiales afin de se préparer et d'appréhender les nouvelles menaces. Cela passe par de l'apprentissage et de l'expérimentation. La curiosité serait-elle finalement une qualité ? Malheureusement, cette démarche active n'est pas toujours encouragée, en particulier dans nos entreprises, voire quand elle n'y est pas entravée.

Tout le comité d'organisation se joint à moi pour remercier les personnes sans qui cet évènement n'aurait pu avoir lieu. Tout d'abord, nous remercions les membres du comité de programme, qui ont eu la lourde tâche de statuer sur les 37 propositions envoyées pour n'en retenir que 14. Nous avons été impressionnés par le travail réalisé par toutes les personnes qui nous ont fait une proposition !

Malheureusement pour ceux qui ont été acceptés, nous leur avons encore imposé un travail supplémentaire, auquel ils se sont pliés de bonne grâce pour réaliser l'ouvrage que vous tenez entre vos mains.

Il y a aussi les "conférenciers invités", Bernard Carayon, Danielle Kaminsky, Philippe Oechslin, David Bénichou, Serge Lefranc, Marie Barel, et Nicolas Fischbach, qui pensaient se la couler douce. Et non ! Eux aussi ont contribué à cet ouvrage, subissant les mêmes exigences que les autres.

Merci à vous tous pour nous avoir permis d'élaborer ce programme attractif, complet et riche. En revanche, ne comptez peut-être pas sur les remerciements de l'assistance qui vous en voudra devant la densité de ce programme. Quoique ...

Nous remercions également nos différents sponsors, la revue MISC, l'ESAT, le CEA, France Telecom R&D, Arche, Supélec Rennes, la société Thalès et la SEE qui nous ont donné les moyens matériels d'arriver à nos fins.

Enfin, avouons le, tout cela n'aurait pas été possible sans l'aide et le soutien de trois personnes :

- comme l'an passé, Robert Longeon nous a fait une promotion incroyable !
- prenant en pitié celui qui a tant souffert pour monter le précédent site web et gérer les inscriptions, Marina Rieidi s'est privée de longues heures de sommeil pour nous faciliter la vie au travers du nouveau site web.
- point focal de SSTIC, l'ESAT et son directeur, le Général Bagaria, qui nous accueille.

En espérant avoir oublié personne. Si tel était le cas, pardonnez notre étourderie et mettez-la sur le compte de l'organisation qui nous accapare beaucoup.

Quoiqu'il en soit, bienvenue dans la matrice.

Juin 2004 Frédéric Raynal pour le comité d'organisation

Organisation

SSTIC'04 est organisé par l'Ecole Supérieure et d'Application des Transmissions en coopération avec l'Ecole Supérieure d'Electricité, le Commissariat à l'Energie Atomique, la société Arche (groupe Omnetica) et le journal de la sécurité informatique MISC et la société Thalès.

Comité d'organisation

Christophe Bidan	Supélec
Philippe Biondi	Arche - groupe Omnetica
Eric Detoisien	
Thiébaut Devergranne	
Eric Filiol	Ecole Sup. et d'Application des Transmissions
Thierry Martineau	Ecole Sup. et d'Application des Transmissions
Laurent Oudot	Commissariat à l'Energie Atomique/DIF
Frédéric Raynal (Président)	MISC Magazine/EADS S&DE
Franck Veysset	France Télécom R&D

Comité de programme

Gildas Avoine	EPFL
Christophe Bidan	Supélec
Renaud Bidou	Iv2 Technologies
Philippe Biondi	Arche - Groupe Omnetica
Patrick Chambet	Edelweb - Groupe ON-X
Pascal Chour	AQL
Yves Correc	CELAR
Éric Detoisien	
Thiébaut Devergranne	
Éric Filiol	ESAT
Caroline Fontaine	CNRS-LIFL
Olivier Heen	Thomson
Pascal Lointier	CLUSIF
Robert Longeon	CNRS
Thierry Martineau	ESAT
Benjamin Morin	France Télécom R&D
Laurent Oudot	CEA/DAM
Frédéric Raynal	MISC magazine
Ludovic Rousseau	Gemplus
Marc Rybowicz	Université de Limoges
Franck Veysset	France Télécom R&D

Sponsors

CLUSIF (CLUb de la sécurité des Systèmes d'Information Français) - Commissariat à l'Energie Atomique - Société Arche - Ecole Supérieure et d'Application des Transmissions - MISC/Le journal de la sécurité informatique - Société de l'Électricité, de l'Électronique, et des Technologies de l'Information et de la Communication - Société Thalès

Table des matières

Conférences invitées

Kerberos et la Sécurité

Emmanuel Bouillon

Commissariat à l'Énergie Atomique
Direction des Applications Militaires
BP 12, 91680 Bruyères-le-Chatel, France
emmanuel.bouillon@cea.fr

Résumé Afin d'évaluer la pertinence de Kerberos dans un contexte donné, il convient de comprendre non seulement l'intérêt de ce protocole mais aussi ses limites, y compris en termes de sécurité. Cet article propose de rappeler brièvement le fonctionnement du protocole Kerberos, puis de décrire certains problèmes de sécurité qui peuvent se poser lors de son déploiement.

Les points abordés sont regroupés en deux catégories : d'une part les problèmes inhérents au protocole lui-même, d'autre part les problèmes liés aux difficultés pratiques de déploiement. Les aspects de la première catégorie rappelleront les limitations connues de Kerberos et seront présentés à la lumière des implémentations actuelles. Les aspects de la deuxième catégorie aborderont les difficultés liées à la mise en œuvre de ce protocole, ces difficultés pouvant aboutir à des compromis impactant la sécurité.

Les éléments présentés dans cet article visent à aider à évaluer : l'apport de Kerberos dans un environnement donné, la cohérence de cet apport au vu des mesures de sécurité existantes dans cet environnement et la pertinence de cette solution par rapport à d'autres mesures de sécurité.

1 Introduction

1.1 Qu'est-ce que Kerberos ?

Selon la mythologie, Kerberos est le chien à trois têtes gardant le pont menant aux portes de l'enfer. Plus "récemment", Kerberos fut le nom choisi pour le mécanisme d'authentification d'un projet mené au MIT appelé Athena. Aujourd'hui, Kerberos est un protocole d'authentification réseau. La version actuelle du protocole, la version 5, est un standard défini dans la RFC 1510 [1]. Les versions 1 à 3 ne sont restées que des versions de développement. La version 4 est décrite dans [2] et [3].

1.2 Intérêts du protocole

Peu de mécanismes résolvent le problème de l'authentification réseau unifiée en milieu hétérogène, qui plus est, mixte Unix/MS Windows. L'utilisation pour l'authentification, des NIS (Network Information Services), de SSH pour la distribution de fichiers passwd et shadow, d'UIS (Unix Integration Services)

service Microsoft qui n'est plus supporté, des SFU (Service For Unix) sous Windows, de LDAP, sont des techniques qui souffrent toutes de défauts de mise en œuvre et de sécurité. Kerberos présente l'avantage d'être un standard disponible sous les systèmes "libres" (Linux, *BSD) et sous les systèmes propriétaires, Solaris (SunMicrosystem), Irix (SGI), Windows 2000 et XP. La portabilité des implémentations "OpenSource" permet de compléter cette liste. Bien sûr, Kerberos résout plusieurs problèmes de sécurité classiques dans l'authentification des clients et des services au sein d'un réseau. Enfin, Kerberos est aujourd'hui le principal mécanisme de sécurité sous-jacent supporté dans les implémentations de la GSSAPI [4,5]. Cette API étant la base des RPCSEC_GSS, mécanisme désigné pour sécuriser la prochaine version de NFS (version 4, [6]).

1.3 Objectif de cet article

Pour mieux évaluer la pertinence de ce protocole dans un contexte donné, il convient d'en comprendre non seulement les avantages mais aussi les limites, y compris en termes de sécurité. Cet article propose de rappeler le fonctionnement du protocole Kerberos, puis de décrire certains problèmes de sécurité qui peuvent se poser lors du déploiement de Kerberos. Les points abordés sont regroupés en deux catégories :

- Les problèmes inhérents au protocole lui-même ;
- Les problèmes liés aux difficultés pratiques de déploiement.

Dans la suite, une application sera dite "kerbérisée" pour signifier qu'elle supporte le protocole Kerberos.

2 Le protocole Kerberos

2.1 Caractéristiques du protocole

Le protocole Kerberos se base sur les deux articles [7] et [8]. Il permet l'authentification des utilisateurs et des services sur un réseau supposé non sûr. Par exemple, les données sur ce réseau peuvent être lues ou modifiées, les adresses des machines peuvent être faussées. Ces informations ne peuvent donc servir à l'authentification. Kerberos utilise une tierce partie de confiance. Toutes les entités du réseau, utilisateurs et services, appelés principaux, font confiance à cette tierce partie (le serveur Kerberos ou KDC pour Key Distribution Center). Enfin, Kerberos utilise des mécanismes de chiffrement basés sur des algorithmes à clef symétrique (ou secrète) : tous les principaux partagent cette clef secrète avec le serveur Kerberos.

2.2 Fonctionnement détaillé : de l'authentification par secret partagé à Kerberos v5

Dans toute la suite, Alice (A) sera l'entité qui initie la communication (analogie avec le client, l'utilisateur), Bob (B) l'entité qui répond (analogie avec le service, serveur applicatif). K_{ab} sera la clef partagée entre Alice et Bob (on a $K_{ab} = K_{ba}$).

Authentification par secret partagé Kerberos utilise des algorithmes de chiffrement à clef secrète pour l'authentification. La manière la plus simple de faire de l'authentification à l'aide de tels algorithmes est la suivante :

1. Alice appelle Bob
2. Bob répond à Alice avec un nombre généré aléatoirement, N_b
3. Alice retourne $K_{ab}\{N_b\}$

A l'issue de cet échange, Bob en déchiffrant la réponse de Alice, vérifie que Alice connaît bien la clef secrète K_{ab}. Pour une authentification mutuelle, Alice émet à son tour un nombre aléatoire (N_a) que Bob retourne chiffré par K_{ab}.

Une autre méthode pour l'authentification mutuelle à l'aide d'un algorithme à clef secrète est de faire envoyer par Bob, non pas un nombre aléatoire, mais ce nombre chiffré par K_{ab}. Alice déchiffre ce message, effectue une opération triviale sur le nombre obtenu et le renvoie chiffré. On obtient le schéma suivant :

1. Alice appelle Bob
2. Bob envoie $K_{ab}\{N_b\}$
3. Alice déchiffre le message, obtient N_b et envoie $K_{ab}\{N_b - 1\}$
4. Alice envoie $\mathrm{K}_{ab}\{N_a\}$
5. Bob déchiffre le message, obtient N_a et envoie $K_{ab}\{N_a - 1\}$

Ce qui, en regroupant les messages indépendants, donne (schéma 1) :

1. Alice envoie $K_{ab}\{N_a\}$
2. Bob déchiffre le message, obtient N_a, envoie $K_{ab}\{N_a - 1\}$ et $K_{ab}\{N_b\}$
3. Alice déchiffre le message, obtient N_b et envoie $K_{ab}\{N_b - 1\}$

C'est ce schéma qui apparaît dans le protocole de Needham et Schroeder [7]. Avec une amélioration toutefois :

Utilisation d'une tierce partie de confiance L'un des inconvénients majeurs du schéma 1 est son manque d'extensibilité. La généralisation à m utilisateurs et n services, implique la distribution préalable de $m \times n$ clefs partagées. Une solution à cette problématique est l'introduction d'une tierce partie de confiance (une autre solution est l'utilisation d'un algorithme de chiffrement asymétrique pour négocier une clef de session symétrique). Cette tierce partie (TP) de confiance connaît les secrets partagés de chaque entité à authentifier (K_a pour Alice, K_b pour Bob). L'authentification mutuelle peut alors se faire comme suit :

1. Alice demande à TP d'accéder à Bob
2. TP génère une clef de session entre Alice et Bob, K_{ab}
3. TP envoie $K_a\{K_{ab}, Bob\}$ à Alice et $K_b\{K_{ab}, Alice\}$ à Bob

A ce niveau, on peut appliquer le schéma 1 pour l'authentification mutuelle par algorithme à clef secrète. En effet, Alice connaissant K_a, peut en déduire K_{ab}. De même, Bob connaissant K_b peut en déduire K_{ab}.

On remarque cependant que pour la distribution de K_{ab}, Alice, le client envoie un message alors que la tierce partie commune à tous les utilisateurs en envoie deux. On préfère en général faire porter la charge sur le client et non sur la tierce partie qui doit gérer tous les clients. On modifie donc légèrement le schéma précédent qui devient :

1. Alice demande à TP d'accéder à Bob
2. TP génère une clef de session entre Alice et Bob, K_{ab}
3. TP envoie $K_a\{K_{ab}, Bob\}$ et $K_b\{K_{ab}, Alice\}$ à Alice
4. Alice appelle Bob et lui envoie $K_b\{K_{ab}, Alice\}$

Alice, qui ne connaît pas K_b, ne peut rien faire de $K_b\{K_{ab}, Alice\}$ si ce n'est l'envoyer à Bob. Cette information est le ticket pour dialoguer avec Bob.

On abouti alors au schéma de Needham et Schroeder, le nombre aléatoire N_1, envoyé avec la première requête n'est là que pour rendre unique cette requête et sa réponse pour éviter des attaques de type "rejeu".

1. Alice demande à TP d'accéder à Bob et joint N_1
2. TP génère une clef de session entre Alice et Bob, K_{ab}
3. TP envoie $K_a\{N_1, K_{ab}, Bob, Ticket_b\}$ à Alice, où $Ticket_b = K_b\{K_{ab}, Alice\}$
4. Alice déchiffre la réponse, en tire $Ticket_b$ et K_{ab}
5. Alice appelle Bob avec $Ticket_b$ et $K_{ab}\{N_a\}$ (schéma 1)
6. Bob déchiffre le Ticket, en tire K_{ab}, répond avec $K_{ab}\{N_a - 1, N_b\}$
7. Alice répond avec $K_{ab}\{N_b - 1\}$

Kerberos Le protocole Kerberos améliore le schéma de Needham et Schroeder d'une part en remplaçant les nombres aléatoires par des dates (timestamps), d'autre part en séparant le rôle de la tierce partie de confiance en deux services :

- Le service d'authentification (AS pour Authentication Service)
- Le générateur de ticket de service (TGS pour Ticket Granting Service)

L'utilisation des timestamps permet d'introduire la péremption de ticket et de réduire la fenêtre de temps pendant laquelle le rejeu de ticket est possible. Ce type d'attaque est surtout empêché par l'utilisation d'un cache stockant les authentificateurs reçus et dont la date de validité n'a pas expiré. On note que cette propriété introduit une dépendance entre Kerberos et le service de temps.

La séparation du rôle du KDC en deux services induit deux types de ticket différents :

- Le TGT (Ticket Granting Ticket) est envoyé par l'AS. Une demande de TGT est en fait une demande d'accès au service TGS. Le TGT est donc le ticket du service TGS. La réponse de l'AS contient aussi la clef de session entre le client (A) et le service TGS ($K_{a,tgs}$) chiffrée par la clef secrète du client (K_a).
- Le TS (Ticket Service) pour un service donné (S) est envoyé par le TGS sur présentation d'un TGT et d'un timestamps chiffré avec la clef de session $K_{a,tgs}$. Le TGS sait donc que le client a pu déchiffrer la réponse de l'AS

et donc connaît la clef secrète K_a. Le TGS génère une clef de session entre A et S, $K_{a,s}$. La réponse du TGS contient donc notamment, cette clef de session chiffrée par $K_{a,tgs}$ (A pourra ainsi la déduire) et le TS proprement dit, constitué, entre autre, de $K_{a,s}$ chiffré par K_s, la clef secrète du service S (S, sur présentation de ce ticket pourra en déduire $K_{a,s}$).

La même clef $K_{a,tgs}$ permet d'obtenir les clefs de session des services accédés pendant toute la validité du TGT. Le principal avantage de cette séparation des rôles du KDC est de permettre le SSO (Single Sign On). En effet, seul l'exploitation du TGT nécessite l'utilisation de la clef secrète du client. Ainsi, tant que son TGT est valide, le client pourra acquérir des TS sans réutiliser sa clef, qui est directement déduite du mot de passe dans le cas d'un utilisateur. Cette propriété, couplée avec la possibilité de rendre le TGT "forwardable", c'est-à-dire réutilisable une fois connecté au service, permet à Kerberos de fournir un SSO.

La pré-authentification Au cours du dialogue précédent, on constate qu'il n'est pas nécessaire de connaître K_a pour obtenir de la tierce partie de confiance un TGT et un message chiffré par K_a. Il suffit de le demander à l'AS pour l'obtenir.

La pré-authentification résout ce problème. Elle exige que la requête à l'AS, émise par le client, soit accompagnée d'un timestamp chiffré par K_a. L'AS peut ainsi valider que la requête émane bien d'une entité connaissant sa clef secrète.

2.3 Les relations de confiance inter-royaume

Kerberos prévoit la possibilité qu'un principal puisse s'authentifier auprès de principaux n'appartenant pas à son royaume : Il s'agit de l'authentification inter-royaume. Cela implique d'établir une relation de confiance entre les deux royaumes. Cette relation peut être unilatérale ou bilatérale.

Deux méthodes sont possibles pour établir une telle relation.

Méthode directe Les deux KDC partagent une clef secrète utilisée pour prouver l'identité d'un principal lorsqu'il change de royaume. L'inconvénient de cette méthode est son manque d'extensibilité. La généralisation à n royaumes impose l'échange de $n \times (n-1)/2$ clefs ou $n \times (n-1)$ clefs dans le cas de relations bilatérales. Ceci peut être résolu par l'utilisation de la méthode transitive.

Méthode transitive Dans ce cas, on définit le chemin des royaumes partageant une clef secrète. Ce chemin peut être soit explicite (mécanisme des CApath, Certificate Authority Path) soit hiérarchique via le DNS.

3 Kerberos et la sécurité

Cette partie rappelle certains des problèmes de sécurité qui peuvent être rencontrés au cours du déploiement de Kerberos sur un réseau.

3.1 Problèmes inhérents au protocole

Les problèmes décrits ci-après sont connus de longue date. Pour certains, Dug Song [9] notamment a démontré la faisabilité de leur exploitation. L'objectif de ce paragraphe est d'en rappeler le principe et de montrer que sans précaution, ces attaques sont toujours valables dans les implémentations actuelles du protocole.

Attaque par dictionnaire et pré-authentification Dans la version 4 du protocole, le mécanisme de pré-authentification n'était pas prévu. N'importe qui pouvait obtenir un message chiffré avec la clef secrète d'un utilisateur donné. Toute la sécurité reposait sur l'incapacité de déduire la clef secrète de ce message. Bien sûr, compte tenu de la propension des utilisateurs à choisir des mots de passe faibles, une attaque par dictionnaire avait clairement de fortes chances d'être fructueuse.

Sans pré-authentification, écrire un outil qui ferait l'équivalent d'un `ypcat passwd` avec les NIS ne pose pas de difficultés.

Dès lors, adapter un outil de craquage de mots de passe pour lui permettre d'attaquer ces messages chiffrés est assez simple. Dug Song propose un patch pour John the Ripper [10] et la version 4 de Kerberos. Avec l'avènement de la version 5 de Kerberos et l'apport de la pré-authentification, la conscience de cette faiblesse du protocole s'est estompée. Les documentations insistent peu sur son utilité. Il convient alors de faire les remarques suivantes sur ce sujet :

- La pré-authentification n'est pas activée par défaut sur nombre d'implémentations du protocole. Elle est notamment impossible si l'on souhaite une compatibilité avec Kerberos version 4.
- L'activation de la pré-authentification diminue le risque mais ne l'élimine pas puisque dans le cas d'une authentification licite, le message chiffré peut être lu sur le réseau. Modifier un sniffer à cette fin est simple.
- Adapter un outil de craquage de mot de passe à la version 5 du protocole est simple.

En conséquence, il est important de rappeler que la pré-authentification est une option indispensable à la sécurité de Kerberos mais n'élimine pas complètement le risque d'attaque par dictionnaire tant que la clef secrète sera dérivée d'un mot de passe statique choisi par l'utilisateur.

La mise en place de mesures assurant la robustesse des mots de passe choisis par les utilisateurs est un moyen efficace de lutter contre ce problème. Le support par Kerberos d'algorithmes de chiffrement asymétrique (PKINIT [11]) couplé avec l'utilisation d'un deuxième facteur pour l'authentification, une carte à puce par exemple, supprimerait le risque d'une telle attaque. Espérons que les efforts actuels pour l'intégration de PKINIT et de la pré-authentfication "matérielle" aboutiront rapidement.

Usurpation du KDC Cette deuxième attaque a aussi été mise en évidence par Dug Song notamment. Le but de ce paragraphe est d'en expliquer le principe, de donner les moyens de s'en protéger et leurs limites, ceux-ci n'étant que rarement préconisés dans les documentations.

Fondamentalement, Kerberos permet d'authentifier un utilisateur auprès d'un service kerbérisé et réciproquement. Cependant, on peut dans le cadre du déploiement de Kerberos, utiliser ce protocole comme point d'entrée sur un réseau, c'est-à-dire au niveau de l'authentification système.

Le moyen le plus simple pour y parvenir est l'utilisation d'un module PAM, d'autres font exécuter la commande `kinit` à la connexion de l'utilisateur ou encore changent la commande de login. Dans le cas de l'utilisation d'un module PAM, le système se sert de la capacité à déchiffrer la réponse du KDC avec le mot de passe fourni par l'utilisateur pour l'authentifier. Il ne s'agit pas d'une réelle authentification Kerberos auprès d'un service. Le défaut de cette configuration est qu'il n'y pas authentification de la réponse du KDC. La requête et l'analyse de sa réponse sont faites par le client. Si la clef donnée par l'utilisateur permet de déchiffrer le TGT obtenu, l'utilisateur est autorisé à se connecter et possède un TGT. Ce n'est que s'il cherche à accéder à un service kerbérisé qu'il s'en servira pour faire une demande de TS.

Ce manque d'authentification de la réponse du KDC permet une attaque par usurpation du KDC si l'on a la possibilité d'écouter et d'injecter du trafic entre le système cible et le KDC. Si la seule présentation d'un TGT correspondant à la clef secrète fournie au login permet de se connecter à un système, il est possible de contourner cette authentification comme suit :

1. L'attaquant se connecte au nom de la victime en entrant un mot de passe quelconque
2. Le système émet une requête de TGT
3. L'attaquant répond en utilisant la clef dérivée du mot de passe qu'il a lui-même fournie au système
4. La réponse peut être déchiffrée par la clef : l'utilisateur est autorisé à se connecter

Bien sûr, le TGT ainsi obtenu ne permettra pas d'obtenir de TS valide. Mais une fois connecté, l'utilisateur peut accéder au disque et à tous les services non kerbérisés (Home en NFSv3 non kerbérisé par exemple).

Une contre-mesure pour cette attaque est de vérifier l'authenticité de la réponse du KDC en demandant un TS avec ce TGT pour le service `host` du système local et d'en vérifier la validité. Si la réponse est valide, le KDC partage non seulement la clef secrète de l'utilisateur mais aussi celle du service host du système. Il est à noter que le module PAM Kerberos le plus utilisé [12] prévoit cette fonctionnalité mais elle n'est pas documentée.

Kerberos et son environnement On décrit parfois Kerberos comme un protocole faisant peu d'hypothèses sur la sécurité du réseau où il est déployé. Il convient cependant de comprendre les conséquences d'une compromission même partielle du réseau sur l'authentification.

La compromission, au niveau adminstrateur, d'une machine quelconque du réseau aboutit à la compromission des clefs des services de cette machine et

des tickets des utilisateurs qui y sont connectés. L'attaquant peut alors usurper l'identité :

- des services de cette machine tant que leurs clefs ne sont pas changées
- des utilisateurs authentifiés sur cette machine pendant la durée de validité de leurs tickets

L'installation d'un cheval de Troie pourra aboutir à la compromission de mots de passe des utilisateurs, rendant l'usurpation de leur identité possible jusqu'au changement de leur mot de passe. En conséquence, le déploiement de Kerberos ne permet pas de s'abstenir d'autres mesures élémentaires de sécurité comme l'installation des mises à jour de sécurité. Enfin, dans un environnement kerbérisé, la compromission d'une seule machine spécifique, un KDC, conduit au contrôle total par l'attaquant de tous les authentifiants de entités Kerberos (utilisateurs et services) de ce royaume. On comprend la nécessité de sécuriser avec soin les machines assurant cette fonction et aussi de cloisonner les différents groupes d'utilisateurs dans des royaumes différents.

Kerberos et l'autorisation L'objectif d'une solution de sécurité est de s'attaquer à la problématique des trois 'A' (Authentication, Autorization, Accounting). Kerberos cherche à résoudre le problème du premier 'A', l'authentification. Il fournit au service l'assurance que l'utilisateur qui se présente à lui est bien celui qu'il prétend être (à la sécurité de la clef secrète près). Réciproquement, l'utilisateur sait qu'il s'adresse bien au service escompté (à la sécurité du fichier `keytab` près). De plus, en centralisant les demandes (échouées ou réussies) de connexions des clients aux services, Kerberos permet de tracer précisément l'accès de qui à quoi. Il participe ainsi à la traçabilité, le troisième 'A'. Par contre, Kerberos ne participe pas au processus d'autorisation. Celui-ci revient au service, confiant dans l'identité de son client. Cependant, les services classiques des implémentations de Kerberos, ne fournissent pas de mécanisme d'autorisation générique. L'utilisation à des fins d'autorisation du fichier `.k5login` est limitée. L'intégration du support des PAM, par exemple, est difficile compte tenu principalement du fait que ce mécanisme ne garantit pas la distinction entre authentification et autorisation.

3.2 Problèmes liés aux difficultés pratiques de déploiement

Kerberos peut améliorer notablement la sécurité d'un réseau. Cependant son déploiement est difficile, à tel point qu'on le déconseille parfois [13] :

"In our opinion, most sites are better off without it [sic]."

Ces difficultés aboutissent souvent à un compromis entre la meilleure solution du point de vue de la sécurité et les contraintes d'administration opérationnelle. Ce paragraphe présente une liste non exhaustive de difficultés pratiques qui peuvent être rencontrées lors du déploiement de Kerberos et dont les solutions ou contournements auront des conséquences sur la sécurité.

Installation via le réseau À son installation, la machine n'est pas kerbérisée. Ce processus automatique ne peut donc pas complètement être sécurisé par Kerberos. Un moyen de distribuer les fichiers `keytab` d'une machine est d'utiliser la commande `kadmin`, de donner le mot de passe d'un principal privilégié et d'extraire le fichier `keytab`. Toute solution entièrement automatique aboutit à un certain niveau d'exposition du fichier *keytab*. Il est évidemment des environnements où l'intervention d'un administrateur à chaque (ré-)installation n'est pas envisageable.

Accès à un service sans mot de passe Plusieurs tâches d'administration classiques nécessitent l'accès à un service sans fournir de manière interactive l'authentifiant de l'identité utilisée. De telles procédures sont bien évidemment difficilement compatibles avec une méthode d'authentification sûre. Citons quelques exemples :

- Exécution de procédures automatiques :
  L'obtention d'un TGT nécessite toujours l'utilisation de la clef secrète du principal concerné. Si une procédure automatique a besoin d'accéder à un service kerbérisé (un script en crontab par exemple), comment lui faire obtenir un TGT puis un TS pour ce service. Plusieurs solutions ont été proposées pour résoudre ce problème [14,15]. Elles correspondent chacune à un compromis entre la sécurité et la faisabilité.
- Dépannage des utilisateurs :
  Il est parfois nécessaire à un administrateur de prendre l'identité d'un utilisateur, par exemple pour déboguer un problème qu'il n'arrive pas à reproduire dans son environnement. Que faire si ce problème nécessite l'accès à un service kerbérisé. Comment l'administrateur peut-il obtenir un TGT au nom de l'utilisateur ?
  Une solution évidente pour les implémentations qui utilisent un cache de ticket local sur le disque dur est d'utiliser ce fichier sous root. Ceci nécessite toutefois que l'utilisateur soit simultanément connecté. Cette contrainte peut être jugée trop importante. Dans ces conditions, comment un utilisateur peut obtenir le TGT d'un autre utilisateur ?
  Fondamentalement, un administrateur du KDC est omnipotent sur tous les principaux. Il peut donc obtenir un TGT pour un utilisateur. Cependant, certaines implémentations rendent ceci compliqué si l'on ne souhaite pas réinitialiser le mot de passe de l'utilisateur. C'est le cas pour l'implémentation du MIT, pas pour Heimdal. De plus, la population des administrateurs des KDC n'est pas nécessairement la même que celle qui dépanne les utilisateurs.
- Exécution de commandes en mode batch :
  Cette problématique regroupe les deux difficultés précédentes puisqu'il s'agit pour un processus système d'exécuter des commandes de manière automatique, au nom d'un utilisateur quelconque. Situation que l'on peut rencontrer au niveau de l'ordonnancement d'un serveur de calcul parallèle par exemple. [16] notamment propose une solution. Les relations inter-

royaume peuvent aussi être utilisées efficacement. Cela revient toutefois à dénaturer la sécurité de Kerberos en redonnant aux administrateurs une partie de leur pouvoir.

Relation de confiance unilatérale et ticket "forwardable" Etablir une relation de confiance inter-royaume est le moyen souvent préconisé pour permettre à un utilisateur authentifié dans un royaume (ROYAUME-1) d'accéder à un service d'un autre royaume (ROYAUME-2). Ceci implique que les responsables de ROYAUME-2 font confiance dans l'authentification et la sécurité mise en place dans ROYAUME-1. Si cette confiance n'est pas réciproque, il convient de ne mettre en place qu'une relation de confiance inter-royaume unilatérale : Un utilisateur avec un TGT de ROYAUME-2 ne pourra pas obtenir un TS pour un service du ROYAUME-1.

Comme rappelé précédemment, l'option "forwardable" d'un ticket permet d'obtenir un SSO. Par exemple, à chaque rebond de machine en machine, c'est-à-dire à chaque obtention de TS, le TGT est dupliqué sur la machine cible. Ainsi, le mot de passe de l'utilisateur n'a pas à être fourni de nouveau, l'authentification Kerberos est toujours possible. C'est pourquoi cette option est si prisée des utilisateurs quand il s'agit du service `host/machine`.

Cependant, si cette option est activée dans ROYAUME-1 et acceptée dans ROYAUME-2, le TGT d'un utilisateur authentifié dans ROYAUME-1 et accédant à un service de ROYAUME-2 sera dupliqué sur la machine cible. Dès lors, un administrateur de ROYAUME-2 peut accéder au TGT de cet utilisateur. Ainsi, la relation de confiance unilatérale entre ROYAUME-2 et ROYAUME-1 nécessite toutefois, dans ces conditions (ticket "forwardable" et donc intégration dans le SSO de ROYAUME-1), que les responsables de ROYAUME-1 fassent confiance aux administrateurs de ROYAUME-2. Ce qui fait nettement évoluer la signification de la relation de confiance "unilatérale".

Protection des administrateurs Les points précédents confirment une fois de plus que les administrateurs occupent une place cruciale pour la sécurité d'un réseau, quand bien même kerbérisé. Les efforts consentis pour le déploiement de Kerberos doivent donc être en rapport avec ceux dédiés à la protection des administrateurs et de leurs stations de travail. Les premiers seraient inutiles si les homes des administrateurs sont exportés par NFS à tous et leurs displays accessibles, par exemple.

Authentification applicative Le support de l'authentification système via Kerberos est maintenant largement répandu. Son déploiement peut donc être envisagé même en environnement hétérogène. Cependant, le système d'exploitation n'est souvent pas l'unique partie du système d'information où une authentification est requise. L'accès au mail, via les protocoles POP ou IMAP, doit être pris en considération. L'accès à une base de données ou à une appplication Web par exemple, peut exiger une authentification supplémentaire ou simplement la même mais renouvelée pour accorder son accès.

Dans le cas d'une authentification renouvelée, c'est-à-dire où l'application utilise le même protocole que l'authentification système, il est évident que le déploiement de Kerberos aura des conséquences sur le fonctionnement de l'application. Même si des standards, le plus souvent basés sur la GSSAPI, voient progressivement le jour [17,18], leur intégration n'est jamais transparente. Cet impact est donc à prendre en compte dès l'origine du projet de déploiement. Dans le cas d'une authentification applicative reposant sur un protocole différent, le problème repose de nouveau sur la cohérence des mécanismes de sécurité mis en œuvre. Par exemple, si les deux méthodes d'authentification reposent sur un mot de passe au choix de l'utilisateur, le mécanisme d'authentification le plus faible affaiblit le plus fort compte tenu de la probabilité d'un choix identique pour ces mots de passe.

Compatibilité des implémentations Le protocole Kerberos dans sa version 5 est un standard bien défini [1]. Lors de son déploiement se pose la question du choix d'une ou plusieurs implémentations de ce protocole. Certaines sont "OpenSource" comme l'implémentation du MIT ou Heimdal, d'autres propriétaires comme SEAM, fournie par Sun Microsystem ou Active Directory. Deux catégories de problèmes peuvent apparaître :

- Les incompatibilités avec le protocole lui-même : l'implémentation est-elle respectueuse de la RFC de Kerberos, de la RFC de la GSSAPI ?
- Les incompatibilités entre implémentations si le choix a été fait d'en utiliser plusieurs : notamment, des problèmes peuvent apparaître au niveau de l'API d'administration, du changement de mot de passe, des relations d'approbation et des algorithmes supportés. Sur ce dernier point par exemple, pour établir une relation d'approbation entre un KDC MIT et un KDC Windows 2000 Server SP4, il est nécessaire de dégrager la liste des algorithmes supportés. Un tel contournement a évidemment un impact sur la sécurité. Ce problème est résolu avec un KDC sous Windows 2003.

4 Conclusion

Kerberos offre un moyen puissant et efficace de sécuriser un réseau. Son adoption par un grand nombre de systèmes d'exploitation est gage de pérennité. Néanmoins, il est important d'en connaître le fonctionnement, les limites ainsi que les difficultés de son déploiement et de son administration. Ces dernières pouvant éventuellement aboutir à des compromis impactant la sécurité. On pourra alors appréhender de manière cohérente la sécurité d'un réseau et éviter le danger d'en surévaluer le niveau.

Références

1. J. Kohl et B. Clifford Neuman, The Kerberos Network Authentication Service (Version 5), *Internet Request for Comments RFC 1510*, septembre 1993.

2. B. Clifford Neuman et T. Ts'o, Kerberos : An Authentication Service for Computer Networks, *IEEE Communications*, Vol. 32, number 9, pp. 33-38, septembre 1994.
3. J. Kohl, B. Clifford Neuman et T. T'so, The Evolution of the Kerberos Authentication System In Distributed Open Systems, *IEEE Computer Society Press*, pp. 78-94, 1994.
4. E. Bouillon et P. Deniel, Premiers pas avec la GSSAPI, *Linux Magazine*, volume 52, juillet 2003.
5. J. Linn, The Kerberos Version 5 GSS-API Mechanism, *Internet Request for Comments RFC 1964*, juin 2001.
6. S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler et D. Noveck, Network File System (NFS) verion 4 Protocol, *Internet Request for Comments RFC 3050*, avril 2003.
7. R. Needham et M. Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM*, Vol. 21, number 12, pp. 993-999, décembre 1978.
8. D. Denning et G. Sacco, Time stamps in Key Distribution Protocols, *Communications of the ACM*, Vol. 24 numéro 8, pp. 533-536, août 1981.
9. `http://www.monkey.org/~dugsong/`
10. `http://www.openwall.com/john`
11. B. Tung, C. Neuman, M. Hur, A. Medvinski, J. Wray et J. Trostle, Public Key Cryptography for Initial Authentication in Kerberos, *IETF Internet-Draft.*
12. `http://sourceforge.net/projects/pam-krb5/`
13. E. Nemeth, G. Snyder, S. Seebass et T. Hein, *Unix system administration Handbook*, Prentice Hall, 3ème édition, 2001.
14. Kerberos FAQ, `http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html`
15. I. Hollander, P. Rajaram, et C. Tanno, Morgan Stanley & Co, Kerberos on Wall street, *Usenix*, 1996.
16. A. Rubin et P. Honeyman, Long Running Jobs in an Authenticated Environment, *Usenix*, 1993.
17. E. Baize et D. Pinkas, The Simple and Protected GSS-API Negotiation Mechanism, *Internet Request for Comments RFC 2478*, décembre, 1998.
18. R. J. Detry, S. D. Kleban et P. C. Moore, The Generalized Security Framework, `http://www.prod.sandia.gov/cgi-bin/techlib/access-control.pl/2001/018338.pdf`, juin 2001.

Backdoors furtives et autres fourberies dans le noyau

Arnaud Ebalard[1], Pierre Lalet[2], and Olivier Matz[3]

Droids Corporation
team@droids-corp.org
http ://www.droids-corp.org/
[1]troglocan@droids-corp.org [2]pierre@droids-corp.org [3]zer0@droids-corp.org

Remarques préliminaires

Cet article expose les raisons pour lesquelles il peut être intéressant de se cacher dans le noyau d'un système d'exploitation, que l'on soit animé de bonnes ou de mauvaises intentions. Nous présenterons quelques techniques pour le faire, ainsi que quelques contre-mesures possibles.

Nos connaissances dans le domaine des noyaux portent exclusivement sur les systèmes de type Unix (Linux et ⋆BSD principalement), mais les techniques que nous allons décrire peuvent sans doute être adaptées à des systèmes d'exploitation alternatifs (BeOS, QNX, MS-Windows, ...).

Notre expérience est venue, au départ, du port de Sebek2 (voir *Know Your Enemy : Sebek*[1]) pour FreeBSD (dans le cadre d'un projet scolaire) puis pour NetBSD et OpenBSD, ainsi que de l'étude des ports *officiels* pour Linux et OpenBSD (nos travaux à propos de Sebek et des honeypots en général sont consultables sur `http://honeynet.droids-corp.org/`).

Sommairement, Sebek est un outil lié aux pots à miel. Installé sur une machine, il envoie sur le réseau le contenu de la totalité des appels à `read()`. On peut ainsi récupérer tout ce qu'a fait le pirate qui a pris la main sur le pot à miel (y compris ce qui a pu être protégé par des solutions cryptographiques comme ssh).

Pour l'implémentation de ces ports, nous nous sommes assez peu inspirés des versions existantes et nous avons préféré recommencer en respectant le protocole tel qu'il est décrit dans le document de présentation de Sebek précédemment cité. Nous avons recherché des exemples de codes similaires, c'est-à-dire, des façons de détourner le code du noyau. Nous avons rencontré beaucoup de choses liées au *côté obscur* de la bidouille informatique, et nous avons mesuré l'intérêt de se placer dans le noyau pour écrire une *backdoor*. Cela nous a donné l'idée de cette présentation.

[1] `http://www.honeynet.org/papers/sebek.pdf`

1 Introduction

1.1 De l'intérêt de se cacher dans le noyau

Le noyau est l'élément central du système d'exploitation. Cette phrase, sous les dehors d'une belle banalité digne d'un commercial, est essentielle dans les domaines de la dissimulation, de l'interception de données, et de l'action sur le système corrompu. Cela veut dire que toute tentative de détection passera par lui, que tous les actes effectués sur le système passeront par lui, et qu'il est capable de tout faire sur le système (plus encore que `root`, dont les privilèges sont susceptibles d'être limités par des patchs noyau sous Linux, ou encore par le `securelevel` sous BSD). Il est omniscient et omnipotent.

En effet, le noyau fournit une interface normalisée permettant aux applications de réaliser des tâches faisant appel au matériel (accès disque, émission/réception de trafic réseau, accès au clavier, ...). Ainsi, le noyau est en quelque sorte le point de convergence obligé des données transitant entre les processus de l'utilisateur et les périphériques du système. De plus, celui-ci contrôle et limite la façon dont les processus évoluent sur le système (identification et contrôle d'accès aux ressources).

1.2 Les choses intéressantes à faire

Il existe à partir de là deux raisons essentielles pour vouloir se cacher sur une machine : celle du pirate, qui a pris une machine et souhaite y laisser une porte dérobée, et celle de l'administrateur système, qui souhaite par exemple mettre en place un pot à miel, et veut être sûr de pouvoir accéder à tout ce qu'a fait le pirate.

Globalement, les besoins de ces deux types particuliers d'utilisateurs sont les mêmes : pouvoir effectuer certaines actions sans que l'autre ne s'en rende compte. Ceci inclut l'envoi et la réception de trafic réseau, l'espionnage des actions de l'autre, le masquage de ses propres actions, ...

En voici quelques exemples :

- récupération des frappes claviers ou de toute autre donnée ;
- lecture, modification, suppression et dissimulation de fichiers ;
- écoute du réseau ;
- émission de données sur le réseau ;
- dissimulation de trafic réseau ;
- ...

1.3 Où se placer ?

Selon les actions que l'on souhaite réaliser à l'intérieur du noyau, il existe plusieurs endroits où il peut être intéressant de venir se positionner. Le schéma qui suit présente quelques-unes des possibilités de *hook*, dont certaines seront détaillées par la suite :

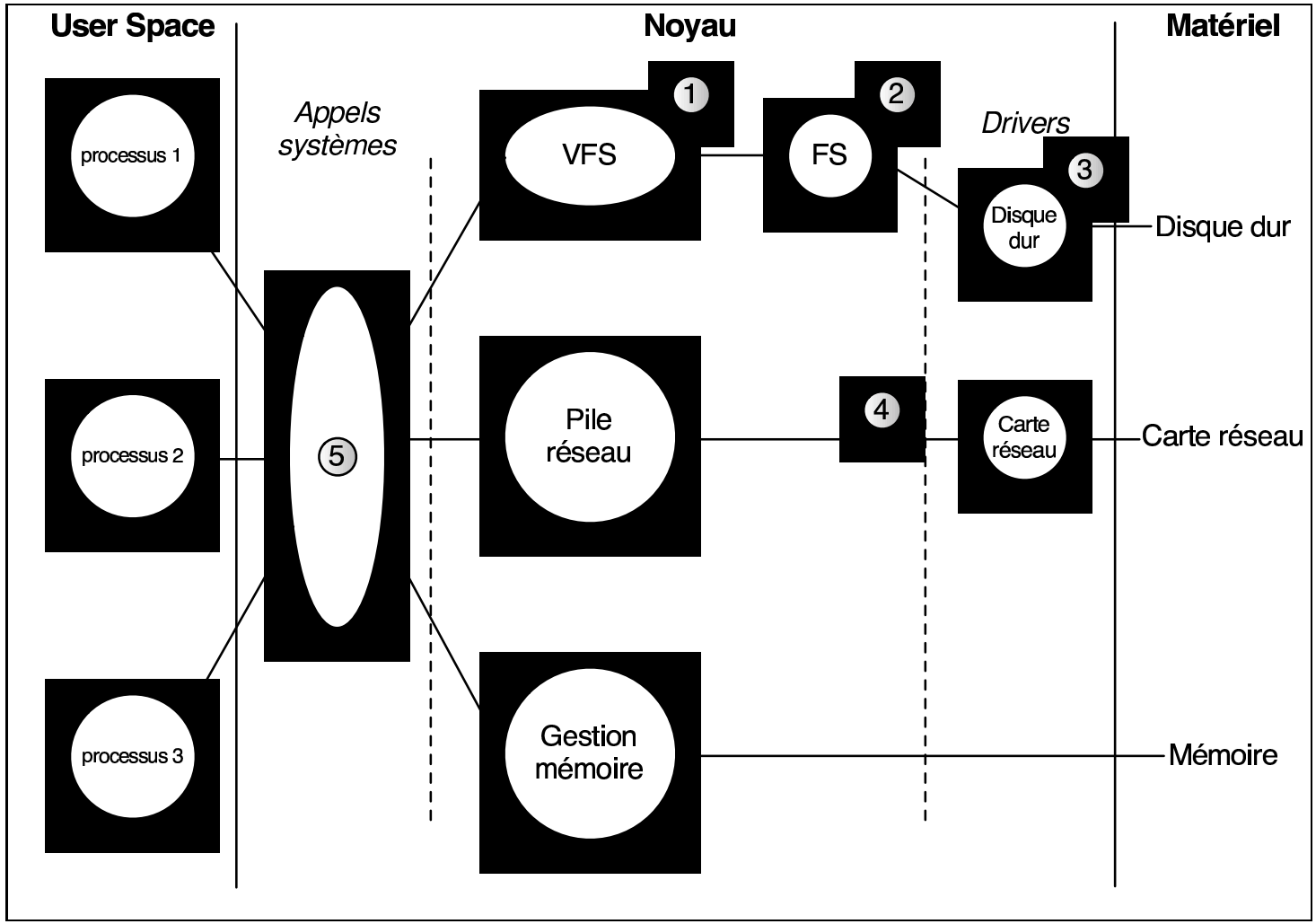


FIG. 1. Où se *hooker* dans le noyau

1. **VFS** : un hook à cet endroit permet par exemple de masquer certains répertoires ou fichiers aux utilisateurs.
2. **Système de fichier particulier** : à la différence du VFS, on est limité à des interactions avec un seul type de système de fichier (*e.g.* `ext2fs`).
3. **Driver** : les drivers sont la partie du noyau la plus proche du matériel. Ainsi, se placer à ce niveau pour un périphérique réseau permet d'envoyer et de recevoir des trames directement sur ce contrôleur particulier. On n'a par contre pas accès aux cartes réseau gérées par un autre driver.
4. **Entre la pile réseau et le driver** : dans le cas du réseau, il s'agit de la dernière couche indépendante du matériel utilisé. Elle offre donc un point d'accès global sur tous les périphériques réseaux présents sur le système.
5. **Appels système** : ils fournissent l'interface entre les processus en espace utilisateur et le noyau. Ils permettent à ces processus d'exécuter des actions qui nécessitent de travailler en mode kernel ou d'avoir d'autres types de droits particuliers (accès au matériel, communications interprocessus, ...). Ce hook est intéressant car il offre notamment un accès à l'ensemble des données transitant lors des demandes de lecture (au clavier, dans des fichiers, sur le réseau, ...).

Cette liste est loin d'être exhaustive ; il est par exemple possible de se placer au niveau du gestionnaire d'interruption pour récupérer les frappes clavier ou encore d'utiliser l'interface `bpf` pour avoir accès au trafic entrant. Nous reviendrons sur ces cas plus loin.

1.4 Comment entrer ?

Ce document n'a pas pour but d'évoquer les manières de s'introduire dans un système (dans le cas de l'administrateur, le problème ne se pose pas). Nous nous intéresserons uniquement à la manière d'injecter du code au niveau du noyau une fois cette étape d'entrée effectuée.

Selon les cas, la façon de procéder va être différente : il est en effet plus difficile de changer le noyau quand on n'est pas le propriétaire *légitime* de la machine cible. Plutôt que de modifier le noyau, il est envisageable de corrompre quelques binaires essentiels, mais ceci est facilement contournable car l'*adversaire* peut amener les siens.

Prenons un exemple simple : vous êtes un pirate et vous voulez vous assurer que vous n'êtes pas sur un pot à miel, ou bien vous êtes un administrateur, et soupçonnez que la machine soit corrompue. Vous souhaitez par exemple connaître le trafic réseau en cours. Quel que soit le programme que vous allez utiliser pour cela, il fera appel au noyau. Si celui-ci lui masque certains flux, vous ne pourrez pas y avoir accès. Les fichiers de log de votre porte dérobée peuvent également être rendus presque totalement invisibles.

Nous allons nous intéresser à quelques méthodes permettant de récupérer des informations depuis le noyau en détournant les actions de celui-ci pour qu'elles offrent les renseignements souhaités. Dans un second temps, nous traiterons de l'envoi et de la réception furtifs de données sur le réseau.

2 Insertion du code

Nous devons donc placer notre code dans le noyau, mais comment faire ? Trois solutions majeures existent. Ces solutions sont très différentes, et le choix se fera donc en fonction des possibilités offertes par la configuration de la machine, de notre niveau technique, et selon que l'on est le propriétaire de la machine ou pas.

2.1 Remplacer le fichier du noyau

On peut tout d'abord remplacer le noyau (comprendre, le fichier du noyau sur le disque), et redémarrer la machine (on peut même penser à provoquer un plantage de la machine pour contraindre l'administrateur à la redémarrer). Il s'agit de la solution quand on est le propriétaire de la machine, et quand on peut la redémarrer (et il n'y a pas vraiment de raison, *a priori*, que l'on ne puisse pas redémarrer quand on met en place un pot à miel).

2.2 LKM

On peut aussi, si le noyau le supporte, utiliser les LKM ; c'est le plus simple lorsqu'on n'est pas vraiment maître de la machine. LKM signifie *Loadable Kernel Module*, et désigne les fameux modules du noyau. Ceux-ci permettent d'ajouter dynamiquement au noyau des fonctionnalités dont on n'a habituellement pas besoin tout le temps (*e.g.*, drivers pour une Webcam).

Cela devient d'un seul coup plus facile, mais aussi plus visible. En effet, des commandes comme `lsmod` sous Linux, `kldstat` sous FreeBSD ou `modstat` sous NetBSD et OpenBSD existent pour connaître la liste des modules actuellement chargés. Il est toutefois possible de modifier cette liste de manière à masquer la présence d'un module donné. Pour cela, il suffit d'insérer un second module qui va se contenter de supprimer une entrée de la liste. Cependant, nous allons voir dans la partie suivante que ce n'est pas forcément une solution miracle.

Une autre solution (truff, Phrack 61) est de profiter des modules existants sur le système et d'*infecter* l'un d'entre eux de manière à lui ajouter les fonctionnalités qui nous intéressent. Il suffit pour cela de disposer d'un module effectuant les actions souhaitées, d'injecter le code de celui-ci dans le module présent sur le système (réalisable à l'aide de `ld`). Il ne reste alors qu'à modifier la table des symboles (section `.symtab`) de l'objet ELF (Executable and Linking Format) obtenu pour faire en sorte de remplacer, par exemple, la routine appelée à l'initialisation par une des notres. Comme de nombreux systèmes utilisent le format ELF pour leurs modules (Linux, ⋆BSD, ...), ceci est réalisable sur un grand nombre de machines. Au final, le module dans lequel se trouve le code étant l'un de ceux de l'utilisateur, il sera beaucoup plus furtif et pour peu que celui-ci fournisse un service important (comme celui d'une carte réseau, par exemple) il sera chargé de manière presque certaine.

Quand on a accès à la mémoire du noyau, on a souvent besoin de la table des appels systèmes. Quand ce symbole n'est pas exporté, on peut, comme dans Sebek-Linux, parcourir tout une plage de mémoire à la recherche de ce symbole

(on cherche un pointeur `p` tel que `p[__NR_close] == sys_close`, par exemple). Lorsqu'on a besoin de certains symboles, il est également parfois utile, sous Linux, de jeter un œil au fichier `/boot/System.map`.

Notons que le mécanisme securelevel des BSD interdit (entre autres) l'insertion de modules dans le kernel après le boot du système (si `kern.securelevel` est passé à une valeur strictement positive).

2.3 `/dev/kmem`

La troisième option est d'utiliser les fichiers spéciaux `/dev/mem` et `/dev/kmem` afin de modifier le noyau à la volée (en mémoire) sans avoir recours aux modules. En effet, ceux-ci peuvent avoir été interdits, et puis ce sera de toutes façons beaucoup plus discret. Revers de la médaille, ceci est plus difficile techniquement, et le mécanisme des securelevel des BSD précédemment évoqué rend également inefficace ce procédé en interdisant l'écriture vers ces périphériques (toujours avec la même condition que `kern.securelevel` doit être passé à une valeur strictement positive). De la même manière, des patchs noyaux de linux comme *GrSecurity* permettent de limiter l'accès à `/dev/mem` et `/dev/kmem`.

Pour cette troisième option, vous devez avoir les permissions nécessaires pour écrire dans le périphérique `/dev/kmem` (au sens du système de fichier virtuel VFS) ; mais (dans le cas de linux) ce n'est pas suffisant, car un test[2] supplémentaire est effectué au niveau de la gestion des périphériques de type `mem` (dans le fichier `drivers/char/mem.c`, fonction `open_port`), pour vérifier que l'utilisateur est autorisé à effectuer des entrées / sorties directes. Dans le cas des BSD, c'est, comme nous venons de le voir, la valeur de `securelevel`, qui peut être un obstacle, même si les permissions sont supposées permettre l'écriture.

Depuis l'espace utilisateur, on utilise les fonctions classiques `open`, `lseek`, `read`, `write`, `close`, ... pour manipuler ce périphérique.

Comme on a besoin d'obtenir certaines adresses de symboles, sous Linux, le réflexe est de se tourner vers `/boot/System.map`. Cependant, comme il arrive que ce fichier soit supprimé par un administrateur parano (et visiblement, à juste titre...), on retombe dans le même *Système D* que dans le cas des LKM, en pire, vu que l'on n'a aucun symbole exporté.

On commence donc par chercher la table des appels système, qui nous donnera beaucoup d'informations. Pour cela, une méthode classique[3] consiste à demander la table des interruptions, puis à en déduire l'adresse de la fonction de traitement de l'interruption `0x80`, qui contient un appel (`CALL`) à l'adresse de la table plus un offset qui dépend de l'appel système à traiter. On cherche alors cet appel et on en déduit l'adresse tant convoitée.

Un problème auquel on est confronté, auquel on n'avait pas à faire face dans le cas des LKM, est celui de la réservation de mémoire pour stocker notre code. Il faut pour cela localiser la fonction `kmalloc`. Une solution (nous n'en avons

[2] (*capable(CAP_SYS_RAWIO)*)

[3] Cette méthode est entre autres décrite dans le phrack 58 (Volume 0x0b, Issue 0x3a, Phile #0x07 of 0x0e, signé sd `<sd@sf.cz>` et devik `<devik@cdi.cz>`).

pas trouvé d'autre) est de se tourner vers la recherche de motifs sur le code (ou le code supposé) de la fonction. C'est assez fastidieux et dépend fortement du système et de sa version.

Un endroit où placer son code tout en évitant que celui-ci puisse être effacé de manière intempestive par le noyau (lors d'un besoin de mémoire) est proposé par Silvio Cesare dans le cas Linux. La zone mémoire permettant de servir les requêtes effectuées par appel à `kmalloc` démarre sur un début de page. La zone qui la précède en mémoire est celle réservée au noyau à la compilation. Celle-ci ne termine pas forcément une page. Un *padding* est donc nécessaire pour que le pool réservé aux appels à `kmalloc` débute sur une nouvelle page. Cette zone de *padding*, dont le noyau ne se sert pas est donc utilisable, sans risque d'être réclamée par celui-ci et ceci, sans nécessité d'allocation.

Enfin, il est possible de faire certaines choses amusantes sans avoir besoin de réserver de la mémoire. Il s'agira par exemple de supprimer le test effectué, lors de l'appel système `open`, pour vérifier que l'utilisateur qui fait cette demande dispose bien des permissions nécessaires.

Prenons l'exemple d'un noyau Linux 2.4.26 sur architecture i386. La vérification est faite après un appel à la fonction `permission` dans la fonction `open_namei`. Dans les conditions de notre expérience, l'analyse du code assembleur révèle que pour atteindre notre objectif, on peut remplacer une instruction `JNE` par une série de six `NOP` (l'instruction `JNE` et son argument occupent dans ce cas en effet six octets en mémoire). La fonction `sys_open` est appelée lorsqu'un programme en espace utilisateur effectue un appel système `open`. On cherche la troisième instruction `CALL`, ce qui nous donne l'adresse de la fonction `filp_open`. Dans cette fonction, on cherche le premier `CALL`, qui est un appel à la fonction `open_namei`. Une fois dans cette fonction, il faut chercher le troisième `CALL`, qui est celui qui appelle la fonction `permission`. En restant dans le code de `filp_open`, le saut conditionnel `JNE` qui suit (peu d'instructions après) est celui que l'on veut supprimer. On remplace alors les six octets par six `0x90`. Il n'y a plus de vérification de permissions lors d'un appel à `open` (jusqu'au prochain reboot bien entendu).

Ces exemples montrent, nous l'espérons, la relative difficulté induite par l'utilisation de `/dev/kmem` pour installer une backdoor à la volée. En contre-partie, quand ceci est bien fait, c'est beaucoup plus élégant, efficace et furtif.

3 Récupération des données depuis le noyau

Comme nous l'avons déjà évoqué précédemment, la modification des appels systèmes est une méthode privilégiée pour récupérer des données provenant des processus. Les exemples présentés ici permettent de se rendre compte dans le cas particulier du détournement des appels systèmes des techniques qui permettent l'insertion de code dans le noyau. Certaines de ces techniques ne s'appliquent d'ailleurs pas uniquement au détournement des appels systèmes mais permettent aussi de placer du code à d'autres endroits dans le noyau.

3.1 Détournement d'appels système

Si l'on a choisi d'écrire un module, on va maintenant vouloir détourner quelques appels système ou quelques fonctions clefs, que ce soit pour espionner les utilisateurs, les applications, ou les pirates dans le cas d'un honeypot.

Si l'on a choisi de modifier le noyau au moyen d'un patch, ce problème ne se pose pas, car le code est directement ajouté à l'intérieur des fonctions.

Table des appels système Lorsqu'on veut détourner un appel système (au hasard, `read`), on peut modifier l'entrée qui le concerne dans la table des appels système ; cependant, ceci peut être repéré très facilement.

Dans le cas de FreeBSD, si l'on veut détourner l'appel système `read`, on peut insérer un module contenant une fonction dont le prototype est :

```
static int false_read (struct thread *td, struct read_args *uap)
```

Notre module modifie la table des appels système dans le cas `MOD_LOAD` du `handler`, c'est à dire dans le code exécuté lors du chargement du module :

```
sysent[SYS_read].sy_call = (sy_call_t *) false_read;
```

A chaque fois que l'appel système `read` est appelé, le système interroge la table des appels systèmes, trouve l'adresse de notre fonction, et l'utilise. Notre fonction `false_read` peut même faire appel à la fonction originale, dont le code n'est pas modifié.

Mais cette méthode possède des parades simples : il suffit en effet de regarder la table des appels système, et de la comparer si possible avec une table *saine*. En effet, dans ce cas, les adresses des fonctions pointées par la tables sont proches et se suivent. Une fois la table modifiée, la fonction pointée possède une adresse différente des autres.

Utilisation d'un `JUMP` Ainsi, il peut être judicieux de ne pas modifer la table des appels systèmes, puisque ceci est facilement repérable. Une autre technique consiste alors à écraser le début du code de la fonction à remplacer par un `JUMP` vers du code qui effectue le même travail que la fonction concernée, et un petit supplément (log, analyse des données pour éventuellement donner plus de privilèges ou pour exécuter du code, envoi sur le réseau, ...).

Prenons un exemple pour FreeBSD[4]. Ces lignes doivent être placées dans le cas `MOD_LOAD` du `handler` (code d'initialisation exécuté lors du chargement du module, équivalent de `init_module` sous Linux) :

```
int *jump_address;
char *read_address = (char *) read;

jump_address = (int *) (read_address + 1);
*read_address = 0xE9;
*jump_address = (int) false_read - (int) read - 5;
```

[4] Cet exemple provient de la première version de Sebek pour FreeBSD

Nous sommes sur une architecture compatible i386 (`0xE9` signifie `JUMP`). Nous avons dans notre code une fonction dont le prototype est :

```
static int false_read (struct thread *td, struct read_args *uap)
```

Ce code va donc modifier, lors de l'insertion du module, le code de la fonction `read()` pour placer au début un `JUMP` vers une adresse relative correspondant à la différence entre les adresses en mémoire des fonctions `false_read()` et `read()` moins 5 (la taille en octets de l'instruction `JUMP` et de son argument). Cela signifie que lorsque la fonction `read()` est appelée, le `JUMP` est suivi, et le code de la routine `false_read()` est exécuté. Le reste du code de la fonction `read()` n'est jamais exécuté. Bien évidemment, notre nouvelle fonction ne peut plus réutiliser facilement la routine `read()` originale, comme c'était le cas lorsqu'on se contentait de modifier la table des appels systèmes. Il est ainsi nécessaire que notre fonction contienne tout le code de la fonction dont le début a été écrasé pour fournir les fonctionnalités désirées de celle-ci.

Pour la détection, on peut chercher à lire le début du code assembleur de quelques fonctions essentielles ; ceci se fait par exemple très simplement en insérant un module qui lit et affiche le code d'une fonction. Exemple simpliste, toujours pour FreeBSD, permettant de réaliser cela :

```
int i;
char *code = (char *) read;
for(i=0; i<10; i++)
  printf("%x ", code[i]);
printf("\n");
```

Amélioration de la furtivité Une première idée consisterait à placer un `JUMP` ou un `JUMP` avec condition (`JNE`, `JGE`, ...) plus discret, non pas au début de la fonction initiale mais au début d'une fonction légitimement appelée par cette fonction initiale. Ainsi, cette démarche oblige la personne souhaitant se rendre compte de cette modification à descendre encore d'un niveau d'appel pour réaliser ses tests (scruter le début de chaque fonction appelée par la routine initiale). Mais il faut bien se rendre compte que ceci n'est possible que lorsque la fonction initiale appelle une autre fonction avec des paramètres similaires aux siens. C'est le cas de `read` qui appelle `dofileread` avec les mêmes paramètres.

Il existe bien sûr d'autres méthodes plus compliquées permettant d'accroître la furtivité des modifications apportées.

3.2 Détournement d'interruptions

Toujours dans la famille récupération de données, un autre exemple d'endroit intéressant où se placer est le gestionnaire d'interruption. Une interruption est un événement, qui, lorsqu'il survient, a pour conséquence automatique l'exécution d'un code situé à un endroit défini. Il existe plusieurs types d'interruptions :

les interruptions synchrones au CPU, telles les exceptions et interruptions logicielles, et les interruptions asynchrones, comme celles qui sont générées par les périphériques, par exemple la souris ou la carte réseau.

Tout comme pour les appels systèmes, il existe une table qui associe à chaque numéro d'interruption un pointeur vers une fonction du gestionnaire d'interruption. C'est l'IDT (*Interrupt Descriptor Table*).

Des techniques décrites dans les articles 4 et 14 du phrack 59[5] expliquent comment il est possible de détourner l'IRQ 1 qui gère les interruptions générées par le clavier (sur les architectures de type x86).

L'idée générale est de modifier l'adresse du gestionnaire d'interruption contenue dans l'IDT. Notre propre gestionnaire est invoqué à sa place, et peut alors mémoriser les lectures effectuées sur le port d'entrée/sortie du clavier, ainsi que sur le port d'état, nous permettant d'accéder au touches enfoncées ou relachées.

En fait, on se rend compte qu'il est tout à fait possible d'utiliser le même genre de techniques que lorsqu'on veut détourner un appel système. En effet, il existe des outils permettant en userland (à partir de `/dev/kmem`) d'afficher le contenu de l'IDT. On peut de la même manière ou à l'aide d'un module, modifier soit son contenu, c'est à dire le pointeur vers le gestionnaire d'interruption (cela impose de réécrire la partie assembleur du gestionnaire d'interruption, le gestionnaire intermédiaire), mais on peut aussi modifier l'adresse du gestionnaire réel (qui lui est écrit en C), que l'on peut retrouver car le gestionnaire intermédiare l'appelle au moyen d'un `call`. La technique du `JUMP`, décrite plus haut, est certainement elle aussi utilisable.

Par contre, le détournement d'interruptions présente un inconvénient majeur, il est propre à une architecture, et il n'est donc par conséquent pas portable.

3.3 Détection & contre-mesures

Si la *backdoor* est suffisamment bien faite, elle est presque impossible à détecter localement. Si l'on est un administrateur système et que l'on constate que le noyau est corrompu, le plus sage pour remettre en place les machines est (après avoir sauvegardé les données pour l'*autopsie*) de tout réinstaller *from scratch*. Mais si l'on est un vil pirate convaincu d'être sur un *honeypot*, on voudra peut-être supprimer Sebek (par exemple), jouer un peu avec la bête, et prendre ainsi l'administrateur à son propre piège.

Dans le cas où il est permis d'insérer des LKM, rien de plus simple. On va se contenter de recoder les fonctions sensibles telles qu'elles sont supposées être, puis modifier la table des appels système et/ou utiliser la technique du `JUMP` (voir 3.1).

Si l'on ne peut pas utiliser les LKM, il va falloir se tourner vers des techniques beaucoup plus délicates, par exemple celle consistant à lire (pour détecter), et éventuellement écrire (pour mettre la *backdoor* hors d'état de nuire) directement dans `/dev/kmem`.

[5] L'article 14 du phrack 59 présente surtout d'autres endroits dans lesquels il est possible de se placer pour effectuer un *keylogger*, autres que les interruptions

Que ce soit pour les LKM ou pour le périphérique `/dev/kmem`, les techniques sont les mêmes dans ce cas que dans celui de l'insertion du code (voir la partie 2).

Pour le cas des BSD, nous avons vu que le système peut passer le `securelevel` à `1` lors du démarrage (le *super-utilisateur* peut aussi le faire à tout moment), ce qui est une action irréversible (seul `init` peut diminuer cette valeur). Celle-ci empêche, entre autres, d'insérer un module, ainsi que d'écrire dans les périphériques `/dev/mem` et `/dev/kmem` (pour les effets précis des différentes valeurs possibles de `securelevel`, voir, selon le système, `securelevel(7)`, *OpenBSD Reference Manual*, `init(8)`, *FreeBSD System Manager's Manual*, et `init(8)`, *NetBSD System Manager's Manual*).

Notons que si OpenBSD passe le `securelevel` à `1` par défaut, ce n'est pas le cas de FreeBSD ni de NetBSD. Cependant, il est possible de configurer le système pour qu'il le fasse (cela se passe dans le fichier `/etc/rc.conf` qui permet d'écraser les valeurs par défaut du fichier `/etc/defaults/rc.conf`). Cette manipulation est simple, et rend la machine nettement plus sûre ; même si cela oblige parfois à réamorcer le système, et, sauf cas exceptionnel (pot à miel, programmation LKM, système haute disponibilité, ...), il n'y a pas de raison valable pour être en `securelevel -1` (*permanently insecure*).

4 Communication depuis le noyau

Après avoir présenté dans la section précédente quelques moyens d'obtenir des informations sur le système, nous allons nous intéresser dans cette section à la manière de faire sortir ces informations mais aussi d'en recevoir de l'extérieur.

4.1 Emettre des trames

Selon la configuration dans laquelle on se trouve, on agira juste au dessous du niveau deux (ethernet ou ppp par exemple) ou trois (ip). La création d'un paquet dans le noyau est réalisée par encapsulations successives, des couches supérieures vers les couches basses ; il y a donc plusieurs couches parcourues lorsqu'une application userland souhaite émettre une trame sur le réseau.

Dans le cas de Sebek, nous agissons directement au dessous de la couche ethernet, juste au dessus du driver, sans passer par les couches supérieures (l'inconvénient est que son utilisation est limitée à une interface ethernet), ce qui permet notamment de s'abstraire des règles de filtrage qui peuvent être en place sur la machine et qui n'affectent que les trames construites par l'intermédiaire de la pile réseau.

Se placer plus haut peut simplifier la création des paquets, mais le cheminement de ceux-ci est alors grandement dépendant de la configuration réseau de la machine (table de routage, firewall, ...) et cette solution, qui a été retennue pour l'implémentation de Sebek-OpenBSD, devient par conséquent moins furtive et moins efficace.

Plus précisément, une partie de la furtivité s'obtient en forgeant soi-même les paquets contenant les données et en insérant ces trames de niveau deux valides directement dans la file d'envoi de l'interface considérée. Par exemple sous BSD, après avoir créé la trame dans la structure adaptée du noyau (une chaîne de structures `mbuf`), celle-ci est placée par la macro `IF_ENQUEUE` dans la file de l'interface. Ensuite, son émission sur le réseau est réalisée à l'aide d'une fonction spécifique au matériel (le pointeur vers cette fonction est stocké dans la structure de l'interface).

4.2 Recevoir des commandes

Nous allons voir deux façons différentes de recevoir des commandes. Il s'agit de faire remonter à notre code (dans le kernel) des données à traiter. Nous n'allons pas aborder ici ce que l'on peut faire de cette communication, mais plutôt des méthodes pour l'établir.

On souhaite pouvoir recevoir des commandes qui proviennent du réseau. On pense donc *a priori* à se placer dans des fonctions de la pile réseau du système. Plus on se place bas, plus on évite ce qui pourrait causer des ennuis (*firewall* par exemple).

Pour éviter que ce trafic soit repéré, on peut placer les commandes soit dans du trafic apparaissant légitime (cela dépend de l'utilisation qui est faite de la machine), soit masquer le trafic qui la contient (il faudra alors se tourner, en plus de la pile réseau, vers `bpf`, voir 4.3). Cependant, si l'on se place au niveau du driver, cela ne peut être intéressant que lors d'une attaque très ciblée (en contre-partie, ce sera sans doute le plus efficace).

Une solution alternative est encore de se placer dans la fonction `read`. Il faut, pour que cela fonctionne, qu'un programme dans l'espace utilisateur récupère les paquets, c'est à dire avoir un service qui les recoive, ou bien un programme qui les dumpe (de type `libpcap`). L'intérêt est que l'on peut alors cacher les commandes, en limitant les traces dans les logs. Par exemple, le démon OpenSSH ne loggue rien tant qu'aucune tentative d'authentification n'est effectuée. On peut donc envoyer la commande à la place de la ligne de version qui débute l'échange ssh.

4.3 Masquer notre trafic

La figure 2 décrit l'organisation de la pile réseau sur les BSD. Les paquets sont créés dans les couches hautes et sont passées au couches inférieures, jusqu'au driver. Ce dernier duplique les paquets entrants et sortants, pour les passer à bpf. Son rôle est de pouvoir analyser le traffic circulant sur l'interface, en appliquant des filtres qui correspondront ou pas à certains paquets que l'on désire analyser. Ce trafic est perçu par les outils utilisant, par exemple, la fameuse `libpcap`[6].

De la même manière que pour recevoir des commandes, il convient entre autres de se placer dans la fonction `bpf_filter()` pour cacher les paquets devant

[6] Une librairie pour la capture de paquets, utilisée entre autres par `tcpdump`.

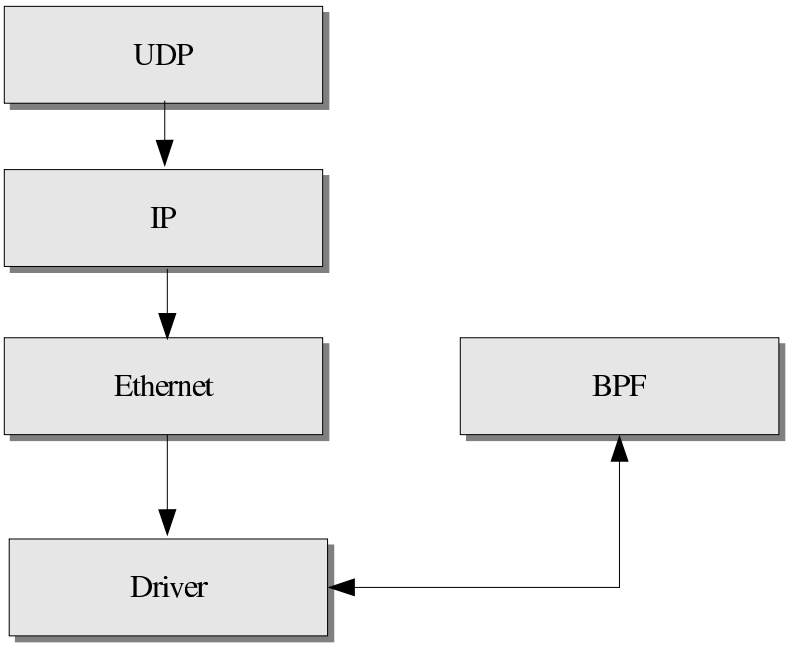


Fig. 2. Placement de bpf par rapport à la pile réseau.

être furtifs. Il suffit pour cela de faire retourner cette fonction avant qu'elle n'agisse, si le paquet doit être masqué.

Cependant, cela n'est pas vraiment suffisant pour bien faire, et il va falloir se placer plus bas. En effet, le paquet, lorsqu'il est soumis pour approbation à `bpf_filter()`, a déjà été comptabilisé (on peut voir le compteur à la fin d'une session `tcpdump` sous la forme `x packets received by filter, y packets dropped by kernel`).

Par exemple, dans les codes de Sebek-⋆BSD que nous avons développés, le filtre se trouve dans les fonctions du fichier `bpf.c` faisant appel à `bpf_filter()`, et qui comptabilisent les paquets. Juste avant d'incrémenter le compteur et d'appeler `bpf_filter()`, on vérifie si le paquet doit être filtré.

Conclusion

Que l'on soit dans la situation d'un pot à miel avec un outil de type Sebek, ou dans celle d'une machine potentiellement corrompue, la guerre entre celui qui veut altérer le noyau et celui qui souhaite un noyau *propre* (ou corrompu différemment, bien entendu) n'a aucune limite, si ce n'est le degré de paranoïa des intéressés, leur habileté technique, et leur imagination.

Si certaines techniques utilisées sont *fingerprintables*, il est difficile, voire impossible (particulièrement à distance), d'être absolument certain qu'un noyau est propre. De plus, on ne doit pas oublier que le fait que le noyau soit corrompu n'empêche pas, au contraire, que les binaires essentiels le soient aussi. Dans le cas d'une suspicion, il vaut mieux, avant de reprendre l'activité, tout réinstaller à partir de média sains.

La solution qui nous semble la plus efficace pour prévenir la mise en place de tels systèmes est l'utilisation de mécanismes semblables aux `securelevel` des BSD.

Pour terminer, nous tenons à remercier Laurent Oudot (Team RSTACK[7]), pour son encadrement dans l'écriture du port de Sebek2 pour les BSD. Merci aussi au French Honeynet Project[8].

[7] `http://www.rstack.org/`
[8] `http://www.frenchhoneynet.org/`

Sécurité des Téléphones Mobiles de Nouvelle Génération (Smartphones)

Maliha Rashid and Pascal Mercier[1] and Luc Delpha[2]

Cyber Networks,
Dpt Cyber Risk Consulting,
Suresnes (92), FRANCE
pmercier@cyber-networks.fr, ldelpha@cyber-networks.fr,
Page Web : http ://www.cyber-networks.fr

Résumé Les téléphones portables de dernière génération allient aux fonctionnalités des PDA une connectivité à Internet et aux réseaux sans-fil. Ces smartphones sont accessibles au grand public et leur utilisation étend le système d'information de l'entreprise au-delà des frontières du périmètre de confiance traditionnel. Cette présentation aborde les problématiques de sécurité engendrées par l'utilisation de ces terminaux mobiles de poche. Après une introduction aux fonctionnalités et à l'architecture des smartphones, les risques techniques associés à leur connectivité sans-fil (Bluetooth, GPRS, ...) et aux applications Java seront présentés. La présentation terminera avec les problématiques juridiques et une démonstration proof-of-concept.

1 Introduction

Les technologies sans-fil émergentes permettent aujourd'hui d'accéder à Internet, aux emails, aux applications d'entreprise et de synchroniser calendriers, contacts et autres applications, en temps-réel, à tout moment, à tout endroit.

Les téléphones portables de dernière génération combinent ces technologies aux systèmes d'exploitation dédiés et sont dotés de fonctionnalités multimédia avancées, d'où le terme smartphone.

Les smartphones facilitent le flux d'information à travers des réseaux hétérogènes et des domaines à risques. Ce flux d'information présente des risques pour le propriétaire du smartphone et le propriétaire de l'information en question.

Au vu de leur popularité croissante, les smartphones doivent être pris en compte lorsque l'on considère un système d'information, et plus particulièrement lorsque l'on considère la sécurité d'un système d'information.

Après une introduction aux fonctionnalités et à l'architecture des smartphones (Symbian...), les risques techniques associés à la connectivité aux réseaux sans-fil (Bluetooth, GPRS...) et au code mobile malicieux (Java MIDP...) entre autres seront abordés.

Les problématiques juridiques seront ensuite traitées, en mettant l'accent sur les limites légales concernant le contrôle d'utilisation de ce type d'équipement, et le partage de responsabilité juridique.

1.1 Pourquoi les Smartphones

Etant donné leur attrait gadget et leur accès grand public, les smartphones sont de plus en plus convoités. Puisque les smartphones sont principalement utilisés et vendus pour la téléphonie, ils ciblent une population plus large que les PDA (Personal Digital Assistant ou Assistant Personnel Numérique) classiques.

Bien que les smartphones soient principalement acquis pour un usage personnel, leurs fonctions d'agenda et de client mail ont tendance à être utilisées à des fins professionnelles. Dans un contexte professionnel, il est difficile de contrôler l'utilisation des smartphones à cet effet. Cette difficulté est due aux limitations pratiques et légales, et persiste même lorsque le smartphone est acquis à des fins professionnelles étant donné le caractère fortement personnel d'interaction avec l'équipement.

1.2 Caractéristiques des Smartphones

Mis à part les communications à travers les réseaux GSM et GPRS, les smartphones permettent la navigation sur Internet et sont dotés de fonctions agenda, email et multimédia type écran couleur haute résolution, appareil photo numérique, lecteur MP3. De plus, ils supportent les applications Java.

Les smartphones permettent aux utilisateurs de synchroniser les données PIM (Personal Information Management : calendrier, contacts et tâches) et les emails en utilisant des protocoles de synchronisation tels que SyncML, HotSync, ActiveSync ou IntelliSync sur Bluetooth, IrDA et GPRS.

Les smartphones aujourd'hui sont équipés de systèmes d'exploitation dédiés, conçus pour l'optimisation des ressources. Les systèmes d'exploitation dominants sont Symbian, Windows Mobile et PalmOS. Quelques smartphones fonctionnant autour d'un noyeau GNU/Linux aujourd'hui (Motorola A760) commencent à voir le jour.

Symbian OS version 8.0 est basé sur un noyau temps-réel multi-tâche et multi-threading (variant EKA2). Il supporte les dernières architectures CPU, périphériques, mémoire interne et externe. Il est équipé de moteurs d'application de PIM, messagerie, navigation Internet, contrôle d'utilitaire et système, Java, Bluetooth, les jeux, les graphiques 3D (bibliothèques OpenGL)

1.3 Smartphones et réseau sans fil

La conception des smartphones est axée autour de l'échange et des flux d'informations, utilisant les technologies sans-fil récentes à cet effet. Les technologies sans-fil utilisées aujourd'hui vont des WPAN tels que Bluetooth aux WWAN tels que le GPRS. Ces protocoles sont détaillés ci-dessous.

Bluetooth Bluetooth est défini par une spécification *Core* et un ensemble de profils relatifs à différents usages.

Comme l'indique la figure 2, le core [2] définit les quatres couches basses suivantes et leurs protocoles associés :

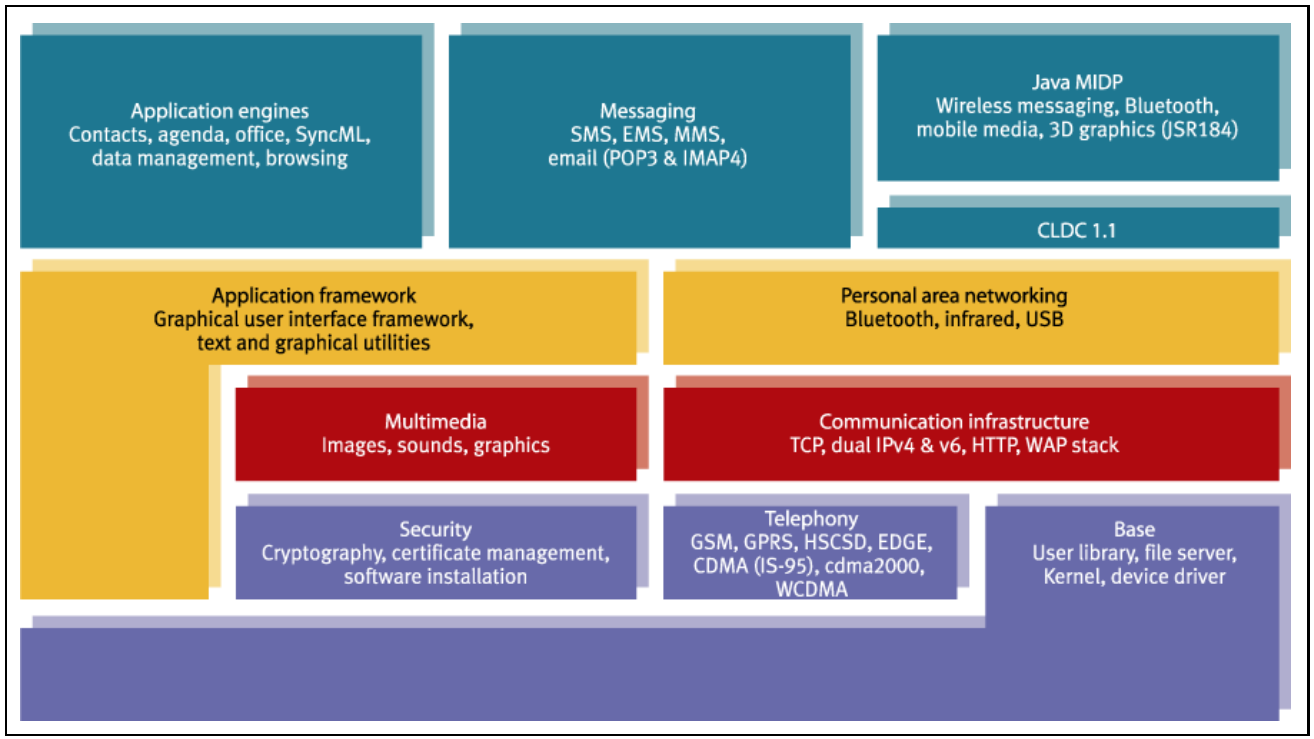


Fig. 1. Architecture of Symbian OS version 8.0 [1]

- Radio
- Baseband
- Link Manager
- L2CAP (Logical Link Control and Adaptation Protocol)

Le core couvre aussi :

- SDP (Service Discovery Protocol)
- Les prérequis généraux pour les profils : GAP (Generic Access Profile)

Les profils que l'on retrouve dans les smartphones sont :

- Service discovery application profile
- Synchronisation profile
- Generic Object Exchange Profile

Ce dernier profil utilise le protocole OBEX (Object Exchange) conçu par l'IrDA Association [3]. Le protocole OBEX est un équivalent binaire du protocole http, permet l'échange d'objets à travers des fonctions de Push (méthode PUT) et Pull (méthode GET).

Etant donné que l'intéropérabilité des équipements est prioritaire dans Bluetooth, la spécification n'est pas explicite quant à l'implantation.

GPRS Le GPRS s'appuie sur l'infrastructure GSM existante, permettant des débits plus élevés à travers un backbone IP à commutation par paquets.

Les réseaux GPRS permettent une connexion permanente à Internet, comme l'indique la figure 3, à travers deux éléments principaux :

- Le SGSN (Serving GPRS Support Node) pour :
  - Superviser l'état du terminal mobile et tracer ses mouvements
  - Etablir et gérer les connexions data entre le terminal mobile et le réseau destination
- Le GGSN (Gateway GPRS Support Node) qui constitue le point de raccordement du domaine GPRS au réseaux de données externes tels que Internet et les intranets d'entreprises.

Un parefeu est placé entre le GGSN et le réseau de données externe pour protéger le domaine GPRS d'attaques en provenance d'Internet.

Les tracés rouges dans la Figure 3 montrent l'interconnexion entre le domaine GPRS et les réseaux de données externes.

2 Les Risques

Les smartphones, que ce soit par leurs caractéristiques propres, leur utilisation (ou mauvaise utilisation) ou les technologies associées à leur utilisation, présentent des risques.

2.1 Risques liés aux caractéristiques propres des smartphones

Comme expliqué ci-dessus, les smartphones sont équipés de systèmes d'exploitation dédiés. Cela en soi induira de nouveaux risques tels que l'émergence de failles de sécurité et de bogues, principalement dûs à l'architecture complexe

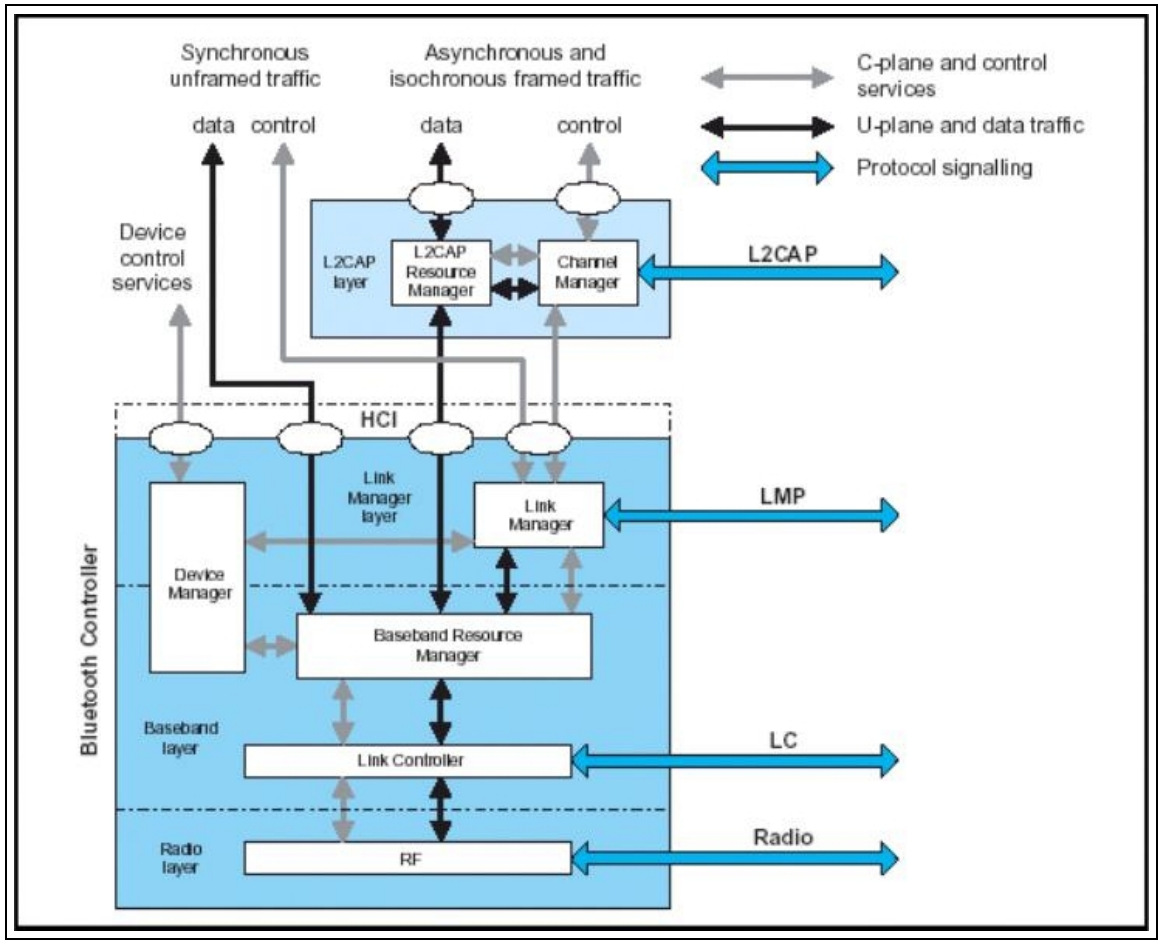


Fig. 2. Core specification 1.2

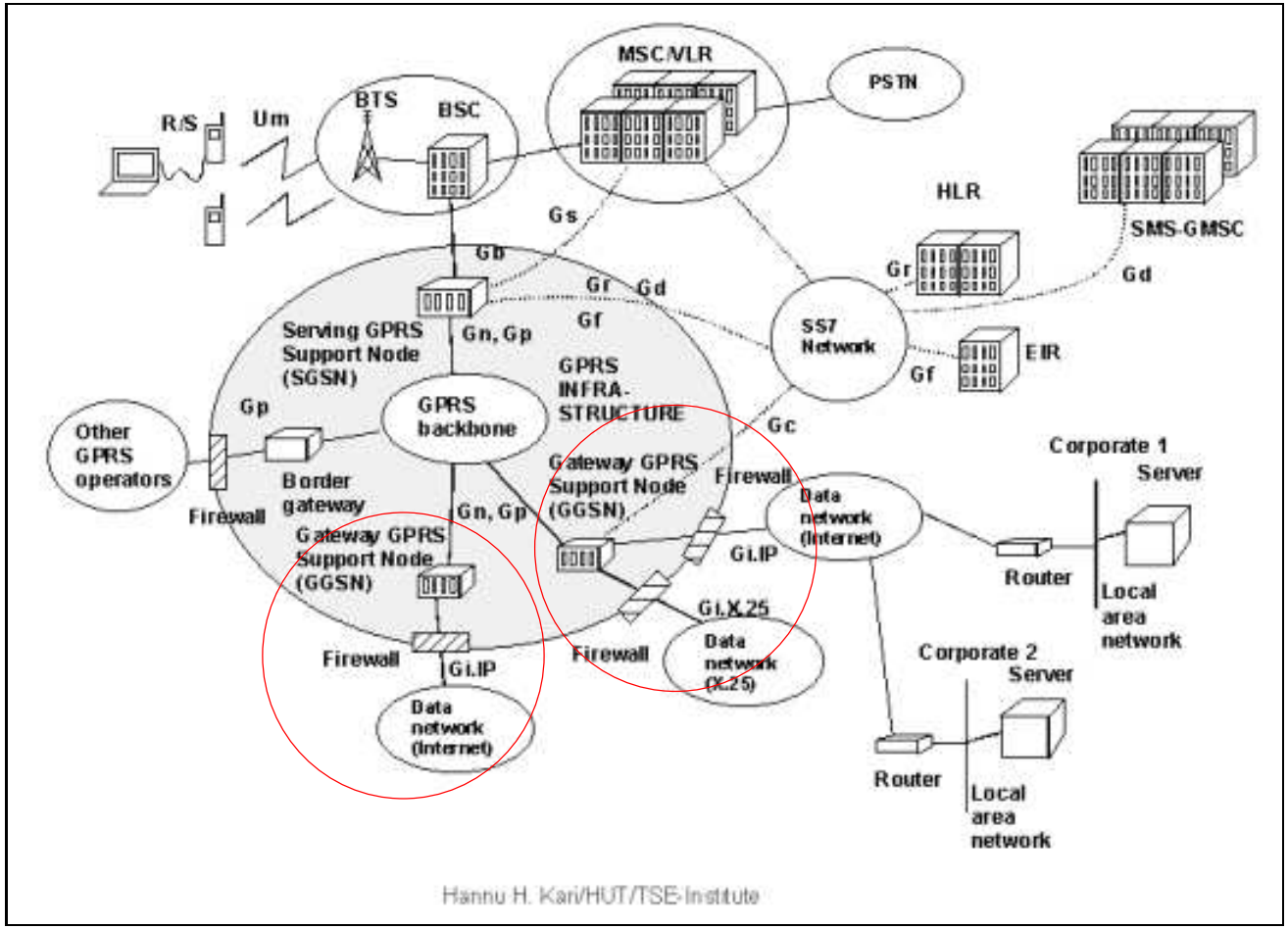


Fig. 3. General Packet Radio Service Architecture

des ces systèmes d'exploitation. Par exemple, des bogues liés à l'implantation de Java MIDP 2.0 ont été documentés dans [4]. Des bogues sur les smartphones sous Windows ont également été documentés [5]. Il est possible d'exploiter ces bogues afin de bloquer les smartphones et provoquer un reset. Cela effacerait les données stoquées sur le terminal. Les vulnérabilités système seront de plus en plus présentes sur les smartphones, comme pour les ordinateurs, au fur et à mesure que les systèmes d'exploitation des smartphones gagneront en sophistication.

Une autre problématique liée à la nature inhérente des smartphones se centre autour du contrôle d'accès et la sécurité des données.

Puisqu'il n'y a pas de chiffrement en natif pour protéger les données dans les équipements, l'information se trouve à disposition pour toute personne ayant un accès physique au terminal. Outre le code PIN, il n'y a pas, dans les smartphones les plus courants, de forme d'authentification en natif bien que des données personnelles voire confidentielles puissent y être stockées (voir Figure 4). Ceci présente un risque que les fabricants de smartphones devraient prendre en compte.

Même si un code PIN protège l'accès aux fonctionnalités du téléphone, les données restent accessibles. D'autant plus que les données sur la plupart des terminaux sont stockées sur des chipset flash (ou des cartes mémoire externes). Donc n'importe quelle personne ayant un accès physique au chipset peut contourner les contrôles d'accès et voler les données. Et enfin, le terminal peut facilement être détruit.

2.2 Les Risques Associés aux Utilisateurs

Si l'on en croit une étude récente menée par Pointsec Mobile Technologie :

Ce tableau montre que les utilisateurs de terminaux mobiles ont une forte propention à se servir de cet équipement pour stocker des informations extrêmement sensibles, sans même avoir conscience des risques auxquels ces données sont exposées.

Il est en effet possible pour un personne tierce d'accéder à ces données, que ce soit suite à un vole ou une perte de l'appareil, ou par le biais d'une compromission entraînant une prise de contrôle à distance.

Les conséquences peuvent aller de l'usurpation d'identité à la perte ou la modification d'informations (personnelles, professionnelles ou clients) sensibles. Losque l'ont sait l'importance que revêt de nos jours l'information, il est évident que ces conséquences peuvent s'avérer dramatiques.

A titre d'exemple, il y a quelque temps, le site eBay, spécialiste de la vente aux enchères sur Internet, proposait au plus offrant un Blackberry. Cet équipement, vendu $15.50, appartenait à un ancien vice-président de la banque **Morgan Stanley**. Or, il s'est avéré que cet appareil contenait une grande quantité d'informations confidentielles, telles que la liste des adresses mail de la plupart des acteurs principaux de la banque, ou encore des emails contenant les conditions de prêts de certains gros clients.

- 85 % des terminaux mobiles sont utilisés comme agenda dans un cadre professionnel
- 80 % des terminaux mobiles sont utilisés pour stocker des listes de contacts professionnels
- 79 % des terminaux mobiles sont utilisés pour stocker des listes de contacts personnels
- 75 % des terminaux mobiles sont utilisés comme agenda personnel
- 48 % des terminaux mobiles ont une utilisation multimedia (jeu, musique, ...)
- 35 % des terminaux mobiles sont utilisés pour visionner des documents
- 33 % des terminaux mobiles sont utilisés pour stocker des mots de passe (ou codes PIN)
- 32 % des terminaux mobiles sont utilisés pour recevoir ou envoyer des e-mails
- 25 % des terminaux mobiles sont utilisés pour stocker des information banquaires
- 25 % des terminaux mobiles sont utilisés pour stocker des informations propres à l'entreprise

Tab. 1. Etude sur l'utilisation des terminaux mobiles, réalisée par Pointsec Mobile Technologies

Après enquête, il est apparu que le Blackberry aurait dû être restitué par le vice-président à la banque afin que celle-ci efface toutes les données de l'appareil. Le vice-président pensait, lui, que cette procédure avait été effectuée à distance.

Enfin, il est facile de configurer un smartphone pour accéder à ses emails professionnels où à des données de l'entreprise. La plupart du temps, la synchronisation, par le biais d'un support ou d'une interface infra-rouge (IrDA), ne necessite pas une authentification préalable, ce qui peut donc constituer une menace pour le système d'information dans le cas où l'appareil mobile a été compromis (par exemple, par un virus utilisant une pièce jointe, et un mécanisme de reproduction par partage réseau)..

2.3 Les Risques Associés aux Réseaux Sans-Fil

La possibilité de connexion des smartphones à différents types de réseaux tend à les exposer aux risques inhérents à chacun d'entre eux. De plus, les smartphones sont désormais connectés en permanence aux réseaux 2.5 et 3G, ce qui augmente de manière considérable leur accessibilité. Ce phénomène ne fera d'ailleurs qu'augmenter avec l'arrivée des réseaux sans-fils 4G.

Les paragraphes suivants détaillent les risques induits par deux types de réseaux : Bluetooth et GPRS.

Bluetooth Le protocole Bluetooth met en oeuvre des mécanismes de sécurité. Cependant, la plupart de ces mécanismes ne sont pas mis en oeuvre ni activés par défaut dans les smartphones.

Dans la plupart des cas, les seuls modules de sécurité à être activés sont :
- Le mécanisme d'association (pairing)
- La possibilté de se mettre en *mode caché*

D'ores et déjà, des outils tels que Redfang [6] et BTscanner sont capable de scanner les appareils Bluetooth, même en mode caché. Pour ce faire, ils effectuent une attaque de type Brute-Force sur les 6 derniers octets de l'adresse MAC Bluetooth et appellent la fonction read_remote_name(). L'outil Redfang ne fonctionne que sous un environnement GNU/Linux.

D'autres outils tels que BTbrowsers ont été développés pour les smartphone Nokia 6600 et SonyEricsson P900 (les deux fonctionnant sous Symbian). Ces outils permettent à un utilisateur de lister les appareils environnants possédant le connectivité Bluetooth, puis d'accéder aux fichiers ou aux données PIM de ces périphériques.

Il ne faudra pas attendre longtemps avant de pouvoir disposer de portages d'outils tels que Redfang sur smartphones (l'arrivée des smartphones sous Linux ne faisant qu'accélérer les choses). L'émergence de tels outils ne fera qu'amplifier le récent phénomène de mode qu'est le *Bluejacking* (sorte de wardriving orienté Bluetooth). Ce type d'attaque consiste à définir un message en lieu et place de son identifiant Bluetooth, puis de l'envoyer à un appareil voisin lors d'une requête d'association. En soit, cette pratique ne parrait pas dangereuse. Cependant, si le message est construit dans le but d'inciter l'utilisateur victime à effectuer une action desactivant les mécanismes de sécurité, cela peut devenir très dangereux.

En complement à cela, différentes vulnérabilités ont récemment été découvertes dans l'implémentation du protocole Bluetooth sur certains smartphones, ce qui les rend encore plus vulnérables. Par exemple, un certain nombre de smartphones Nokia possédant le Bluetooth sont vulnérables à une attaque par Buffer Overflow, rendue possible grâce à l'envoie de messages OBEX mal formés (CAN-2004-0143)

Une autre vulnérabilité récente concerne les rapports de confiance établis lors d'une association. Dans certains cas de figure, cette relation de confiance persiste après le fin de l'association. A ce moment là, une personne mal-intentionnée peut accéder aux fichiers du smartphone de la victime sans son consentement.

La complexité du protocole tend à induire un nombre constant d'erreurs, qui seront autant de portes ouvertes pour les attaquants.

GPRS Les smartphones connectés au réseau GPRS sont exposés aux risques inhérents au backbone IP GPRS. La sécurité du Backbone GPRS repose sur les mesures prises par l'opérateur dans sa sécurisation des GGSN (Gateway GPRS Support Node). Si le GGSN est compromis, tous les clients de cet opérateur GPRS seront exposés aux différentes attaques provenant d'Internet.

Une attaque s'appuyant sur le mécanisme de translation d'adresse du réseau GPRS est décrite dans l'article pointé par le [10], et reste simple à mettre en oeuvre. Le même article décrit une autre attaque consistant à *flooder* une connexion GRPS avec un flux TCP inutile depuis Internet. Les fonctionalités que permettent la connectivité GPRS - toujours connecté, que ce soit pour la

reception de mails par push, la synchronisation, ou l'*OTA* (Over The Air) *provisionning* - sont précisemment celles par lesquelle le danger peut arriver.

Les smartphones qui permettent plusieurs contextes PDP (Packet Data Protocol) actifs présentent des risques. Le fait d'autoriser simultanemment des connexions publiques et privées peuvent conduire à une exposition des communications privées à travers le contexte des communications publiques. Symbian OS version 8 permet le support de plusieurs contextes PDP actifs simultanemment. Ceci permet à une personne mal-intentionnée d'obtenir l'accès au système d'information par le simple fait que le smartphone est connecté simultanemment au système d'information, et à un navigateur web ou à son système de messagerie.

2.4 Les Risques Associés aux Applications : Stand-Alone (Java MIDlets) – Browser-Based

Le développement d'applications pour les smartphones est un secteur en plein expansion, principalement pour les jeux et les applications visant le commerce mobile. Ces applications existent sous une forme autonome (*standalone*) ou basées sur l'utilisation d'un navigateur (*browser-based*, telles que les *applets*).

Java MIDlets La technologie J2ME (Java 2 Micro Edition) fournit un environnement d'execution Java optimisé pour les appareils disposant de peu de mémoire, ou d'une puissance de calcul limitée. Cet environnement est composé par le CLDC (*Connected Limited Device Configuration*) qui définit un ensemble d'APIs spécifiques aux smartphones, et le MIDP (*Mobile Information Device Profile*). Les applications Java autonomes construites pour les smartphones sont appelées MIDlets. Il existe à l'heure actuelle deux version de MIDP : MIDP 1.0 et MIDP 2.0

Les applications Java MIDP 1.0 n'ont comme mécanisme de sécurité qu'un accès limité par application aux ressources, un peu comme dans une *sandbox*. Cependant, la vérification du bytecode, trop gourmande en ressource pour un smartphone, est limitée. Il est en de même pour le *Security Manager* et pour un certain nombre de paquets spécifiquement orientés vers la sécurité. De plus, MIDP 1.0 ne supporte pas le protocole HTTPS, et ne permet que le protocole HTTP comme protocole réseau. Au regard de ces limitations, l'utilisation d'un smartphone équipé de MIDP 1.0 représente un grand risque par rapport à la confidentialité ou au caractère privée de l'information.

Etant donnée les possibiltés limitées et le manques de sécurité de MIDP 1.0, Les spécifications MIDP 2.0 tendent à ajouter une couche sécurité supplémentaire.

Java MIDP 2.0 introduit le concepte de *trusted MIDlets*. Si la MIDlet n'est pas reconnue comme étant digne de confiance, elle s'exécutera dans une sandbox ne lui permettant qu'un accès très restreint aux ressources systèmes. Dans le cas contraire, MIDP 2.0 octroiera une plus grand liberté d'accès aux ressource du système, permettant par là-même l'utilisation d'APIs pour le PIM, l'accès aux réseaux, les appels téléphoniques et la messagerie.

Il existe deux types de permissions pour les accès aux ressources sensibles :

- ***Allowed*** (sans autorisation explicite de la part de l'utilisateur)
- ***User*** (Un choix est soumis à l'utilisateur : *blanket*, *session*, *oneshot* ou *refused*)

Les permissions sont attribuées automatiquement en fonction du domaine auquel la MIDlet appartient. Si la MIDlet appartient au domaine de confiance (opérateur ou fabricant), les permissions seront fixées à ***Allowed***. Dans le cas contraire (tierce partie ou non-signée), le choix sera laissé à l'utilisateur.

Ces permissions sont laissées à la discrétion du développeur de la MIDlet et de l'utilisateur final. Un utilisateur final qui télécharge et installe une MIDlet gratuite acceptera vraisemblablement n'importe quel message et peut très bien, par exemple, autoriser l'accès à l'API PIM, donnant à la MIDlet l'accès aux contacts et au calendrier de l'utilisateur. La MIDlet pourrait alors transférer ces informations à un serveur distant.

Problèmes Liés aux Navigateurs Web Des problématiques de sécurité peuvent aussi survenir à travers l'utilisation de navigateurs web sur les smartphones. Si une version vulnérable d'Internet Explorer ou d'Opera, par exemple, est utilisée sur le smartphone, celui-ci pourra être compromis par les mêmes attaques touchant les PC, ciblées sur ces navigateurs.

3 Les Challenges

3.1 Les Problématiques Juridiques et politique de sécurité

Comme il a été vu précédemment, l'utilisation des smartphones par les employés étend le système d'information au delà du contrôle de l'entreprise. Le contrôle est également limité en raison de considérations légales, notamment lorsque le smartphone appartient à l'employé.

Dans ce cas, un certain nombre de possibilités s'offre aux entreprises :

- Interdire l'utilisation personnelle des smartphones
  - Non réaliste : de la même manière que le serait l'interdiction de l'usage personnel d'Internet
  - Impossible à contrôler et renforcer physiquement
- Poser des limites à l'utilisation personnelle des smartphones
  - En définissant de manière précise les interactions autorisées entre le smartphone et le système d'information
  - Impossible à contrôler et renforcer physiquement

Lorsqu'il y fuite d'infomations confidentielles lors de la perte ou le vol du terminal, la responsabilité de l'entreprise peut être engagée pour les pertes occasionnées, même si le smartphone appartient à l'employé. Cela est d'autant plus vrai lorsque l'entreprise n'a pas pris les mesures appropriées afin de se protéger contre les pertes en accord avec l'état de l'art.

Le premier pas à faire afin de contrer les risques induits par les smartphones dans l'entreprise est d'admettre leur existence et d'adapter la politique de sécurité à leur utilisation, et plus particulièrement aux utilisateurs des smartphones.

Lorsque les utilisateurs s'avèrent être de la direction générale, ce qui est le cas la plupart du temps, la clé est d'inclure ces utilisateurs dans la boucle, et de les informer non seulement des risques générés par leur utilisation des smartphones, mais aussi des limitations des smartphones et des technologies associées.

Les actions principales importantes à définir dans la politique de sécurité sont :

- La synchronisation (PIM, emails avec ou sans pièces jointes)
- L'utilisation des smartphones dans les lieux publiques (attention aux hotspots, désactiver Bluetooth)
- Le téléchargement et le transfert de fichiers du smartphone vers le système d'information

3.2 Un framework sécurisé pour les smartphones

Intégrer les smartphones en tant qu'éléments du système d'information implique un framework sécurisé qui inclut :

- Une solution d'administration centralisée
- Une authentification mutuelle entre terminaux et serveurs
- Un chiffrement de bout en bout
- Le renforcement des terminaux

Une solution d'administration centralisée permettra de mieux contrôler les smartphones ainsi que leurs interactions avec le système d'information, en utilisant l'authentification mutuelle entre le système d'information et le smartphone.

Le chiffrement des communications et des données stockées sur le smartphone combiné à une administration centralisée permettra d'atténuer le risque de fuites d'informations confidentielles. En cas de perte ou vol du terminal, la solution d'administration centralisée devrait permettre de retirer les données stockées sur le terminal.

Enfin, la sécurité des smartphones peut être renforcée de la même manière que les ordinateurs portables classiques, notamment à l'aide d'un antivirus et d'un firewall personnel.

3.3 Perspectives

La sécurité des smartphones est complexe. Son modèle implique une variété d'acteurs, allant des fabriquants de smartphones, opérateurs de télécommunications et concepteurs de protocoles d'un côté, aux utilisateurs et administrateurs de l'autre.

Pour que ce modèle fonctionne, tous ces acteurs doivent prendre en compte les risques associés à l'utilisation des smartphones, et de prendre les actions adaptées afin d'atténuer ces risques.

Cela est cependant difficile à coordonner, et il sera peut être nécessaire que ces acteurs soient encadrés par la législation dans leurs actions.

L'avenir de la sécurité des smartphones est un défi. En effet, les réseaux 4G, qui interconnecteront les réseaux WPAN, WLAN et WWAN, offriront-ils assez de sécurité ?

4 Conclusion

Les smartphones sont complexes aussi bien au niveau de leur conception que de leur architecture. Il en est de même pour les protocoles réseau utilisés par les smartphones. Cette complexité ouvre les portes aux erreurs d'implémentation et faiblesses structurelles. Cela rend les smartphones vulnérables à certains types d'attaques. Ces attaques sont simples à mettre en œuvre, et l'intérêt croissant pour les technologies associées aux smartphones mènera à la découverte d'autres faiblesses et multipliera les risques d'attaques. Afin de sécuriser un framework incorporant des smartphones au système d'information contre les attaques, la première action à mener par les entreprises est de faire prendre conscience aux utilisateurs des problématiques de sécurité des smartphones et de développer une politique de sécurité adaptée. La mise en place de mesures de sécurisation des intéractions entre les smartphones et le système d'information ainsi que le renforcement des smartphones peut alors atténuer les risques.

Références

1. `http://www.symbian.com/technology/symbos-v8x-det.html`
2. Bluetooth Core Specification 1.2
3. `http://www.irda.org/standards/specifications.asp`
4. `http://ncsp.forum.nokia.com/downloads/nokia/documents/Known_Issues_in_the_Nokia_6600_v1_0_en.pdf`
5. `http://www.cewindows.net/bugs/wm2003netsec.htm`
6. `http://www.atstake.com/research/tools/info_gathering/`
7. `http://www.atstake.com/research/reports/acrobat/atstake_war_nibbling.pdf`
8. `http://www.pentest.co.uk/documents/ptl-2004-01.html`
9. `http://www.bluestumbler.org/`
10. Candolin Lundberg, *Attacks on GPRS.*
11. Virtanen Kolsi, *MIDP 2.0 Security Enhancements.*

Présentation de l'outil Nmap-Stateful

Olivier Courtay

IRISA / Université de Rennes 1
olivier@courtay.org

1 Introduction

Nmap-Stateful est un projet d'extension de Nmap. Nmap possède un module de détection active de système d'exploitation distant (OS Fingerprinting) qui utilise des paquets soigneusement choisis et étudie les réactions des machines testées. Malheureusement, ces tests ne sont pas configurables car inclus dans le source du programme et ces tests ne permettent que l'étude des ports dans l'état TCP LISTEN ou CLOSE. Nmap-Stateful permet de lever ces contraintes et ainsi d'améliorer ou de concevoir d'autres utilisations que l'OS Fingerprinting. Après un rappel rapide du fonctionnement de Nmap nous introduisons les nouvelles fonctionnalités apportées à l'outil puis nous présentons le fonctionnement et l'utilisation de Nmap-Stateful à travers plusieurs expériences enfin nous conclurons sur les évolutions possibles de l'outil.

2 Rappels sur Nmap

Nmap [7] est un outil réseau créé par Fyodor, il contient un ensemble de fonctionnalités incluant : un scan de port, la détection de machines, l'OS Fingerprinting et récemment un module de détection de service. Dans ce papier, nous nous intéressons uniquement au module d'OS Fingerprinting [6] de Nmap, car Nmap-Stateful se base sur ce module.

Nmap dispose de neuf tests pour prendre une empreinte d'une machine distante :

- un test sur un port UDP fermé qui renvoie un message d'erreur ICMP.
- un test permettant à partir d'un échantillon de paquets TCP de faire des analyses statistiques sur la génération des *IP ID* et des *TCP sequence number*.
- quatre tests sur des ports TCP ouverts.
- trois tests sur des ports TCP fermés.

Ces neuf tests permettent de créer une empreinte de la machine en notant la valeur de champs intéressants dans les paquets TCP/IP, comme le montre cet exemple d'empreinte :

```
TSeq(Class=RI%gcd=<6%SI=<269E81A&>62D97%IPID=Z%TS=1000HZ)
T1(DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
```

```
T4(DF=Y%W=0%ACK=0%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=0%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=D0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134
%DAT=E)
```

est associée par Nmap à l'OS :

```
Fingerprint Linux 2.6.0 (X86)
Class Linux | Linux | 2.6.X | general purpose
```

Chaque ligne correspond à un test. Pour chaque test, un ensemble d'éléments est étudié, par exemple les flags TCP (*Flags*=*AS*) ou les options TCP (*Ops*=*MNNTNW*). Ces empreintes sont ensuite comparées à une base d'empreintes connues et associées à un système.

3 Fonctionnement de Nmap-Stateful

Nmap-Stateful se présente comme une extension apportée à Nmap : le code de base utilisé est donc celui de Nmap qui a été modifié pour ajouter les fonctionnalités voulues. Deux éléments montrent que Nmap peut être amélioré : la non-pertinence de certains tests et des retours d'expérience de l'outil Cron-Os [2]. En effet, Nmap positionne dans son premier test sur un port TCP ouvert (qui est l'un des tests les plus pertinents) le flag ECN. Le résultat ne diffère pas que ce flag soit positionné ou non mais les paquets contenant le flag ECN sont arrêtés par certains Firewall (CheckPoint FW-1 par exemple). Ce qui prouve que les tests de Nmap ne sont pas forcément les plus pertinents. Lors de l'utilisation de l'outil Cron-Os, la pile TCP/IP de la machine distante est amenée dans un état voulu avant d'êre analysé (par une analyse temporelle). Bien qu'il appparaît que ces états peuvent être sources de renseignements utiles, ils ne sont pas exploités par Nmap. Ces éléments montrent l'intérêt de concevoir un outil plus souple sur les tests, ayant la couverture des états TCP la plus large et la plus configurable possible.

Nmap-Stateful semble le seul outil qui rassemble toutes ces fonctionnalités. Pour comparaison, on peut citer les autres principaux outils d'OS Fingerprinting :

- Nmap bien sûr (la référence du domaine) mais qui devient un sous-ensemble de Nmap-Stateful.
- Xprobe [3] qui est un outil basé uniquement sur l'analyse des paquets ICMP. Cependant les messages ICMP sont souvent bloqués par les Firewall, ce qui empêche le fonctionnement de l'outil dans des milieux fortement filtrés.
- Cron-Os utilise les délais de retransmission TCP. Cron-OS utilise aussi plusieurs états TCP mais pas les caractéristiques TCP des paquets.

– pOf [4] est un outil d'OS Fingerprinting passif (il ne fait qu'écouter les paquets transitant sur le réseau). Il utilise trois types de paquets pour faire son analyse : les SYN, les SYN-ACK et les RST. Nmap-Stateful peut aussi utiliser ces paquets.

Contrairement à Nmap, les tests utilisés par Nmap-Stateful sont lus dans un fichier de configuration ce qui permet à un utilisateur de pouvoir les modifier et les adapter à ses besoins. Le comportement des cibles est mis sous forme d'empreintes comme dans Nmap, et un fichier de sortie peut être précisé pour stocker l'empreinte. En effet, comme on le verra par la suite, d'autres tests que de l'OS Fingerprinting sont possibles. Comme les tests ne sont plus fixés, on peut construire soi-même son fichier de référence d'empreintes associées aux tests et il est possible aussi de spécifier le fichier de base d'empreintes à utiliser.

4 Apport de Nmap-Stateful

Après avoir décrit les innovations présentes dans Nmap-Stateful, nous allons maintenant nous intéresser aux apports de l'outil. Le premier apport est l'augmentation de l'acuité de l'OS Fingerprinting. En effet, en augmentant les possibilités de tests, la pertinence et la puissance de détection sont augmentées. Nous avions montré avec l'outil Cron-OS que l'étude d'autres états de la pile TCP permettait dans certains cas de voir le véritable système d'exploitation se trouvant derriére un Syn-Relay.

Le deuxième apport est d'en faire un outil de test de pile TCP/IP. En effet grâce à Nmap-Stateful, on peut amener une pile TCP/IP dans un état voulu puis lui envoyer des paquets non-standard. On augmente ainsi la couverture de tests. Cela peut amener à renforcer la prise en compte de certains cas non-standard au sein de la pile et ainsi la rendre moins vulnérable.

Le troisième apport, qui nous semble le plus original, est le test de Firewall Stateful. Si l'on considère un Firewall Stateful (c'est-à-dire un Firewall qui maintient l'état des connexions dont les paquets transitent par ce Firewall), il peut être intéressant de tester et de connaître les mécanismes de transition d'états ou les filtres appliqués aux paquets. On peut bien sur jouer sur le flag TCP : que fait un Firewall lorsqu'il reçoit un paquet avec le flag SYN dans une connexion déjà existante ? Le considère-t-il comme une nouvelle connexion ou rejette-t'il le paquet ? En effet, dans un état ESTABLISHED, un paquet ne doit pas comporter le flag SYN. On peut aussi jouer sur les *sequence number* et *acknowledge number* pour découvrir la fenêtre dans laquelle le Firewall considère que le paquet est valide. C'est la raison pour laquelle ces champs sont modifiables dans la définition des tests de Nmap-Stateful. On consultera le livre de Richard Stevens [8] pour plus de détails sur TCP/IP. Un diagramme d'état TCP simplifié est présenté en figure 1 pour aider à la compréhension.

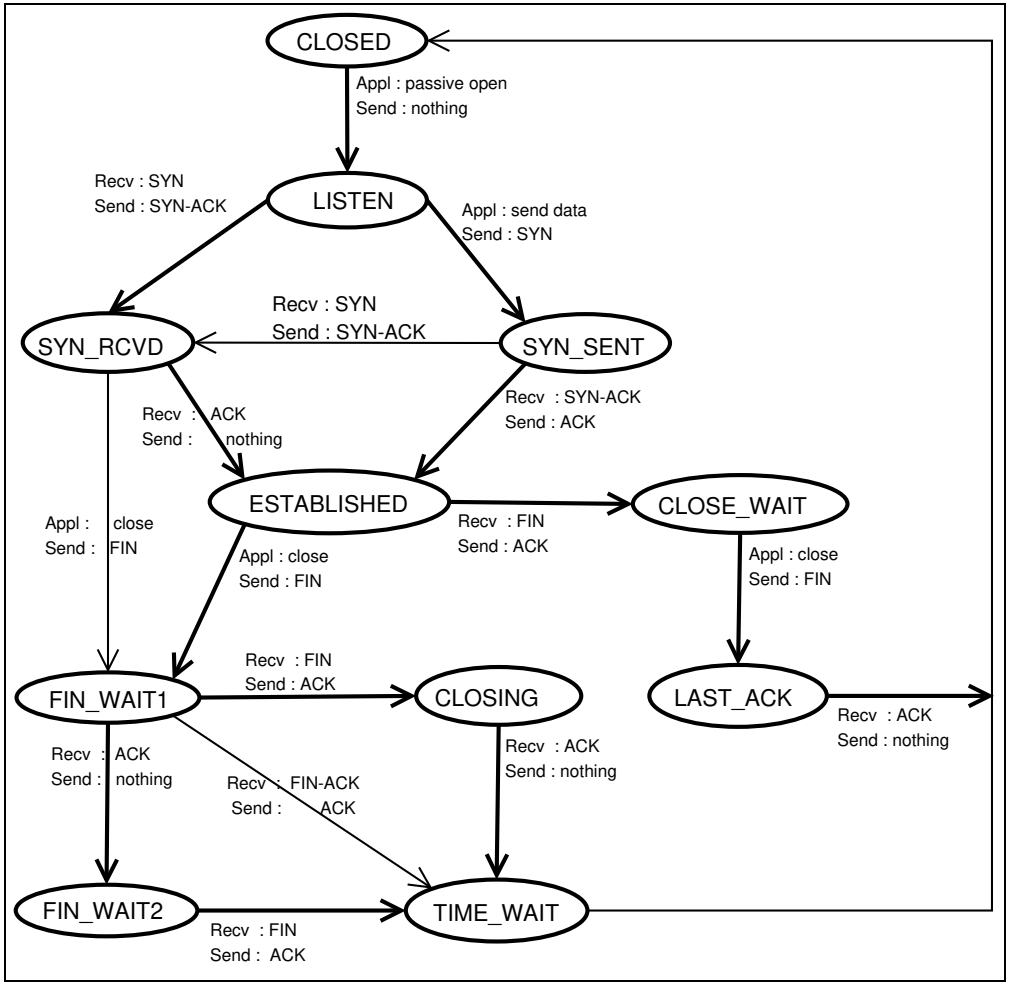


Fig. 1. Udiagramme d'état TCP, Appl : action de l'application possédant la socket, Recv : flags TCP reçus par la socket, Send : flags TCP émis par la socket.

5 Le programme Nmap-Stateful

Nmap-Stateful peut être téléchargé sur la page du projet [1]. Comme explicité plus haut, la principale caractéristique de Nmap-Stateful par rapport à Nmap est d'être capable d'amener la machine auditée dans l'état TCP désiré avant de lancer le test. Pour cela, deux éléments sont nécessaires : avoir une pile TCP/IP simplifiée permettant de lancer les paquets TCP correct par rapport aux standards (par exemple renvoyer un ACK au SYN-ACK de la machine distante pour terminer la connexion TCP), et il faut aussi bloquer la réception des paquets de la machine distante pour éviter que la pile TCP/IP de notre propre machine ne réponde. Pour cela, Nmap-Stateful positionne automatiquement des règles sur le Firewall de la machine locale pour bloquer la réception des paquets de la machine distante. Actuellement, Nmap-Stateful ne fonctionne que sous Linux 2.4 ou 2.6 (car il ne connaît que l'outil iptables) mais un portage vers d'autres Unix (FreeBSD, OpenBSD, Solaris) est possible grâce à la libdnet [5]. Nous allons définir quelques éléments nécessaires pour la bonne compréhension du fonctionnement de l'outil, ces éléments seront ensuite utilisés dans un exemple complet dans le prochain chapitre.

5.1 La définition des tests

Dans cette section, nous présentons briévement la grammaire de Nmap-Stateful permettant d'exprimer les tests.

Un test est défini entre les deux mots clé NAME= et END. Avant de lancer le test, on peut définir l'état dans lequel on désire que soit la connexion TCP. Tous les états ne sont pas accessibles car certains sont des états transitoires. À terme les états suivants devraient être disponibles : LISTEN, CLOSE, SYN_RCV, ESTABLISED, FIN_WAIT1, CLOSING, FIN_WAIT2, TIME_WAIT, LAST_ACK. On peut aussi définir les flags TCP à positionner dans le paquet. Les options TCP peuvent être précisées ainsi que les données contenues dans le paquet. Les *sequence number* et *acknowledge number* sont modifiables dans le paquet de test. En effet une connexion TCP possède un *sequence* et *acknowledge number* courant, par défaut le paquet de test les utilise mais on peut altérer ces nombres en ajoutant une valeur précisée dans la description du test.

Exemple de test :

```
NAME=T1
LISTEN
TH_SYN
TH_ECE
OPTIONS=\003\003\012\001\002\004\001\011\010\012\077\077\077\077
OPTIONSLEN=14
SEQ=2
ACK=-3
DATA=example
DATALEN=6
```

```
END
```

Ce test s'appelle T1, il envoit son paquet de test lorsque la cible est dans l'état LISTEN. Le paquet de test contient les flags SYN et ECN et des options TCP. Le paquet transport des données (*example*) et le *sequence number* courant de la connexion sera incrementé de 2 et le *acknowledge number* sera decrementé de 3.

Pour assurer une certaine compatibilité avec Nmap, un fichier de tests correspondant aux tests de Nmap est incorporé dans Nmap-Stateful.

5.2 Le fonctionnement de Nmap-Stateful

Plusieurs options ont été rajoutées à Nmap :
--os_tests_file *file* : spécifie le fichier de tests à utiliser (ou --otf).
--os_result_in_file *file* : spécifie un fichier où écrire les résultats (ou --orf).
--os_fingerprint_file *file* : spécifie le fichier d'empreintes de référence à utiliser (ou --otf).

Il existe d'autres options pour préciser le nombre maximum de tests à effectuer en même temps (pour éviter de stresser la machine) et le nombre de secondes à attendre la réception d'un paquet réponse.

Pour illustrer le fonctionnement de Nmap-Stateful, nous allons commencer par un exemple simple qui consiste à lancer un test lorsque la machine auditée est dans l'état ESTABLISHED. Le résultat du test est seulement affiché car on ne fournit pas de fichier de base d'empreintes.

La machine testée est un linux 2.6 sur un réseau local qui écoute sur le port 22 (SSH). le fichier de tests *test_example* est le suivant :

```
NAME=ESTABLISHED_SUPE
ESTABLISHED
TH_SYN
TH_URG
TH_ECE
TH_PUSH
DATA=example
DATALEN=7
ACK=1
END
```

La trace d'exécution est :

```
#./nmap-stateful --otf test -p 22 192.168.1.1
.....
Send SYN 38557 => 22 for test:ESTABLISHED_SUPE
.....
SYN_SENT seq:1281454224 sp:38557
                --> dp:22 ack:0000000000 SYN_RECV fl:S
```

```
ESTABLI ack:1281454225 dp:38557
                <-- sp:22 seq:1426449042 SYN_RECV fl:SA
Launch test:ESTABLISHED_S with source port:38557
UNKONWN seq:1281454225 sp:38557
                --> dp:22 ack:1426449043 UNKONWN fl:A
UNKONWN seq:1281454225 sp:38557
                --> dp:22 ack:1426449043 UNKONWN fl:SPUB
LISTEN ack:1281454233 dp:38557
                <-- sp:22 seq:0000000000 UNKONWN fl:RA
.....
ESTABLISHED_SUPE(Resp=Y%DF=Y%W=0%ACK=0%Flags=AR%Ops=)
```

La première ligne correspond à la ligne de commande, on utilise donc le fichier de test *test*, on précise le port (-p 22) et la machine auditée (192.168.1.1). Puisque le test exige que la connexion soit dans l'état ESTABLISHED avant de lancer le paquet de test, le programme envoi automatiquement un paquet de début de connexion (port source 38557 vers le port destination 22). Les lignes suivantes sont des lignes d'informations sur les échanges de paquets, le format est :

- l'état de notre machine (SYN_SENT)
- le *sequence number* du paquet (seq :1281454224)
- le port source (sp :38557)
- la direction du paquet (–>)
- le port destination (dp :00022)
- l'*acknowledge number* du paquet (ack :0000000000)
- les flags TCP présent dans ce paquet (fl :SA)

Lorsque le paquet de test est lancé, les états des machines dans cette connexion sont positionnés à UNKNOWN car les conséquences de la réception du paquet de test sur la connexion ne sont pas connues. Le programme note la réaction de la machine, la fait passer par le l'analyseur de paquet TCP de Nmap, puis affiche l'empreinte de la réaction de la cible.

5.3 Création de tests pertinents

Nous avons plus haut que le nombre de tests potentiels est très important (la combinaison des flags TCP et des états est important), il faut trouver une méthode pour sélectionner les tests pertinents.

Dans un premier temps il faut se poser la question de la définition d'un test pertinent. Un test pertinent est un test stable, *i.e.*, qui renvoie toujours le même résultat sur le même type de cible et dans le même environnement. Par exemple le *Window size* sera toujours 0xE240 sur FreeBSD 4.9 configuré par défaut. Un test pertinent est aussi un test discriminant ,*i.e.*, qui permet de montrer une différence de comportement entre deux types de cibles ou d'environnements.

Pour cela, un outil est livré avec Nmap-Stateful pour aider les utilisateurs à créer leurs propres tests en générant des ensembles de tests automatiquement

et en les triant suivant leur résultats. Cet outil s'appelle *fingerprinttool* et voici quelques exemples d'utilisations :

La génération automatique d'un ensemble de tests, en utilisant un test de base, par exemple :

```
NAME=example
ESTABLISHED
TH_URG
TH_FIN
END
```

On le met dans un fichier (appelé *test_sample* ici) et on appelle l'outil fingerprinttool :

```
#./fingerprinttool -g test\_sample -o generate\_tests
```

Cet outil prend en argument la commande -g (pour *generate*) suivi du fichier de test (*test_sample*) et écrit le résultat dans le fichier *generate_tests*. Le fichier *generate_tests* contient l'ensemble des tests générés :

```
NAME=ESTABLISHED_______
ESTABLISHED
END
NAME=ESTABLISHED___F______
ESTABLISHED
TH_FIN
END
NAME=ESTABLISHED_____U____
ESTABLISHED
TH_URG
END
NAME=ESTABLISHED___F_U____
ESTABLISHED
TH_FIN
TH_URG
END
```

Si l'état TCP n'est pas précisé dans le fichier de test source, alors un test par état sera aussi généré. On précise dans le fichier source les flags TCP que l'on compte utiliser, l'outil générera toutes les combinaisons possibles (sauf l'option NO_GEN_FLAGS est ajoutée). On peut de même ajouter aussi des variations sur les *sequence number* et les *acknowledge number* en mettant par exemple les deux lignes SEQ=3 et GEN_SEQ. Dans ce cas, le programme fingerprinttool va générer tout les tests suivant SEQ=-3, SEQ=-2,..., SEQ=2 , SEQ=3.

Après avoir généré un grand nombre de tests, il faut trouver des tests stables (donnant toujours le même résultat). Pour cela on lance la suite de tests sur plusieurs cibles dont on attend le même comportement puis on filtre les tests qui donnent le même résultat :

```
\#./fingerprinttool -s -t tests -o stable\_tests resultat1 /
resultat2 ...  resultatN
```

On utilise la commande -s (pour same) avec comme arguments -t pour lui indiquer le fichier de tests utilisés, -o pour préciser le fichier où les tests jugés stables seront stockés, puis ensuite la liste des fichiers de résultats obtenus avec ces tests sur les différentes cibles.

Maintenant que dans l'ensemble des tests générés on a sélectionné les tests stables, il faut y rechercher les tests différentiateurs. Pour cela il faut lancer les tests stables sur des cibles différentes que l'on veut pouvoir différencier grâce à Nmap-Stateful, puis filtrer ceux qui amènent des résultats différents :

```
#./fingerprinttool -d -t tests -o discriminant\_tests /
fingerprintresult1 resultat1 resultat2 ...  resultatN
```

On utilise la commande -d (pour *different*) pour filtrer les tests qui donnent des résultats différents.

Au lieu de -s ou -d, on peut utiliser la commande -c qui trie les tests du plus différentiateur au moins différentiateur car ils n'existe pas toujours des tests qui ont un résultat différent sur toutes les cibles testées.

Au final, on peut ainsi générer des tests pertinents de façon quasi automatique sans avoir besoin d'être un expert TCP/IP.

6 Un exemple réel complet

Pour bien montrer le fonctionnement de l'outil, ainsi que pour montrer quelques résultats, nous proposons de faire un exemple complet d'utilisation de Nmap-Stateful. On ne va pas chercher ici à obtenir les tests les plus pertinents mais plutôt à montrer qu'il est possible d'utiliser l'outil sans aucune analyse détaillée TCP/IP. Le but sera ici de construire un fichier d'empreinte (fingerprint) permettant de reconnaître les principaux OS.

6.1 Génération des tests

Nous choisissions de nous intéresser à l'état ESTABLISHED, en utilisant les flags TCP : SYN, ACK, PUSH, FIN et en transmettant des données dans le paquet TCP. Le test de base permettant de générer l'ensemble des tests est le suivant :

```
NAME=test_sample
ESTABLISHED
TH_SYN
TH_ACK
TH_FIN
TH_PUSH
DATA=example
```

```
DATALEN=7
SEQ=1
END
```

On peut noter la ligne *SEQ=1* car le paquet de test utilise par défaut le *sequence number* et *acknowledge number* du dernier paquet reçut. Dans le cas de notre test, le SYN-ACK de la connexion TCP, puis nous avons envoyé le ACK de fin de connexion, le *sequence number* doit donc être incrémenté de 1.

La génération des tests s'effectue donc ainsi :

```
#./fingerprinttool -g test\_sample -o generate\_tests
```

Dans le fichier *generate_ tests*, on a par exemple les tests suivant :

```
NAME=ESTABLI_
ESTABLISHED
DATA=example
DATALEN=7
SEQ=1
END
NAME=ESTABLI_SAFP
ESTABLISHED
TH_SYN
TH_ACK
TH_FIN
TH_PUSH
DATA=example
DATALEN=7
SEQ=1
END
```

6.2 Sélections des tests stables

Pour sélectionner les tests stables il faut vérifier que les tests générés donnent toujours le même résultat sur le même type d'OS. Pour cela, on va lancer les tests sur plusieurs cibles ayant le même OS (on peut aussi lancer le test plusieurs fois sur la même cible pour augmenter la fiabilité). Il est important de choisir des cibles qui ne sont pas protégées par des Firewalls comme nous le verrons par la suite (cela n'a malheureusement pas toujours été possible dans les tests réalisés ci-dessous).

Exemple sur l'OS Linux 2.4 :

Première cible :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--orf generate_tests_Linux2.4-1  premiere_cible
...
ESTABLI_(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
```

```
ESTABLI_S(Resp=Y%DF=Y%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_A(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
ESTABLI_SA(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_F(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
ESTABLI_SF(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
ESTABLI_AF(Resp=N)
ESTABLI_SAF(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_P(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
ESTABLI_SP(Resp=Y%DF=Y%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_AP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=Y%DF=Y%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_AFP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
ESTABLI_SAFP(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
```

Les options -n, -P0, -sS sont des options appartenants à l'outil Nmap. Les options de Nmap-Stateful utilisées sont :
--ntp 5 : on lance 5 tests en parallèle.
--nts 3 : la réponse de la cible est attendue trois secondes avant de conclure à l'absence de réponse.
--orf *generate_tests_Linux2.4-1* : le résultat est mis dans le fichier indiqué

Sans rentrer dans les détails, on peut s'apercevoir de plusieurs éléments :
- Certains tests amenaient des réponses et d'autres non.
- Plusieurs groupes de flags TCP sont retournés (A, AR, AS et R).
- Plusieurs groupes d'options sont retournés (NNTNN, MMNVV ou aucune option).

On recommence le test une autre fois sur la même cible et on stocke le résultat dans le fichier *generate_tests_Linux2.4-2*. Puis sur une autre cible ayant le même OS, les tests sont lancés deux fois, ce qui génère deux nouveaux fichiers résultat *generate_tests_Linux2.4-3* et *generate_tests_Linux2.4-4*.

Nous pouvons vérifier si les tests générés sont stables grâce à l'outil fingerprinttool :

```
#./fingerprinttool -s -t generate_tests -o selected_tests /
generate_tests_Linux2.4-1 generate_tests_Linux2.4-2 /
generate_tests_Linux2.4-3 generate_tests_Linux2.4-4 /
```

- L'option -s : ne sélectionne que les tests ayant le même résultat dans tous les fichiers de résultats donnés en paramètre.
- L'option -t : le fichier de test utilisé pour générer les fichiers de résultats.
- L'option -o : le nom du fichier où seront placés les tests sélectionnés.

Le résultat ici est que tous les tests sont jugés stables. Il arrive parfois que les tests soient jugés non stables à cause d'erreurs réseau et on conclut faussement à la non stabilité des tests, il faut alors éliminer ces résultats et recommencer de nouveaux tests. Une des possibilités pour que deux tests différent est que le

Window Size ne soient pas le même. En effet le *Window Size* correspond à la place libre du buffer de réception, taille qui peut donc varier lorsque la connexion TCP est établie.
Une nouvelle fonctionnalité de l'outil fingerprinttool qui devrait être incluse dans une prochaine version, pour que par exemple si deux tests ne différent que par le *Window Size*, alors le test soit quand même retenu mais que le *Window Size* du résultat du test ne soit pas comparé au *Window Size* de l'empreinte contenue dans le fichier d'empreintes.

6.3 Sélections des tests discriminants

On suppose maintenant que l'on dispose des fichiers de résultat pour chaque type d'OS que l'on souhaite reconnaître avec le fichier de tests stables construit à l'étape précédente. On va maintenant rechercher les tests les plus discriminants. Pour cela, on va une nouvelle fois faire appel à l'outil fingerprinttool.

```
#./fingerprinttool -d -t generate_tests -o selected_tests /
generate_tests_FreeBSD4.9 generate_tests_FreeBSD5.2 /
generate_tests_Linux2.4 generate_tests_Linux2.6 /
generate_tests_MacOSX generate_tests_Solaris /
generate_tests_Windows2000 generate_tests_Windows2003
```

L'option -d : ne sélectionne que les tests ayant un résultat différent dans tous les fichiers de résultats donnés en paramètre. Les autres options sont identiques à celles vues précédemment.

Malheureusement ici le fichier *selected_ tests* généré par l'outil fingerprinttool est vide, aucun test ne permet à lui seul de différencier tous les OS voulus. On va devoir se contenter de sélectionner les tests les plus différentiateurs seulement. L'outil fingerprinttool est capable de trier des moins discriminants au plus discriminants avec l'option -c.

```
#./fingerprinttool -c -t generate_tests -o selected_tests /
generate_tests_FreeBSD4.9 generate_tests_FreeBSD5.2 /
generate_tests_Linux2.4 generate_tests_Linux2.6 /
generate_tests_MacOSX generate_tests_Solaris /
generate_tests_Windows2000 generate_tests_Windows2003

21 occurences of:ESTABLI_P
16 occurences of:ESTABLI_F
16 occurences of:ESTABLI_FP
15 occurences of:ESTABLI_
7 occurences of:ESTABLI_SAFP
5 occurences of:ESTABLI_SFP
4 occurences of:ESTABLI_SA
4 occurences of:ESTABLI_SAF
4 occurences of:ESTABLI_SP
```

```
3 occurences of:ESTABLI_AF
2 occurences of:ESTABLI_S
2 occurences of:ESTABLI_A
2 occurences of:ESTABLI_SF
2 occurences of:ESTABLI_AP
2 occurences of:ESTABLI_AFP
1 occurences of:ESTABLI_SAP
```

Le test qui obtient le plus souvent le même résultat est le test nommé ESTABLI_P (dans l'état ESTABLISHED des données sont envoyés avec le flag PUSH). Les résultats sont les suivants :

```
FreeBSD4.9:ESTABLI_P(Resp=N)
FreeBSD5.2:ESTABLI_P(Resp=N)
Linux2.4:ESTABLI_P(Resp=N)
Linux2.6:ESTABLI_P(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
MacOSX:ESTABLI_P(Resp=N)
Solaris:ESTABLI_P(Resp=N)
Windows2000:ESTABLI_P(Resp=N)
Windows2003:ESTABLI_P(Resp=N)
```

Seul le Linux 2.6 a répondu à ce test. Ce test est donc peut-être une caractéristique. Inversement le test ESTABLI_SAP (dans l'état ESTABLISHED sont envoyés avec avec les flags SYN, ACK et PUSH) ne renvoit presque jamais les mêmes résultats :

```
FreeBSD4.9:ESTABLI_SAP(Resp=Y%DF=Y%W=E000%ACK=0%Flags=AS%Ops=MNWNNT)
FreeBSD5.2:ESTABLI_SAP(Resp=Y%DF=Y%W=FFFF%ACK=0%Flags=AS%Ops=MNWNNT)
Linux2.4:ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
Linux2.6:ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
MacOSX:ESTABLI_SAP(Resp=Y%DF=Y%W=8218%ACK=0%Flags=AF%Ops=NNT)
Solaris:ESTABLI_SAP(Resp=Y%DF=Y%W=6028%ACK=S++%Flags=AR%Ops=)
Windows2000:ESTABLI_SAP(Resp=Y%DF=Y%W=FFFF%ACK=0%Flags=AS% /
   Ops=MNWNNTNNM)
Windows2003:ESTABLI_SAP(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

Pour le reste de l'exemple, on continuera avec le même fichier de test car tous les tests apportent une part d'information pour différentier les OS, même s'il y a une possibilité d'avoir des tests redondants.

6.4 Génération du fichier d'empreintes

Après avoir construit des fichiers de résultats pour chaque type d'OS que l'on souhaite reconnaître, il faut tous les rassembler dans un seul fichier pour en faire notre base de référence par rapport aux tests sélectionnés durant l'opération précédente.

Pour cela, il faut commencer à éditer le fichier de résultat et à remplir correctement la ligne commençant par Fingerprint. Cette ligne est ajouté automatiquement lors de la génération du fichier, elle comporte des informations sur la cible testée : son adresse IP, le port ouvert utilisé, le port fermé utilisé (le port 0 est indiqué si ce port n'existe pas) et la date (format machine) à laquelle le test a été effectué. Exemple pour le fichier correspondant a l'OS FreeBSD 4.9 :

```
Fingerprint 131.254.200.7 53 0 1081951693
ESTABLI_(Resp=N)
ESTABLI_S(Resp=Y%DF=N%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_A(Resp=Y%DF=Y%W=E240%ACK=0%Flags=A%Ops=NNT)
ESTABLI_SA(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_F(Resp=N)
ESTABLI_SF(Resp=Y%DF=Y%W=E000%ACK=0%Flags=AS%Ops=MNWNNT)
ESTABLI_AF(Resp=Y%DF=Y%W=E240%ACK=0%Flags=A%Ops=NNT)
ESTABLI_SAF(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_P(Resp=N)
ESTABLI_SP(Resp=Y%DF=N%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_AP(Resp=Y%DF=Y%W=E240%ACK=0%Flags=A%Ops=NNT)
ESTABLI_SAP(Resp=Y%DF=Y%W=E000%ACK=0%Flags=AS%Ops=MNWNNT)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=Y%DF=N%W=0%ACK=0%Flags=AR%Ops=)
ESTABLI_AFP(Resp=Y%DF=Y%W=E240%ACK=0%Flags=A%Ops=NNT)
ESTABLI_SAFP(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

Ici on change donc la première ligne en : Fingerprint FreeBSD 4.9. Puis on rassemble tous les fichiers de résultats en un seul fichier :

```
cat generate_tests_Linux2.4 generate_tests_Linux2.6
generate_tests_FreeBSD4.9 ... generate_tests_Windows2000
 >>  generate_tests_fingerprint
```

6.5 Test de Nmap-Stateful

On a fini la construction des tests et du fichier d'empreintes, nous allons maintenant pouvoir tester la pertinence des tests. La méthodologie consiste à lancer notre suite de tests sur des cibles différentes des machines ayant permis de construire l'empreinte et de vérifier que les OS sont bien découverts.

Sur un autre Linux 2.4 que les deux ayant servis de référence, on trouve :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--off generate_tests_fingerprint -p 80 x.x.x.x
Interesting ports on x.x.x.x:
PORT   STATE SERVICE
80/tcp open  http
OS details: Linux 2.4
```

Toujours sur un Linux 2.4, mais différent de ceux testés précedement, nous obtenons le résultat suivant :

```
80/tcp open  http
Aggressive OS guesses: Linux 2.6 (97%), Linux 2.4 (96%)
No exact OS matches for host (test conditions non-ideal).
```

Nmap-Stateful ne sait pas statuer exactement sur l'OS de la machine testée mais donne les possibilités les plus probables : Linux 2.6 et Linux 2.4. En réalité, l'OS est un Linux 2.4 mais avec une sous version plus récente que les versions ayant permis de faire l'empreinte présente dans le fichier de référence. Il faudrait donc affiner la base pour qu'elle prenne en compte les sous versions du noyau Linux.

Dans la plupart des cas, l'outil va être capable de reconnaître correctement l'OS de la machine cible, directement (comme dans le premier cas) ou indirectement (la plus forte probabilité).

L'outil peut parfois être mis en échec. Prenons le cas où la cible est un Solaris 8 :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--off generate_tests_fingerprint -p 80   x.x.x.x

ESTABLI_(Resp=N)
ESTABLI_S(Resp=N)
ESTABLI_A(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=AF%Ops=)
ESTABLI_SA(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_F(Resp=N)
ESTABLI_SF(Resp=N)
ESTABLI_AF(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAF(Resp=N)
ESTABLI_P(Resp=N)
ESTABLI_SP(Resp=N)
ESTABLI_AP(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAP(Resp=Y%DF=N%W=200%ACK=0%Flags=R%Ops=)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=N)
ESTABLI_AFP(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAFP(Resp=N)
```

Cet échec est dû à la présence d'un Firewall qui perturbe les tests. L'empreinte au final dépend de l'OS et du Firewall et ne correspond plus à celle connue par Nmap-Stateful. Nous étudions cette problématique dans le paragraphe suivant.

7 Les Firewalls

L'action des Firewalls sur la prise d'empreinte est connue et Nmap souffre de ce problème aussi. Il y a plusieurs façons d'aborder ce problème, c'est à la

fois une difficulté qu'il va falloir intégrer à la problématique mais c'est aussi un apport car nous allons aussi pouvoir étudier les comportements des Firewalls et faire ainsi non plus de l'OS Fingerprinting mais aussi du Firewall Fingerprinting. Cette étude débute, mais il existe des techniques qui devraient permettre de réaliser cet objectif. La méthode serait de trouver un grand nombre de cible ayant le même OS et de réaliser une empreinte de ce système pour un grand nombre de tests. Parmi ces cibles, il y aurait des cibles protégées par des Firewalls et d'autres non protégées. La technique serait de sélectionner ensuite les tests qui donnent toujours le même résultat indépendamment de la présence ou non d'un Firewall, ces tests nous permettraient de faire de l'OS Fingerprinting dans toutes les architectures réseau (contrairement à Nmap qui est sensible aux éléments de coupure (Firewall ou Syn-Relay)). Dans un deuxième temps on devrait sélectionner les tests qui donnent des résultats différents suivant le Firewall présent. Ces tests vont donc nous permettre de faire du Firewall Fingerprinting. Cette étude fait partie des travaux restant à réaliser dans ce domaine.

Néanmoins, notre premiére étude montre des résultats encourageants. Lors de la présentation de l'outil fingerprinttool, nous avions fait remarquer que l'on pouvait générer des tests avec des *sequence number* modifiés. Lors de l'essai de ce type de tests sur des cibles nous pouvons observer le phénomène suivant :

Sur un Solaris 8 protégé par un Firewall Stateful :

```
ESTABLI_AP_SEQ-3(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ-2(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ-1(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ0(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ1(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ2(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ3(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
```

Toujours sur un Solaris 8 protégé par un autre type de Firewall Stateful :

```
ESTABLI_AP_SEQ-3(Resp=N)
ESTABLI_AP_SEQ-2(Resp=N)
ESTABLI_AP_SEQ-1(Resp=N)
ESTABLI_AP_SEQ0(Resp=T%DF=Y%W=8325%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ1(Resp=N)
ESTABLI_AP_SEQ2(Resp=N)
ESTABLI_AP_SEQ3(Resp=N)
```

Le test ayant permit de généré tous les tests est :

```
NAME=seq
ESTABLISHED
TH_PUSH
TH_ACK
SEQ=3
GEN_SEQ
```

```
NO_GEN_FLAGS
DATA=example
DATALEN=7
END
```

Le comportement est très diffèrent d'une machine à l'autre, or ce sont les mêmes systèmes d'exploitation, c'est donc bien essentiellement l'empreinte du comportement du Firewall que nous avons. Une remarque est que bien que ce soit deux Firewalls Stateful, un seul d'entre eux vérifie les *sequences number* apparemment et ne laisse passer que les paquets respectant scrupuleusement les standards. Cela montre que l'étude des Firewalls Stateful est possible grâce à Nmap-Stateful et que cette étude peut apporter des éléments de compréhension intéressante dans ce domaine.

8 Conclusion et perspectives

Dans ce papier nous avons montré que les premiers résultats de l'outil sont encourageants et laissent entrevoir que l'outil va tenir ses promesses. L'OS Fingerprinting devrait être le premier domaine où l'outil devrait s'imposer, en effet Nmap-Stateful reprend l'ensemble des méthodes de Nmap et les tests utilisés par pOf peuvent aussi être repris. Une des difficultés qui reste à surmonter est de disposer de suffisamment de plateforme de tests pour pouvoir mettre au point des tests pertinents car le nombre de tests possibles est très grand ainsi que le nombre de combinaisons de cibles possibles (OS et Firewall). La problématique de l'OS Fingerprinting ne relève plus du domaine de TCP/IP mais de la gestion d'un grand nombre d'information.

Références

1. O. Courtay, site de Nmap-Stateful, `http://home.gna.org/nmapstateful/`
2. O. Courtay, O. Heen et F. Veysset, Détection des systèmes d'exploitation avec Cron-OS, *SSTIC 2003*.
3. O. Arkin, F. Yarochkin et M. Kydyraliev, The Present and Future of Xprobe2 - The Next Generation of Active Operating System Fingerprinting, `http://www.sys-security.com/archive/papers/Present\_and\_Future\_Xprobe2-v1.0.pdf`
4. M. Zalewski, L'outil d'OS Fingerprinting passif pOf, `http://lcamtuf.coredump.cx/p0f.shtml`
5. D. Song, Libdnet, `http://libdnet.sourceforge.net/`
6. Fyodor, Remote OS detection via TCP/IP Stack Fingerprinting, Phrack 1998, `http://www.insecure.org/nmap/nmap-fingerprinting-article.txt`
7. Fyodor, Le site de Nmap, `http://www.insecure.org`.
8. R. Stevens, *TCP/IP Illustrated*, Addison-Wesley, 1993.

Gestion sécurisée de groupes de dispositifs dans les réseaux domestiques

Nicolas Prigent*† and Jean-Pierre Andreaux*

*Thomson R&D France, Rennes
{nicolas.prigent|jean-pierre.andreaux@thomson.net}
†Supelec, Rennes

Résumé Un réseau domestique est formé d'un ensemble de dispositifs (téléviseurs, enregistreurs numériques, ordinateurs, assistants numériques personnels, etc.) mis en réseau qui s'auto-configurent et interagissent de manière transparente pour l'utilisateur afin de lui offrir des services augmentés. Une première étape dans la sécurisation de ces réseaux consiste en la définition de leur frontière [3] . Il s'agit, en d'autres termes, de connaître de manière sécurisée les dispositifs qui y appartiennent. Dans cet article, nous proposons un mécanisme totalement distribué de gestion sécurisée de groupes dynamiques dédié aux réseaux domestiques. Ce mécanisme prend en compte les évolutions possibles d'un réseau domestique (ajout ou retrait de dispositifs par exemple), ainsi que le besoin de facilité d'utilisation inhérent à ce type de réseaux.

1 Introduction

Un réseau domestique [3] est formé d'un ensemble de dispositifs (téléviseurs, enregistreurs numériques, ordinateurs, assistants numériques personnels, etc.) mis en réseau qui s'auto-configurent et interagissent de manière transparente pour l'utilisateur afin de lui offrir des services augmentés. *UPnP* [9], *HAVi* [1] et *Rendezvous* [8] sont quelques propositions actuelles de standards de réseaux domestiques.

Les réseaux domestiques doivent être sécurisés si on souhaite leur large déploiement. En effet, il existe des mobiles et de véritables opportunités pour attaquer les réseaux domestiques. Comme nous l'avons montré dans [3], la première étape pour sécuriser un réseau domestique consiste à marquer sa frontière, c'est à dire à définir quels sont les dispositifs qui appartiennent à ce réseau.

En effet, la frontière des réseaux domestiques n'est pas clairement définie. L'utilisation de média hertziens, par nature partagés, la communication entre dispositifs au travers de l'Internet, la découverte et l'échange automatique de services entre dispositifs mis en présence [8,1,9] sont autant de facteurs qui rendent floue la frontière des réseaux domestiques : un dispositif de réseau domestique échangera *a priori* des services avec n'importe quel autre dispositif compatible et présent. Or, le fait qu'il ait la possibilité de communiquer et d'échanger des services avec un autre dispositif n'implique pas qu'il en ait le droit. Hors la mise

en place d'un mécanisme de sécurité spécifique, un dispositif ne peut décider de manière sécurisée si un autre dispositif appartient au même réseau domestique et a par conséquent le droit d'échanger des services avec lui. Ce mécanisme doit prendre en compte les évolutions possibles dans les réseaux domestiques (ajout et retrait de dispositifs), leur dynamicité et leur besoin de facilité d'utilisation.

En effet, contrairement aux réseaux d'entreprises, les réseaux domestiques ne peuvent bénéficier d'administrateurs compétents et disponibles. Leurs utilisateurs n'ont, pour la plupart, ni le temps, ni les connaissances pour gérer la sécurité de leur système. Pire, ils sont souvent considérés comme le maillon faible de la sécurité [6]. Mais s'ils doivent être écartés de la manipulation d'outils de sécurité complexes, les utilisateurs des réseaux domestiques ne peuvent cependant pas se soustraire à l'expression de la politique de sécurité. Dans le cadre de la définition de la frontière, ils sont les seuls à pouvoir décider quels dispositifs appartiennent au réseau domestique, et par conséquent les plus à même de l'exprimer. De ce fait, l'expression de la frontière doit se faire de la manière la plus simple et la plus discrète possible.

Dans cet article, nous présentons un mécanisme sécurisé, facile d'utilisation et totalement décentralisé de gestion de groupes dynamiques qui permet de résoudre le problème de la frontière dans les réseaux domestiques : chaque dispositif gère localement sa propre connaissance des dispositifs qui appartiennent au même réseau domestique que lui. Il maintient cette connaissance la plus à jour possible de manière sécurisée à partir des informations fournies par l'utilisateur et par les autres dispositifs de son réseau domestique. Dans la section 2, nous présentons les différentes opérations d'évolution d'un réseau domestique, qui doivent être prises en compte par un système de gestion de frontière. Dans la section 3, nous traitons des propriétés de topologie dynamique et de connectivité erratique des réseaux domestiques, qui ont imposé le choix d'un mécanisme totalement distribué. Notre proposition est décrite en section 4.

2 Évolutions de la frontière des réseaux domestiques

Les dispositifs d'un même réseau domestique, et donc à l'intérieur de la frontière, sont autorisés par la politique à communiquer et à échanger des services. Ils sont liés par une relation de confiance : un dispositif peut légitimement considérer que les dispositifs que son autorité (c'est à dire l'utilisateur) a déclarés appartenir au réseau domestique se comporteront en accord avec la politique. De plus, il peut avoir confiance en les informations fournies par les autres dispositifs de son réseau domestique.

Notons que tous les dispositifs d'un même réseau domestique n'appartiennent pas forcément à la même personne : chaque membre de la famille dispose de ses propres dispositifs sur lesquels il fait autorité. Néanmoins, tous les utilisateurs partagent le même intérêt dans le fait d'interconnecter leurs dispositifs de manière sécurisée. Un réseau domestique dispose donc de plusieurs autorités, mais

elles partagent toutes la même politique concernant l'appartenance d'un dispositif au réseau domestique. En d'autres termes, lorsqu'un des utilisateurs insère un dispositif dans le réseau domestique, tous les utilisateurs sont d'accord sur cette insertion.

La relation de confiance entre dispositifs d'un même réseau domestique est une relation à long terme : un dispositif entre *a priori* dans un réseau domestique pour une longue durée, et le quitte *a priori* définitivement (lorsqu'il tombe en panne, qu'il est vendu, perdu ou volé). D'un point de vue fonctionnel, il existe pour un réseau domestique quatre opérations d'évolution différentes :

- l'initialisation du réseau,
- l'insertion d'un dispositif,
- le retrait d'un dispositif,
- le bannissement d'un dispositif.

Chaque réseau domestique dispose d'un état initial, lorsque le premier dispositif est installé. Lors de l'initialisation, un dispositif est seul dans son réseau domestique. Le réseau évolue par la suite au gré des insertions et des retraits des dispositifs. La sécurité d'un réseau domestique doit être assurée dès son état initial, et maintenue au fil de ses évolutions.

Lorsqu'un dispositif est inséré dans un réseau domestique, il est capable d'identifier les autres dispositifs qui y appartiennent comme étant du même réseau domestique, et eux même sont capables de l'identifier comme faisant partie du leur.

Lorsqu'un dispositif est retiré du réseau domestique, par exemple parce qu'il est vendu ou donné, l'utilisateur peut y accéder physiquement pour y faire des modifications si nécessaire. Après qu'un dispositif ait été retiré du réseau, il ne doit plus reconnaître les autres dispositifs comme faisant partie de son propre réseau, et eux même ne doivent pas le reconnaître comme étant dans le leur.

Enfin, lorsqu'un dispositif est perdu ou volé, il existe un risque qu'un individu malveillant s'en serve pour accéder aux services offerts par les autres dispositifs du réseau domestique. Pour éviter cela, l'utilisateur doit pouvoir bannir du réseau domestique n'importe quel dispositif corrompu, c'est à dire informer de manière sécurisée les autres dispositifs du réseau domestique que le dispositif corrompu n'appartient plus à leur réseau domestique. L'utilisateur n'ayant *a priori* plus aucun contrôle sur le dispositif banni, aucune hypothèse ne peut être faite à son sujet. La seule propriété qui puisse être garantie est que les autres dispositifs du réseau domestique ne le considèrent plus comme y appartenant.

Remarquons enfin que pour des raisons de survie [5] du réseau domestique, n'importe quel dispositif doit pouvoir en être retiré ou banni sans que cela nuise à la pérennité du réseau. En d'autres termes, aucun dispositif ne doit lui être indispensable. Cette propriété est particulièrement importante dans les réseaux domestiques, qui sont constitués de dispositifs auxquels les utilisateurs ne prêtent pas forcément toute l'attention nécessaire.

3 Topologie dynamique et connectivité erratique

À l'instar des réseaux *ad hoc* [?], les réseaux domestiques sont topologiquement très dynamiques : la manière dont les dispositifs sont interconnectés varie au cours du temps, et certains dispositifs peuvent même être temporairement déconnectés. Par exemple, lorsqu'un utilisateur emporte avec lui ses dispositifs mobiles (ordinateurs portables, téléphones mobiles, assistants numériques personnels, lecteurs audios ou vidéos de poche, etc.), ceux-ci ne sont plus à même de communiquer avec les dispositifs qu'il laisse à son domicile (téléviseurs, ordinateurs de bureau, chaînes Hi-Fi, etc.). Par conséquent, il n'y a aucune certitude que deux dispositifs d'un même réseau domestique puissent communiquer à un instant donné. Aucun dispositif ne peut être considéré toujours présent, et aucun d'entre eux ne peut jouer le rôle de point de contrôle centralisé.

Nonobstant, un réseau domestique doit pouvoir fonctionner et évoluer de manière sécurisée même si un seul de ses dispositifs est présent, et ce quel que soit ce dispositif. Par exemple, lorsqu'un utilisateur achète un PDA et n'a sur lui que son téléphone mobile (qui, pour sa part, appartient déjà au réseau domestique), il doit pouvoir immédiatement interconnecter de manière sécurisée ces deux dispositifs pour qu'ils puissent s'échanger des services. En d'autres termes, l'utilisateur doit pouvoir insérer le PDA dans le réseau domestique en utilisant le téléphone.

De même, lorsqu'un réseau domestique se retrouve physiquement partitionné à un instant donné, aucune des partitions n'est capable de communiquer avec les autres, et chaque partition peut être amenée à évoluer indépendamment. Lorsque les dispositifs peuvent à nouveau communiquer ensemble, ils doivent avoir une connaissance cohérente du réseau.

4 Gestion distribuée de l'évolution sécurisée des réseaux domestiques

Pour marquer la frontière des réseaux domestiques tout en répondant aux contraintes de topologie dynamique et de connectivité erratique, le mécanisme que nous proposons est totalement distribué : il n'y a pas d'élément central (tel qu'un serveur central d'authentification ou une autorité centrale de certification), ni d'information secrète partagée (telle qu'une clé symétrique de réseau). Chaque dispositif se considère l'élément central et gère localement sa propre connaissance du réseau domestique.

4.1 Identité prouvable

Chaque dispositif du réseau domestique dispose d'une identité prouvable, qui lui permet d'être identifié et de s'authentifier auprès des autres dispositifs de son réseau. Nous appelons "identité prouvable" une identité qu'il est facile de vérifier, mais très difficile d'usurper, et qui permet la mise en place sécurisée de matériel

cryptographique. Par exemple, la clé publique d'une paire de clés publique / privée peut être utilisée comme identité prouvable : un dispositif prétendant être identifié par sa clé publique peut signer un *challenge* en utilisant sa clé privée, et est le seul à pouvoir déchiffrer un message qui a été chiffré avec sa clé publique. De plus, en se servant de leurs identités prouvables respectives, deux dispositifs peuvent créer un canal de communication sécurisé, leur permettant notamment de mettre en place des clés de session symétriques en utilisant un protocole de *key-agreement* tel que [2] par exemple. Ces clés de session point-a-point servent aux authentifications subséquentes et pour sécuriser les communications (authenticité et confidentialité) entre les deux dispositifs.

Citons aussi CAM [13] et SUCV [14] comme étant d'autres exemples de mécanismes d'identité prouvable, utilisés notamment dans le cadre de la mobilité IPv6, et pour lesquels à chaque dispositif est associé le résumé cryptographique de sa clé publique.

Du point de vue de la sécurité, chaque dispositif connaît les autres dispositifs de son réseau domestique uniquement par leurs identités prouvables. Par conséquent, si un dispositif change d'identité prouvable, il sera considéré par les autres dispositifs (qu'ils soient de son réseau domestique ou non) comme étant un tout autre dispositif.

4.2 Connaissance locale du réseau domestique

Chaque dispositif gère localement la connaissance qu'il a des dispositifs appartenant à son réseau domestique. Pour maintenir cette connaissance à jour, un dispositif se base en premier lieu sur les informations fournies par son utilisateur / administrateur, qui initie les changements dans le réseau (ajout, retrait ou bannissement d'un dispositif). Ces opérations seront présentées dans la section 4.3. Chaque dispositif échange aussi les informations qu'il possède avec les autres dispositifs de son réseau domestique, afin que tous en ait une vision cohérente. Ces échanges d'information seront présentés dans la section 4.4.

À l'échelle d'un réseau domestique donné, un dispositif, représenté par son identité prouvable, peut être :

- **Inconnu**, si ce dispositif n'appartient pas au réseau domestique et n'y a jamais appartenu.
- **Dedans**, si ce dispositif fait actuellement partie du réseau domestique.
- **Sorti**, si ce dispositif a autrefois fait partie du réseau domestique, mais n'y appartient plus, qu'il ait été retiré ou banni.

Pour un dispositif donné, ces trois états sont strictement ordonnés dans le temps : un dispositif est **Inconnu** jusqu'à ce qu'il soit inséré dans le réseau domestique et donc **Dedans**. Il reste **Dedans** jusqu'à ce qu'il soit **Sorti** du réseau par l'utilisateur. Nous considérons qu'un dispositif (c'est à dire une identité prouvable) **Sorti** du réseau domestique ne pourra pas y être ré-inséré avec la même identité prouvable : il devra d'abord générer une nouvelle identité prouvable, et être inséré sous cette identité.

Localement, un dispositif a peut connaître un autre dispositif b qui appartient ou a appartenu à son réseau domestique dans trois états différents :

- **Confiance Mutuelle**, noté MT pour l'anglais *Mutual Trust*
- **Confiance Unilatérale**, noté UT pour l'anglais *Unilateral trust*
- **Aucune Confiance**, noté DT pour l'anglais *DisTrust*

Un dispositif a connaît un autre dispositif b comme MT lorsque a sait que b appartient à son réseau domestique et que a sait que b sait que a appartient au réseau domestique de b. a et b ont déjà été mis en présence, et a possède un certificat signé par b qui prouve que b considère a comme étant dans son réseau domestique. Remarquons que, par symétrie, ceci est aussi vrai pour b.

a connaît b comme UT lorsque a sait que b appartient au réseau domestique de a, mais n'a jamais été mis en présence de b. Par conséquent, a ne sait pas si b sait que a appartient au réseau domestique de b, et devra peut être en apporter la preuve à b, comme décrit en section 4.4. Chaque dispositif qui, à l'instar de b, est connu par a comme étant UT, a été présenté à a par un dispositif que nous nommerons c que a connaît comme MT (cf. section 4.4). c, pour sa part, connaît b comme MT ou UT. Pour chaque dispositif connu comme UT par a, a dispose d'une chaîne de certificats, fournie par c, qui prouve que, par transitivité, b considère que a est dans son réseau domestique. a présentera cette chaîne de certificats à b lorsqu'ils seront mis en présence, pour lui prouver qu'ils appartiennent au même réseau domestique, et que tout les deux se passent l'un l'autre MT.

Enfin, a connaît b comme DT si a sait que b a appartenu au réseau domestique, mais n'y appartient plus. a n'acceptera plus de certificats émis par b comme étant des preuves d'appartenance au réseau domestique.

Ces états UT, MT et DT sont une représentation locale de l'état réel du dispositif concerné au sein du réseau domestique. Un dispositif a qui connaît b comme étant MT ou UT considère que b appartient à son réseau domestique, ce qui correspond à l'état réel **Dedans**. Par contre, tout dispositif inconnu ou connu comme étant DT n'est pas considéré comme appartenant au réseau domestique. DT est l'état local connu d'un dispositif **Sorti** du réseau. Par construction, les ensembles MT, UT et DT sont disjoints : a ne peut connaître b comme étant dans deux états à la fois.

Ces états locaux représentatifs de l'état réel d'un dispositif dans son réseau domestique sont eux aussi ordonnés. Un dispositif a qui connaît un autre dispositif b comme MT ne peut pas le faire passer à UT (ils ne peuvent pas s'être déjà rencontrés, puis ne s'être jamais rencontrés). Il ne peut pas non plus le faire passer de DT à UT ou MT, car, comme nous l'avons vu, un dispositif qui a été sorti du réseau domestique ne peut pas y être réinséré.

4.3 Évolution sécurisée du réseau domestique

L'évolution sécurisée du réseau domestique est initiée sur n'importe quel dispositif a appartenant à ce réseau par l'autorité locale de a : il ou elle informe

a qu'un dispositif doit être inséré, retiré ou banni. C'est la seule et unique fois que l'utilisateur est impliqué au sujet de cette évolution : par la suite, l'information d'évolution sera transmise par a aux autres dispositifs connus par a comme étant dans son réseau domestique (cf. section 4.4). Notons toutefois que rien n'empêche un utilisateur d'informer plusieurs dispositifs différents de la même évolution du réseau domestique.

Pour éviter qu'un attaquant n'insère de fausses informations sur un dispositif (insertion, bannissement ou retrait), l'utilisateur doit s'authentifier sur le dispositif qu'il a choisi pour initier l'évolution. Le mécanisme d'authentification utilisé sur chaque dispositif est strictement local à ce dispositif, qui peut donc utiliser celui qui lui est le plus approprié (code PIN, carte à puce, biométrie, etc.). Chaque dispositif peut en outre avoir un mécanisme d'authentification totalement différent de celui utilisé sur les autres dispositifs du réseau domestique. Ainsi, il n'existe aucune identité globale de l'utilisateur à l'échelle du réseau domestique, et notre proposition est réellement décentralisée. Du point de vue de la survie, cette solution offre un avantage important : si le mécanisme d'authentification d'un dispositif est compromis (par exemple, si un code PIN est découvert), cela n'a pas d'influence directe sur les autres dispositifs du réseau domestique, et la compromission ne s'étend pas au mécanismes d'authentifications utilisés sur ceux-ci.

Initialisation

Dans son état initial, un dispositif a est seul dans son propre réseau domestique. Pendant la phase d'initialisation, a génère une nouvelle identité prouvable, et marque celle-ci comme MT dans sa base de connaissance locale. Du point de vue de la sécurité, la connaissance qu'a a de son réseau domestique est valide : a ne considère comme étant dans son réseau domestique que les dispositifs qui y appartiennent réellement (ici, lui même), et ne considère pas comme n'y étant plus des dispositifs qui y appartiennent encore. Elle est aussi cohérente avec la connaissance des autres dispositifs qu'il sait appartenir à son réseau domestique (ici, lui même).

Remarquons que, sur demande et après authentification de l'utilisateur, un dispositif peut lancer une ré-initialisation : il efface de manière irréversible son identité prouvable précédente, en génère une nouvelle, vide sa base locale de connaissance, et insère sa nouvelle identité prouvable comme MT. Après une ré-initialisation, le dispositif est dans un état équivalent à celui dans lequel il se trouve après l'initialisation. Du point de vu des autres membres de son ancien réseau domestique, un dispositif ré-initialisé n'a plus rien à voir avec celui qu'il était avant la ré-initialisation, et est donc un dispositif totalement différent.

Insertion d'un dispositif

Du fait du parti que nous avons pris de proposer une solution totalement décentralisée, un dispositif a se basera uniquement sur sa connaissance locale pour décider si un autre dispositif b appartient ou non à son réseau domestique. De ce fait, pour que deux dispositifs a et b communiquent, chacun d'entre eux doit

savoir que l'autre appartient à son réseau domestique, c'est à dire le connaître comme MT ou UT. Par conséquent, l'insertion doit être exprimée sur les deux dispositifs : lorsqu'un utilisateur insère a dans le réseau domestique de b (en agissant sur b), il doit aussi agir sur a pour y insérer b, faute de quoi a ne communiquera pas avec b. Remarquons que lorsque a et b ont déjà d'autres dispositifs dans leurs réseaux domestiques respectifs avant de s'insérer l'un l'autre, l'opération d'insertion est en fait une fusion des deux réseaux domestiques, à l'issue de laquelle les deux réseaux domestiques de a et de b n'en forment plus qu'un.

L'insertion d'un dispositif a dans le réseau domestique d'un autre dispositif b consiste à insérer l'identité prouvable de a comme étant MT dans la base de connaissance de b. De la même manière, l'utilisateur doit insérer l'identité prouvable de b comme étant MT dans la base de connaissance de a.

Bien entendu, pour des raisons de facilité d'utilisation, on ne peut demander à l'utilisateur de rentrer la chaîne hexadécimale correspondant à la clé publique du dispositif à insérer, ni même son résumé cryptographique. La littérature propose plusieurs mécanismes pour résoudre ce problème. Stajano et Anderson ont par exemple proposé le *Resurrecting Duckling* [10,11] pour permettre l'appariement sécurisé de deux dispositifs : l'utilisateur place les deux dispositifs côte à côte, et leur demande de s'apparier. Ceux-ci échangent alors, en utilisant un canal supposé physiquement sûr, les clés qu'il utiliseront par la suite pour sécuriser leurs communications. Des améliorations ont par la suite été proposées [4,12], réduisant les hypothèses sur les propriétés du canal supposé sûr.

Lorsqu'un dispositif a insère un autre dispositif b dans son réseau domestique, il le marque comme MT. De plus, en utilisant son identité prouvable, a génère et délivre à b un certificat portant sur l'identité prouvable de b et qui prouve que b appartient au réseau domestique de a. L'usage qui sera fait de ce certificat sera vu plus avant.

Retrait

L'opération de retrait consiste à retirer un dispositif du réseau domestique alors qu'il est encore disponible et sous contrôle.

Tout d'abord, l'utilisateur s'authentifie sur le dispositif concerné b, et lui demande de quitter le réseau domestique. b informe alors les dispositifs de son réseau domestique qu'il peut contacter qu'il quitte le réseau domestique définitivement en envoyant à chacun un message de retrait authentifié avec la clé symétrique qu'il partage avec lui. Lors de la réception d'un message de retrait en provenance de b (qui appartient à son réseau domestique), un dispositif a vérifie tout d'abord l'authenticité de ce message. Puis a passe b à DT dans sa base de connaissance locale. Dès lors, a ne considère plus b comme appartenant à son réseau domestique.

Puis b se ré-initialise comme décrit précédemment. Ainsi, il ne considére plus les dispositifs du réseau domestique auquel il appartenait comme étant de son réseau domestique, et eux même ne peuvent plus accéder aux services qu'il offre.

Remarquons que si aucun dispositif n'est disponible lors du retrait de b, l'utilisateur devrait, même si cela n'est pas strictement nécessaire, informer plus tard l'un des dispositifs que b a quitté le réseau, en utilisant l'opération de bannissement expliquée ci-après. Ainsi, si la partie privée de l'identité prouvable de b est cassée *a posteriori*, un attaquant ne pourra malgré tout pas s'insérer dans le réseau en usurpant l'identité de b.

Bannissement
Le bannissement d'un dispositif b a lieu lorsque b doit être retiré du réseau domestique mais que l'utilisateur ne peut pas y accéder. Cette opération a par conséquent lieu sur un autre dispositif a appartenant au réseau domestique dont b doit être banni. Pour bannir b, l'utilisateur s'authentifie sur a, et déclare simplement que b est désormais banni. a passe alors b de UT ou MT à DT.

Remarquons que le bannissement peut être utilisé à la place du retrait. Il faut alors en plus ré-initialiser manuellement le dispositif banni.

4.4 Gestion distribuée de la cohérence des connaissances locales

Il serait particulièrement contraignant pour l'utilisateur de devoir spécifier la politique sur chacun des dispositifs du réseau domestique, car il devrait informer chaque dispositif lors de d'une insertion, d'un retrait, ou d'un bannissement.

Pour accroître la facilité d'utilisation et assurer la cohérence des connaissances locales des dispositifs du réseau domestique, chaque dispositif échange des informations avec les autres dispositifs de son réseau domestique, lorsqu'ils sont disponibles. Comme nous l'avons déjà indiqué, tous les dispositifs d'un même réseau domestique sont liés par une relation à long terme et peuvent légitimement se faire confiance quant aux informations qu'ils s'échangent sur les insertions, retraits et bannissements survenus dans le réseau.

De ce fait, la relation de confiance entre dispositifs est transitive : si a considère que b appartient à son réseau domestique et si b considère que c appartient à son réseau domestique, alors a considère que c appartient à son réseau domestique. Cette transitivité est valide car tous les dispositifs d'un même réseau domestique sont soumis à la même politique et se basent donc sur les même critères pour insérer un dispositif dans le réseau domestique. Par conséquent, lorsque a insère b dans son réseau, cette insertion est valide pour tous les dispositifs considérant que a est dans leur réseau. De même, un dispositif faisant confiance aux certificats émis par a fera confiance à ceux émis par b.

Échange d'informations entre dispositifs se connaissant réciproquement comme MT
En plus de la transitivité de la confiance entre dispositifs, la gestion de cohérence des connaissances entre dispositifs d'un même réseau domestique repose sur :

- L'ordre strict des états d'un dispositif dans un réseau domestique.
- Le fait que toute information locale d'un dispositifs reçue de l'utilisateur est supposée vraie, car l'utilisateur a dû s'authentifier sur le dispositif.

De ce fait, quand deux dispositifs du même réseau domestique ont une connaissance différente au sujet d'un troisième (par exemple, l'un le connaît comme **Sorti** alors que l'autre le connaît comme **Dedans**), celui qui connaît l'état le plus avancé (ici, **Sorti**) est forcément le plus à jour, puisque sa connaissance a été acquise légitimement et que les états sont ordonnés. Deux cas sont possible :

- le dispositif a reçu cette information de son utilisateur légitime.
- le dispositif a reçu cette information d'un autre dispositif du réseau domestique.

Dans le second cas, ce dispositif peut l'avoir lui même reçu d'un autre dispositif du réseau domestique, ou de son utilisateur. Par récurrence, cette information provient forcément d'un dispositif du réseau domestique qui l'a reçu d'une autorité légitime.

L'échange d'informations entre deux dispositifs a et b se connaissant l'un l'autre comme MT a lieu lorsque la connaissance de l'un d'entre eux est modifié pour une cause autre que l'échange d'informations entre eux, ou quand a et b sont mis en présence après avoir été déconnectés (la connaissance de chacun ayant pu évoluer). L'échange d'informations suit toujours le même algorithme, et toutes les communications sont protégées par les clés symétriques partagées par a et b .

Tout d'abord, a et b s'échangent les identités prouvables des dispositifs qu'ils connaissent comme DT. Chacun d'entre eux marque DT chaque dispositif c qu'il ne connaissait pas en tant que tel.

a et b comparent ensuite les dispositifs qu'ils connaissent MT. Bien que l'opération soit symétrique, nous présentons, dans un souci de clarté, l'obtention des informations de b par a. Pour chaque dispositif c connu MT par b mais pas par a, a insère c comme UT dans sa base de connaissance locale. Il stocke en même temps le certificat que b a émis lorsque b a inséré a comme MT (cette étape sera décrite un peu plus loin). Ainsi, lorsque a sera mis en présence de c, il pourra fournir à c le certificat en question. b connaissant c comme MT, cela signifie qu'ils se sont déjà rencontrés, et que c considère b comme MT. Par conséquent, il acceptera le certificat comme preuve que a appartient bien à son réseau domestique, et tout deux (a et c) pourront se passer l'un l'autre MT lorsqu'ils seront mis en présence, comme présenté plus loin.

Enfin, a et b comparent les informations qu'ils ont sur les dispositifs UT. Bien que l'opération soit ici aussi symétrique, nous présentons seulement l'obtention des informations de b par a. Pour chaque dispositif c connu UT par b mais pas par a, a insère c comme UT dans sa base de connaissance locale. Il obtient aussi la chaîne de certificats dont b dispose pour c et qui prouve, par transitivité de confiance que c considère b comme étant dans son réseau domestique. a y ajoute le certificat prouvant que b considère que a est dans son réseau domestique. De ce fait, a possède désormais une chaîne de certificats qu'il pourra remettre à c lorsqu'ils seront mis en présence, et qui prouve qu'ils appartiennent au même réseau domestique. Ainsi, tout deux (a et c) pourront se passer l'un l'autre MT.

Mise en présence d'un dispositif connu comme UT
Lorsqu'un dispositif a est mis en présence d'un autre dispositif b qu'il connaît comme UT, il va fournir à b la chaîne de certificats qui prouve que a appartient au réseau domestique de b, ainsi qu'un certificat généré en utilisant son identité prouvable, et prouvant que a considère que b est dans son réseau domestique.

Lorsque b reçoit une chaîne de certificats, il vérifie que cette chaîne commence bien par un dispositif qu'il sait être dans son réseau domestique, puis il vérifie chacun des certificats. Si la chaîne de certificats fournie par a contient un certificat émis par un dispositif **Sorti** du réseau domestique (en d'autres termes, que b connaît comme DT), b ne doit pas insérer a comme MT, pour des raisons évidentes de sécurité.

Notons néanmoins qu'il se pourrait que a appartienne en fait réellement au réseau domestique. Par exemple, a peut avoir été inséré légitimement dans le réseau domestique par un dispositif e, qui a été banni avant que a ait été mis en présence d'autres dispositifs du réseau domestique. Pour sa part, a ne sait pas que e a été banni : il connaît encore e comme MT. Dans ce cas, a connaît les dispositifs du réseau domestique comme UT, mais est inconnu d'eux. L'utilisateur doit simplement insérer explicitement a dans le réseau domestique en utilisant un autre dispositif f qui y appartient encore. Lors de l'échange d'informations par a et f, a apprend de f que e est maintenant DT, et sa connaissance locale du réseau domestique est à jour.

Si tous les certificats de la chaîne reçue par b sont valides, b marque a comme MT, stocke le certificat reçu de a, et lui envoie un certificat généré en utilisant son identité prouvable, prouvant que a est dans le réseau domestique de b. Lorsque a reçoit ce certificat, il marque b comme MT. a et b établissent alors des clés symétriques pour sécuriser leurs communications subséquentes. Enfin, ils échangent des informations au sujet des dispositifs de leur réseau domestique, comme expliqué précédemment.

5 Conclusion

Nous avons présenté dans cet article un mécanisme sécurisé, facile d'utilisation et totalement distribué permettant de marquer la frontière d'un réseau domestique. Ce mécanisme offre les différentes opérations d'évolutions nécessaires dans les réseaux domestiques, tout en tenant compte de la topologie dynamique de ces réseaux et de l'interconnexion erratique des dispositifs qui les composent.

Le rôle de l'utilisateur se limite à exprimer la politique. Celle-ci peut être exprimée simplement sur n'importe quel dispositif du réseau domestique, et sa diffusion sécurisée est ensuite prise en charge par les dispositifs composant le réseau. Ainsi, tous les dispositifs d'un réseau domestique qui peuvent communiquer à un instant donné atteignent une vision cohérente et à jour de la frontière.

Il nous semble désormais intéressant de se pencher sur la gestion de dispositifs "invités" dans le réseau domestique, c'est à dire des dispositifs qui doivent être insérés temporairement, n'ont le droit d'accéder qu'à certains services, etc.

De plus, maintenant que nous avons défini le périmètre de sécurité d'un réseau dont les dispositifs sont tous placés sous la même politique, il serait intéressant de considérer les politiques divergentes à l'intérieur d'un même réseau domestique. Par exemple, certains services ou certaines données à l'intérieur du réseau peuvent être réservés à un sous-ensemble d'utilisateurs. Comment dans ce cas gérer l'expression de ces règles et assurer le contrôle d'accès ?

Références

1. HAVi Inc., *The HAVi Specification*, may 2001.
2. Whitfield Diffie, Paul C. van Oorschot and Michael J. Wiener, Authentication and Authenticated Key Exchange, *Design, Codes and Cryptography*, 1992.
3. Nicolas Prigent, Christophe Bidan, Olivier Heen et Alain Durand, Sécurité des réseaux domestiques : optimaux les grands remèdes ?, *Actes du Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC)*, 2003.
4. D. Balfanz, D. Smetters, P. Stewart et H. Wong, Talking to strangers : Authentication in adhoc wireless networks, *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, Février 2002, `citeseer.nj.nec.com/balfanz02talking.html`
5. R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff et N. R. Mead, Survivability : Protecting Your Critical Systems, *IEEE Internet Computing*, Vol. 3 number 6, 1999.
6. S. W. Smith, Humans in the Loop : Human-Computer Interaction and Security, *IEEE Security & Privacy*, Juin 2003, IEEE Press.
7. S. Corson et J. Macker, RFC 2501 : Mobile Ad hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations, janvier 1999.
8. Erik Guttman, Autoconfiguration for IP Networking : Enabling Local Communication, *IEEE Internet Computing*, mai 2001.
9. The UPnP Initiative, *The UPnP Forum*, `http://www.upnp.org/`
10. Frank Stajano, The Resurrecting Duckling – What Next ?, *Lecture Notes in Computer Science* Vol. 2133, pp 204–211, Springer 2001, `citeseer.nj.nec.com/stajano00resurrecting.html`
11. Frank Stajano and Ross Anderson, The Resurrecting Duckling : Security Issues for Ad-hoc Wireless Networks, In *7th International Workshop on Security Protocols*, pp 172–194, 1999, `citeseer.nj.nec.com/stajano99resurrecting.html`
12. L. Feeney, B. Ahlgren et A. Westerlund, Spontaneous networking : an application-oriented approach to ad hoc networking, *IEEE Communications Magazine*, juin 2001, `citeseer.nj.nec.com/feeney01spontaneous.html`
13. G. O'Shea et M. Rose, Child-proof authentication for MIPv6 (CAM), *ACM SIGCOMM Computer Communication Review*, Vol. 31, number 2, 2001, pp 4–8, `http://doi.acm.org/10.1145/505666.505668`, ACM Press.
14. C. Montenegro et C. Castelluccia, Statistically Unique and Cryptographically Verifiable (SUCV), *NDSS'02*, Février 2002.

Héritage de privilèges dans le modèle Or-BAC : application dans un environnement réseau

Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], and Alexandre Miège[1,2]

[1] GET/ENST Bretagne/Département RSM,
BP 78, 2 rue de la Châtaigneraie, 35512 Cesson Sévigné Cedex, France
[2] GET/ENST/Département INFRES,
46, rue Barrault, 75634 Paris Cedex 13, France
{frederic.cuppens,nora.cuppens,alexandre.miege}@enst-bretagne.fr

Résumé Le concept de hiérarchie de rôles fut introduit pour la première fois dans le modèle RBAC (*Role Based Access Control*). L'héritage des permissions est associée à cette hiérarchie ce qui permet de spécifier des politiques de sécurité de façon modulaire. Cet article propose de généraliser cette approche dans le contexte du modèle Or-BAC (*Organization Based Access Control*). Nous définissons d'abord des hiérarchies de rôles, de vues et d'activités et formalisons un mécanisme d'héritage pour chaque hiérarchie. Nous définissons ensuite une hiérarchie d'organisations. Nous montrons que ce concept fournit un moyen efficace pour dériver, de la spécification d'une politique de sécurité organisationnelle, les politiques de sécurité technique de composants de sécurité. Nous illustrons notre approche dans le cadre des politiques de sécurité réseau, en particulier pour configurer des pare-feux (firewall).

1 Introduction

Dans le domaine de la programmation orientée objet, l'héritage est un moyen efficace pour concevoir une application de façon modulaire. Un concept similaire est utilisé dans le modèle RBAC [4] lorsqu'une hiérarchie de rôles est définie et associée à un mécanisme d'héritage des permissions. Cette hiérarchie de rôles est utile pour structurer la spécification d'une politique de sécurité.

Cependant, le concept de hiérarchie de rôles n'est pas exempt d'ambiguïtés [35,36,42]. Certaines de ces ambiguïtés sont directement liées au concept de rôle lui-même. Si l'on examine les exemples proposés dans [4], plusieurs interprétations du concept de rôles sont possibles. A la base, un rôle permet à un sujet qui est affecté à ce rôle d'effectuer certaines activités. C'est le cas de rôles tels que *médecin*, *infirmier* ou *secrétaire médicale*. Cependant, dans certains exemples, un rôle est indissociable d'une *organisation* particulière [38]. Par exemple, on peut considérer des rôles tels que *infirmier dans un département de cardiologie* ou *infirmier dans une équipe de réanimation*. Cette distinction est importante car les permissions affectées à un rôle peuvent dépendre de l'organisation. Ainsi, un infirmier peut avoir des permissions différentes s'il effectue son activité dans un département de cardiologie ou dans une équipe de réanimation. Un rôle peut

également être associé à l'activité d'administration de l'organisation. *Directeur d'hôpital* ou *chef du département de cardiologie* sont des exemples de tels rôles.

Comme nous le verrons dans la suite, ces différentes interprétations du concept de rôle ne se comportent pas de la même façon vis-à-vis de l'héritage des permissions. C'est la raison pour laquelle il est important de disposer d'un modèle qui permet de rendre explicite ces différences d'interprétation.

Nous proposons d'analyser ce problème dans le contexte du modèle Or-BAC [24]. Dans Or-BAC, une organisation correspond à n'importe quelle entité qui a la responsabilité de gérer un ensemble de règles de sécurité (permissions ou interdictions). Par exemple, un hôpital particulier est une organisation. Un composant de sécurité, tel qu'un firewall, peut également être assimilé à une organisation dans la mesure où il gère un ensemble de règles de sécurité.

La définition d'un rôle dans le modèle Or-BAC est toujours associée à une organisation particulière. Ceci permet d'éviter certaines ambiguïtés lorsqu'on définit des hiérarchies de rôles et qu'on leur associe un mécanisme d'héritage de permissions.

Le modèle Or-BAC introduit deux autres concepts : les *activités* et les *vues*. La politique de sécurité associée à une organisation particulière est définie par des permissions (ou interdictions) pour les rôles de réaliser des activités sur des vues. Dans la suite, nous proposons de définir des hiérarchies d'activités et de vues et modélisons les mécanismes d'héritage associés à ces hiérarchies.

Enfin, dans Or-BAC, on peut également définir des hiérarchies d'*organisations*. Il s'agit peut-être de la contribution principale de cet article car ces hiérarchies fournissent un moyen très efficace pour structurer la spécification de la politique de sécurité. On peut ainsi commencer par la spécification de la politique de sécurité d'une organisation de "haut niveau" telle qu'un hôpital. On peut ensuite décomposer cette organisation en faisant apparaître les différentes sous-organisations de cet hôpital et de proche en proche, arriver à la spécification de la politique de sécurité technique des composants de sécurité tels qu'un firewall.

Dans cet article, nous proposons d'analyser et de formaliser l'héritage des permissions et des interdictions dans ces différentes hiérarchies. Le reste de l'article est organisé de la façon suivante. La section 2 rappelle les principaux concepts de Or-BAC. La section 3 présente les différentes hiérarchies associées respectivement aux rôles, activités et vues. La section 4 étudie les hiérarchies d'organisations. Dans la section 5, nous récapitulons comment spécifier une politique de sécurité dans Or-BAC lorsque les hiérarchies sont utilisées. La section 6 montre comment notre approche s'applique à la spécification de politiques de sécurité réseau. Enfin, la section 7 conclut et suggère plusieurs extensions à ce travail.

2 Or-BAC

2.1 Concepts de base de Or-BAC

Or-BAC [24] est un modèle de contrôle d'accès fondé sur le concept d'organisation. Dans Or-BAC, plusieurs organisations peuvent simultanément spécifier

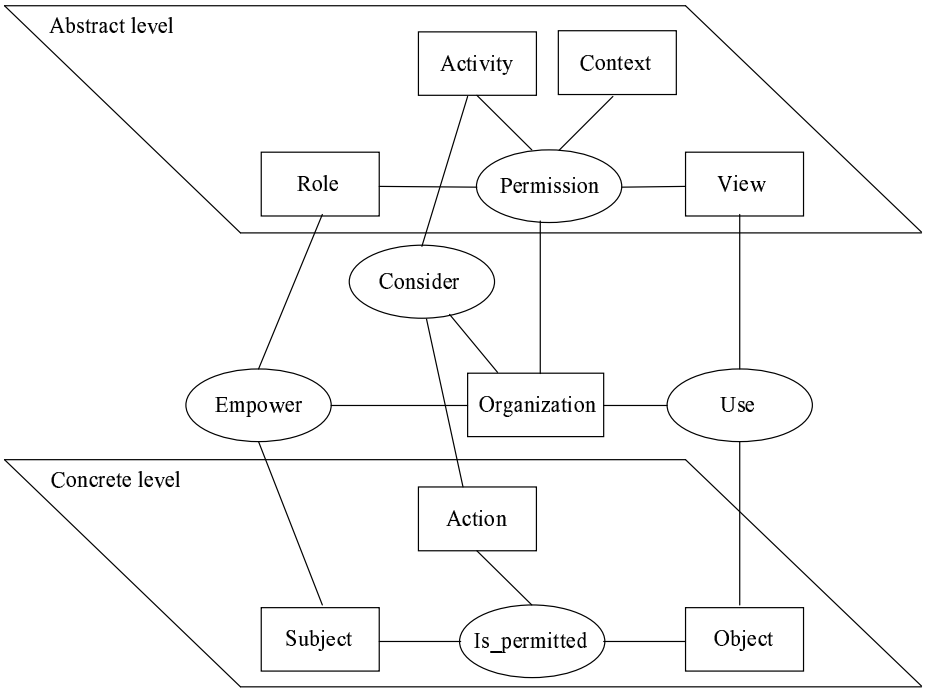


Fig. 1. Structure du modèle Or-BAC

leur propre politique de contrôle d'accès en utilisant huit entités de base (voir figure 1) : Org(organization), $Role$, $Activity$, $View$, $Subject$, $Action$, $Object$ et $Context$. Les prédicats utilisés dans Or-BAC pour modéliser les associations entre ces huit entités sont récapitulés dans la table 2.

Comme nous l'avons mentionné dans l'introduction, une organisation est n'importe quelle entité qui gère un ensemble de règles de sécurité. Un sujet est une entité à qui l'on peut affecter un rôle. Nous supposons que $Org \subseteq Subject$ de sorte qu'il est possible d'affecter un rôle à une organisation. Par exemple, les rôles "département de cardiologie" ou "service des urgences" peuvent être affectés à certaines organisations.

En utilisant l'entité $Role$, nous disposons d'un moyen efficace pour structurer les sujets et mettre à jour les politiques de sécurité lorsque de nouveaux sujets sont insérés dans le système. Une démarche similaire nous conduit à introduire une nouvelle entité pour structurer les objets : l'entité $View$. Intuitivement, une vue correspond à un ensemble d'objets qui satisfont une même propriété[3].

Une autre entité est utilisée pour structurer les actions : l'entité $Activity$. En considérant que les rôles regroupent les sujets qui remplisssent les mêmes fonc-

[3] Il existe clairement une analogie entre cette notion de vue et celle utilisée dans les bases de données relationnelles

Nom du Prédicat	Domaine	Description
$Relevant\_role$	$Org \times Role$	Si org est une organisation et r est un rôle, alors $Relevant\_role(org, r)$ signifie que jouer le rôle r a un sens dans l'organisation org. Ex : $Relevant\_role(H, physician)$
$Relevant\_activity$	$Org \times Activity$	Si org est une organisation et a est une activité, alors $Relevant\_activity(org, a)$ signifie que réaliser l'activité a a un sens dans l'organisation org. Ex : $Relevant\_activity(H, consult)$
$Relevant\_view$	$Org \times View$	Si org est une organisation et v est une vue, alors $Relevant\_view(org, v)$ signifie que l'utilisation de la vue v a un sens dans l'organisation org. Ex : $Relevant\_view(H, medical\_record)$
$Empower$	$Org \times Subject \times Role$	Si org est une organisation, s est un sujet et r est un rôle, alors $Empower(org, s, r)$ signifie que org habilite le sujet s dans le rôle r. Ex : $Empower(H, John, physician)$
$Consider$	$Org \times Action \times Activity$	Si org est une organisation, α est une action et a est une activité, alors $Consider(org, \alpha, a)$ signifie que org considère que l'action α implante l'activité a. Ex : $Consider(H, "SELECT", consult)$
Use	$Org \times Object \times View$	Si org est une organisation, o est un objet et v est une vue, alors $Use(org, o, v)$ signifie que org utilise l'objet o dans la vue v. Ex : $Use(H, med\_27, medical\_record)$
$Hold$	$Org \times Subject \times Action \times Object \times Context$	Si org est une organisation, s est un sujet, α une action, o un objet et c un contexte, alors $Hold(org, s, \alpha, o, c)$ signifie que dans l'organisation org, le contexte c est satisfait entre le sujet s, l'action α et l'objet o. Ex : $\forall s, \forall \alpha, \forall o,$ $Hold(H, s, \alpha, o, working\_hours) \leftarrow (08:00 \leq time(GLOBAL\_CLOCK) \wedge time(GLOBAL\_CLOCK) \leq 19:00)$

Fig. 2. Prédicats de base de Or-BAC

tions et les vues correspondent à des ensembles d'objets qui satisfont une même propriété, les activités associent les actions qui partagent les mêmes principes.

Les sujets, les objets et les actions peuvent avoir des attributs. Ces attributs sont modélisés par un ensemble de prédicats binaires ayant la forme $att(ent, val)$ où ent est un sujet, un objet ou une action et val représente la valeur de l'attribut att de l'entité ent. Par exemple, si med_27 est un dossier médical, alors $name(med\_27, John)$ indique que med_27 est le dossier médical de John.

Nous supposons que $Subject \subseteq Object$ de sorte que l'on peut définir des vues sur les sujets que nous appelons *groupes*. Dans Or-BAC, il y a une distinction claire entre un rôle et un groupe. Des permissions peuvent être affectées à des rôles alors qu'un groupe est seulement un ensemble de sujets qui ont des propriétés communes[4]. Cependant, il est parfois commode d'affecter le même rôle à chacun des sujets appartenant à un certain groupe. Dans ce cas, nous proposons d'utiliser le prédicat $G\_Empower(org, group, role)$ et de spécifier la règle suivante :

- GE : $\forall org, \forall group, \forall role, \forall subject,$
  $Use(org, subject, group) \wedge$
  $G\_Empower(org, group, role)$
  $\rightarrow Empower(org, subject, role)$

Le modèle Or-BAC permet à l'administrateur de sécurité de spécifier que certaines permissions et interdictions ne s'appliquent que dans certains *contextes*. Pour cela, nous introduisons l'entité $Context$. Les contextes sont définis par des règles logiques qui concluent sur le prédicat $Hold$ (voir la table 2 qui donne l'exemple de contexte $working\_hours$). Nous disons que, dans l'organisation org, le sujet s réalise l'action α sur l'objet o dans le contexte c quand la condition $cond$ est satisfaite, c'est-à-dire lorsqu'il existe une règle[5] ayant la forme suivante : $Hold(org, s, \alpha, o, c) \leftarrow cond$. Pour une présentation plus détaillée de la notion de contexte dans Or-BAC, voir [39].

2.2 Permissions et interdictions

Les permissions et les interdictions dans Or-BAC sont définies à l'aide des prédicats présentés dans la figure 3. Une politique de contrôle d'accès est modélisée à deux niveaux différents : un niveau abstrait qui spécifie les permissions et les interdictions entre les rôles, les activités et les vues, et un niveau concret qui permet de dériver les permissions et les interdictions entre les sujets, les actions et les objets (voir figure 1).

Ces deux niveaux sont reliés de la façon suivante. Dans une organisation org, un sujet s a la permission d'effectuer une action α sur un objet o si (1)

[4] Certains modèles proposent d'affecter des permissions à des groupes de sujets. Cette démarche, qui contribue grandement à la confusion entre les notions de rôle et de groupe, est incorrecte et doit être évitée.

[5] On peut supposer que chaque contexte c est défini par une règle *unique* en utilisant le fait que $Hold(org, s, \alpha, o, c) \leftarrow cond_1$, ..., $Hold(org, s, \alpha, o, c) \leftarrow cond_n$ est équivalent à $Hold(org, s, \alpha, o, c) \leftarrow (cond_1 \vee ... \vee cond_n)$.

s est habilité dans un certain rôle r dans org et (2) α implante une certaine activité a dans org et (3) o est utilisé dans une certaine vue v dans org. Si ces trois conditions sont satisfaites et si (4) l'organisation org accorde au rôle r la permission de réaliser l'activité a sur la vue v dans le contexte c et si (5) dans l'organisation org, le contexte c est satisfait entre le sujet s, l'action α et l'objet o, alors une requête par le sujet s de réaliser l'action α portant sur l'objet o est acceptée. La dérivation des permissions concrètes à partir des permissions abstraites est donc modélisée par la règle suivante :

- RG_1 : $\forall org, \forall r, \forall a, \forall v, \forall s, \forall \alpha, \forall o,$
  $Permission(org, r, a, v, c) \wedge$
  $Empower(org, s, r) \wedge$
  $Consider(org, \alpha, a) \wedge$
  $Use(org, o, v) \wedge$
  $Hold(org, s, o, \alpha, c)$
  $\rightarrow Is\_permitted(s, \alpha, o)$

Une autre règle similaire (appelée RG_2) est utilisée pour dériver les *interdictions* concrètes à partir des interdictions abstraites.

Nom de Prédicat	Domaine	Description
$Permission$	$Org \times Role \times Activity \times View \times Context$	Si org est une organisation, r un rôle, a une activité, v une vue et c un contexte, alors $Permission(org, r, a, v, c)$ signifie que l'organisation org accorde au rôle r la permission de réaliser l'activité a sur la vue v dans le contexte c. Ex : $Permission(H, physician,$ $consult, medical\_record, working\_hours)$
$Prohibition$	$Org \times Role \times Activity \times View \times Context$	Si org est une organisation, r un rôle, a une activité, v une vue et c un contexte, alors $Prohibition(org, r, a, v, c)$ signifie que l'organisation org interdit au rôle r de réaliser l'activité a sur la vue v dans le contexte c. Ex : $Prohibition(H, nurse,$ $consult, medical\_record, night)$
$Is\_permitted$	$Subject \times Action \times Object$	Si s est un sujet, α une action et o un objet, alors $Is\_permitted(s, \alpha, o)$ signifie que s a la permission concrète de réaliser l'action α sur l'objet o. Ex : $Is\_permitted(John, "SELECT", med\_27)$
$Is\_prohibited$	$Subject \times Action \times Object$	Si s est un sujet, α une action et o un objet, alors $Is\_prohibited(s, \alpha, o)$ signifie que s a concrètement l'interdiction de réaliser l'action α sur l'objet o. Ex : $Is\_prohibited(Mary, "DELETE", med\_27)$

Fig. 3. Spécification des permissions et des interdictions dans Or-BAC

2.3 Contraintes

Exprimer des contraintes qui s'appliquent à une politique de contrôle d'accès fut proposé pour la première fois dans le modèle RBAC (plus précisément, dans le sous-modèle RBAC_2 [6]) et ensuite davantage analysé dans [34]. Pour spécifier des contraintes dans le modèle Or-BAC, nous introduisons un prédicat $error()$. Une contrainte est modélisée par une règle ayant $error()$ pour conclusion (comme le suggère [29,30]).

Par exemple, nous pouvons spécifier que, dans un hôpital H, un sujet ne peut pas être habilité à la fois dans les rôles *anesthésiste* et *chirurgien* :

- $\text{C}_1 : \forall s,$
  $Empower(H, s, anesthetist) \wedge Empower(H, s, surgeon)$
  $\rightarrow error()$

Dans la suite, nous considérons la contrainte suivante qui s'applique à toutes les organisations :

- $\text{C}_2 : \forall org, \forall s, \forall r,$
  $Empower(org, s, r) \wedge \neg Relevant\_role(org, r)$
  $\rightarrow error()$

La règle C_2 indique qu'une organisation org ne peut pas habiliter un sujet s dans un rôle r si le rôle r n'est pas défini dans l'organisation org.

Il y a d'autres règles semblables à C_2 mais concernant les activités (règle C_3), les vues (règle C_4), les permissions (règle C_5) et les interdictions (règle C_6).

3 Les hiérarchies dans une organisation

Dans le modèle Or-BAC, il est possible de définir des hiérarchies de rôles (comme suggéré dans [6]) mais aussi des hiérarchies de vues et d'activités. Chaque hiérarchie définit respectivement une relation d'ordre partiel sur l'ensemble des rôles, des vues et des activités. Afin de modéliser ces différentes hiérarchies, nous présentons dans cette section les règles générales d'héritage des permissions et des interdictions qui leur sont associées.

3.1 La hiérarchie de rôles

Nous nous attachons dans un premier temps à étudier la hiérarchie de rôles. Pour cela nous introduisons le prédicat $sub\_role(org, r_1, r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est un sous-rôle du rôle r_2.

Remarquons que la hiérarchie de rôles dépend de l'organisation. Ainsi, chaque organisation peut définir sa propre hiérarchie de rôles. L'héritage des permissions à travers la hiérarchie de rôles est modélisé par la règle suivante :

- $\text{RH}_1 : \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c,$
  $sub\_role(org, r_1, r_2) \wedge$
  $Permission(org, r_2, a, v, c)$
  $\rightarrow Permission(org, r_1, a, v, c)$

Cette règle signifie que si le rôle r_1 est un sous-rôle du rôle r_2 dans l'organisation org, alors toutes les permissions attribuées au rôle r_2 dans org sont également attribuées au rôle r_1.

L'héritage des interdictions est plus complexe. En effet, la relation qui lie les rôles et les sous-rôles dans une hiérarchie peut être interprétée de différentes façons. Nous considérons pour le moment deux types de relation :

- La relation de spécialisation/généralisation. Par exemple, le rôle *chirurgien* est une spécialisation du rôle *médecin*. Afin de modéliser ce type de relation, nous introduisons le prédicat $specialized\_role(org, r_1, r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est une spécialisation du rôle r_2.
- La relation de hiérarchie organisationnelle. Par exemple, le rôle *directeur de département* peut être défini comme hiérarchiquement supérieur au rôle *chef d'équipe*. Ce second type de relation est modélisé par le prédicat $senior\_role(org, r_1, r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est un rôle senior du rôle r_2, c'est-à-dire un rôle supérieur dans la hiérarchie organisationnelle.

Nous considérons que la relation $specialized\_role$ est incluse dans la relation sub_role :

- $\mathrm{RH}_2 : \forall org, \forall r_1, \forall r_2,$
  $specialized\_role(org, r_1, r_2)$
  $\rightarrow sub\_role(org, r_1, r_2)$

Par conséquent, la règle RH_1 s'applique à la hiérarchie de spécialisation des rôles. Les permissions sont alors héritées à travers cette hiérarchie. Les interdictions sont également héritées à travers la hiérarchie de spécialisation des rôles conformément à la règle d'héritage suivante :

- $\mathrm{RH}_3 : \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c,$
  $specialized\_role(org, r_1, r_2) \wedge$
  $Prohibition(org, r_2, a, v, c)$
  $\rightarrow Prohibition(org, r_1, a, v, c)$

Par exemple, le rôle *chirurgien* hérite de toutes les interdictions du rôle *médecin*. Cet héritage est tout à fait justifié dans la mesure où le rôle *chirurgien* est un cas particulier du rôle *médecin*.

En revanche, la relation $senior\_role$ n'est généralement pas incluse dans la relation sub_role. Par conséquent, il est possible d'avoir :

- $\exists org, \exists r_1, \exists r_2,$
  $senior\_role(org, r_1, r_2) \wedge \neg sub\_role(org, r_1, r_2)$

Considérons par exemple le rôle *directeur d'hôpital*, hiérarchiquement supérieur au rôle *médecin*. Dans certains hôpitaux, le rôle *directeur d'hôpital* correspond uniquement à une fonction administrative qui n'est pas réalisée par un médecin. Dans ce cas, il n'y a aucune raison de conclure que le rôle *directeur d'hôpital* est un sous-rôle du rôle *médecin*.

Supposons à présent que le rôle r_1 est un rôle senior du rôle r_2 et que r_1 est aussi un sous-rôle de r_2. Supposons également que le rôle r_1 a plus de pouvoir que le rôle r_2. Alors, la règle RH_1 est tout à fait justifiée car elle permet à r_1 d'hériter de toutes les permissions de r_2. En revanche, si r_1 hérite aussi des interdictions de r_2, nous sommes en contradiction avec le fait que le rôle r_1 a plus de pouvoir que r_2. C'est pourquoi nous considérons que dans le cas d'une hié-

rarchie organisationnelle, l'héritage des interdictions se fait dans le sens inverse de l'héritage des permissions :

- RH_4 : $\forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c,$
  $sub\_role(org, r_1, r_2) \wedge senior\_role(org, r_1, r_2) \wedge$
  $Prohibition(org, r_1, a, v, c) \wedge$
  $\rightarrow Prohibition(org, r_2, a, v, c)$

Considérons par exemple que le rôle *directeur de département* est un rôle senior et également un sous-rôle du rôle *chef d'équipe*. Alors, le rôle *directeur de département* hérite des permissions accordées au rôle *chef d'équipe* (règle RH_1) et ce dernier hérite des interdictions attribuées au rôle *directeur de département* (règle RH_4).

Ainsi, nous avons défini trois hiérarchies de rôle : *sub_role*, *specialized_role* (inclus dans *sub_role*) et *senior_role* (généralement non inclus dans *sub_role*). Si nous nous intéressons uniquement à l'héritage des permissions et des interdictions, nous pouvons nous limiter aux deux hiérarchies suivantes : *sub_role* et *specialized_role*. Les règles RH_1, RH_2 et RH_3 s'appliquent conformément à la hiérarchie *specialized_role*, et les règles RH_1 et RH_4 s'appliquent selon la hiérarchie constituée par la différence entre la relation *sub_role* et la relation *specialized_role*. Pour cela, nous remplaçons simplement dans la prémisse de la règle RH_4 la condition : $sub\_role(org, r_1, r_2) \wedge senior\_role(org, r_1, r_2)$
par la condition : $sub\_role(org, r_1, r_2) \wedge \neg specialized\_role(org, r_1, r_2)$.

Ajoutons tout de même que toutes les règles d'héritage présentées dans cette section peuvent être accompagnées d'exceptions. Par exemple, nous pouvons considérer que dans l'hôpital H, les médecins ont l'interdiction de consulter les dossiers médicaux des patients qui ne sont pas les leurs. En appliquant la règle RH_3, nous obtenons que les chirurgiens héritent de cette même interdiction. Néanmoins, il est possible de spécifier explicitement, et ainsi définir une exception, que dans l'hôpital H, les chirurgiens ont la permission de consulter tous les dossiers médicaux, quel que soit le patient. La gestion des exceptions est traitée à la section 5.2.

3.2 La hiérarchie d'activités

Nous définissons dans cette section l'héritage entre les activités. Dans chaque organisation, les activités sont structurées sous forme de hiérarchies. La modélisation de ce type de relation hiérarchique est faite au moyen du prédicat $sub\_activity(org, a_1, a_2)$ qui signifie : dans l'organisation org, l'activité a_1 est une sous-activité de a_2.

La sémantique attribuée à cette hiérarchisation est la spécialisation. Ainsi, dans l'organisation hôpital H, l'activité *gestion* (des dossiers médicaux par exemple) est spécialisée en trois activités *création*, *consultation* et *mise-à-jour*. La structure hiérarchique associée à l'activité *gestion* est exprimée au moyen du prédicat *sub_ activity* défini ci-dessus : *sub_ activity(H,création,gestion)*, *sub_ activity(H,consultation,gestion)* et *sub_ activity(H,mise-à-jour,gestion)*.

A cette hiérarchie est associé un héritage des permissions, modélisé par la règle suivante :

– $\mathrm{AH}_1 : \forall org, \forall r, \forall a_1, \forall a_2, \forall v, \forall c,$
$Permission(org, r, a_2, v, c) \wedge$
$sub\_activity(org, a_1, a_2)$
$\rightarrow Permission(org, r, a_1, v, c)$

Supposons par exemple que, dans l'hôpital H, les médecins ont la permission de gérer les dossiers médicaux de leurs patients. En appliquant la règle AH_1 nous pouvons conclure que les médecins ont également la permission de créer, de consulter et de mettre à jour les dossiers médicaux de leurs patients.

De plus, nous considérons qu'une règle similaire, appelée AH_2, s'applique à l'héritage des interdictions. Si dans l'hôpital H les infirmiers ont l'interdiction de gérer les dossiers médicaux, alors, par application de la règle AH_2, nous obtenons que les infirmiers ont également l'interdiction de créer, de consulter et de mettre à jour ces dossiers médicaux.

3.3 La hiérarchie de vues

Comme pour les rôles et les activités, l'ensemble des vues est structuré par des hiérarchies dépendant de l'organisation. Cette hiérarchisation est modélisée par le prédicat $sub\_view(org, v_1, v_2)$ qui signifie : dans l'organisation org, la vue v_1 est une sous-vue de la vue v_2.

La sémantique que nous attribuons à cette relation hiérarchique est la spécialisation. Cette structuration est en fait proche de la hiérarchie d'héritage de classes utilisée dans les approches orientées objet (la relation Isa).

Dans le contexte du modèle Or-BAC, on associe l'héritage des permissions aux hiérarchies de vues. La règle suivante modélise cet héritage :

– $\mathrm{VH}_1 : \forall org, \forall r, \forall a, \forall v_1, \forall v_2, \forall c,$
$Permission(org, r, a, v_2, c) \wedge$
$sub\_view(org, v_1, v_2)$
$\rightarrow Permission(org, r, a, v_1, c)$

Une règle similaire, appelée VH_2, définit l'héritage des interdictions. Supposons par exemple que la vue *dossier chirurgical* est une sous-vue de la vue *dossier médical*. Dans ce cas, si un rôle a la permission ou l'interdiction de réaliser une certaine activité sur la vue *dossier médical*, alors ce rôle aura la permission ou l'interdiction de réaliser cette activité sur la vue *dossier chirurgical*.

Nous pouvons également définir le concept de vue *dérivée* comme un cas particulier de spécialisation de vue. Ainsi, nous considérons qu'une vue v_1 est dérivée d'une vue v_2 s'il existe une règle ayant la forme suivante :

– $\forall org, \forall obj, \forall v_1, \forall v_2,$
$(Use(org, obj, v_2) \wedge Condition) \rightarrow Use(org, obj, v_1)$
où $Condition$ est une condition logique utilisée pour spécialiser la vue v_2 en la vue v_1.

4 La hiérarchie d'organisations

Dans la section précédente, nous avons présenté comment définir dans une organisation des hiérarchies de rôles, d'activités et de vues. Nous étudions dans

cette section comment construire des hiérarchies d'organisations. Nous introduisons le prédicat $sub\_organization(org_1, org_2)$ qui signifie : l'organisation org_1 est une sous-organisation de l'organisation org_2. Nous supposons que ce prédicat définit une relation d'ordre partiel sur l'ensemble des organisations.

Ainsi, par exemple, si l'organisation H est un hôpital et $dept8$ son département des urgences, alors nous avons : $sub\_organization(dept8, H)$

Il est possible de spécifier que toutes les sous-organisations org_1 d'une certaine organisation org_2 sont affectées à un rôle donné. Cette spécification est modélisée par la contrainte suivante :

- C_7 : $\forall org_1, \forall org_2, \forall r,$
  $sub\_organization(org_1, org_2) \wedge \neg Empower(org_2, org_1, r)$
  $\rightarrow error()$

Dans l'exemple précédent, la contrainte C_7 est satisfaite si l'on a $Empower(H, dept8, casualty\_dept)$.

Remarquons qu'un rôle défini dans une certaine organisation n'est pas nécessairement défini dans toutes ses sous-organisations. Soit, par exemple, $dept7$ le département administratif de l'hôpital H et *infirmier* un rôle donné. Le rôle *infirmier* peut ne pas être défini dans le département $dept7$ (dans ce cas nous n'avons pas le fait $Relevant\_role(dept7, infirmier)$) alors que ce rôle est défini dans H $(Relevant\_role(H, infirmier))$.

Inversement, si org_1 est une sous-organisation de org_2, alors certains rôles peuvent être définis dans org_1 sans l'être dans org_2.

Les mêmes remarques sont valables dans le cas des vues et des activités. Ainsi, si org_1 est une sous-organisation de org_2, alors l'ensemble des vues, respectivement des activités, définies dans org_1 peut être disjoint de l'ensemble des vues, respectivement des activités, définies dans org_2.

4.1 L'héritage des hiérarchies

Supposons que org_1 est une sous-organisation de org_2. Le sous-ensemble des rôles définis dans org_2 qui sont également définis dans org_1, se voit appliquer dans $org1$ la même structure hiérarchique que celle définie dans org_2. La modélisation de cet état de faits est donnée par la règle suivante :

- HH_1 : $\forall org_1, \forall org_2, \forall r_1, \forall r_2,$
  $sub\_organization(org_2, org_1)$
  $sub\_role(org_1, r_1, r_2) \wedge$
  $Relevant\_role(org_2, r_1) \wedge Relevant\_role(org_2, r_2)$
  $\rightarrow sub\_role(org_2, r_1, r_2)$

Le même principe s'applique à l'héritage des privilèges dans la hiérarchie de spécialisation des rôles et également à l'héritage dans les hiérarchies de spécialisation des activités et des vues. Nous obtenons ainsi trois nouvelles règles, HH_2, HH_3 et HH_4, en remplaçant le prédicat sub_role dans la règle HH_1 respectivement par les prédicats $specialized\_role$, $sub\_activity$ et sub_view.

4.2 L'héritage des permissions et des interdictions

Nous appliquons les mêmes principes d'héritage des permissions et des interdictions à travers la hiérarchie d'organisations lorsque les rôles, les activités et les vues concernés dans les permissions et les interdictions sont pertinents pour la sous-organisation. Ce fait est modélisé par la règle suivante :

- OH_1 : $\forall org_1, \forall org_2, \forall r, \forall a, \forall v, \forall c,$
  $sub\_organization(org_2, org_1)$
  $Permission(org_1, r, a, v, c) \wedge$
  $Relevant\_role(org_2, r) \wedge$
  $Relevant\_activity(org_2, a) \wedge$
  $Relevant\_view(org_2, v)$
  $\rightarrow Permission(org_2, r, a, v, c)$

Une règle similaire, appelée OH_2, s'applique dans le cas de l'héritage des interdictions.

5 Spécification d'une politique de sécurité dans Or-BAC

5.1 Formalisation d'une politique de sécurité

En résumé, nous pouvons modéliser une politique de sécurité comportant des hiérarchies comme une théorie logique, conformément à la définition ci-après.
Définition 1 : Une politique de sécurité *pol* est modélisée dans Or-BAC comme une théorie logique T_{pol} définie de la façon suivante :

- Des ensembles de faits utilisant les prédicats *Relevant_role*, *Relevant_view* et *Relevant_activity*
- Des ensembles de faits utilisant les prédicats *Empower*, *Use*, *Consider*, *Permission* et *Prohibition*
- La règle *GE* et des faits utilisant le prédicat *G_Empower*
- Un ensemble de règles pour la définition des vues dérivées (section 3.3)
- Un ensemble de faits utilisant des prédicats binaires pour décrire les valeurs des attributs des sujets, des actions et des objets.
- Un ensemble des règles de définition de contextes, *i.e.* des règles dont la conclusion est le prédicat $Hold(org, s, \alpha, o, c)$
- Des ensembles de faits (hiérarchies d'héritage) utilisant les prédicats *sub_role*, *specialized_role*, *sub_activity* et *sub_view*
- Les règles RG_1 et RG_2 pour dériver des permissions et des interdictions concrètes (section 2.1)
- Les règles RH_1 à RH_4 (règles d'héritage de rôles), AH_1 et AH_2 (règles d'héritage des activités) et VH_1 et VH_2 (règles d'héritages des vues)
- Les règles HH_1 à HH_4 (règles d'héritage des hiérarchies)
- Les règles OH_1 et OH_2 (règles d'héritage des organisations)
- Un ensemble de contraintes, *i.e.* des règles dont la conclusion est le prédicat *error*().

Définition 2 : Une politique de sécurité *pol* viole une contrainte s'il est possible de dériver *error*() à partir de T_{pol} : $T_{pol} \vdash error()$

5.2 Les Conflits

Le modèle Or-BAC permet de spécifier des permissions ainsi que des interdictions. Cette possibilité offerte par le modèle peut malheureusement conduire à la génération de conflits. Une situation de conflits correspond à une situation où un sujet a à la fois la permission et l'interdiction d'effectuer une action sur un objet donné.

Signalons que les exceptions dans les règles d'héritage peuvent générer des conflits. Ainsi, dans l'exemple présenté dans la section 3.1, un chirurgien peut avoir à la fois l'interdiction de consulter le dossier médical d'un malade qui ne fait pas partie de ses patients (une interdiction héritée du rôle *médecin*) et la permission de le faire (cas où il existe une permission explicite affectée au rôle *chirurgien*).

Nous avons étudié ce type de problème dans [40]. La solution que nous proposons pour gérer de tels conflits consiste à définir des priorités entre les permissions et les interdictions. Dans l'exemple ci-dessus, l'interdiction héritée de *médecin* peut avoir une priorité plus faible qu'une permission explicite attribuée à un *chirurgien*. Ainsi, le chirurgien peut consulter le dossier médical d'un malade même si celui-ci ne fait pas partie de ses patients (ce qui peut arriver par exemple dans les cas d'urgences ou d'échanges de diagnostics). Cependant, comme les aspects relatifs à la gestion des conflits dans le modèle Or-BAC sortent du cadre du sujet traité dans cet article, nous ne les développons pas davantage, de plus amples informations pouvant être trouvées dans [40].

6 Application

Afin d'illustrer l'intérêt de la hiérarchisation des différentes entités définies dans le modèle Or-BAC étendu et son applicabilité à des cas concrêts qui ne se limitent pas au niveau organisationnel, nous présentons dans cette section un modèle d'un réseau local d'entreprise (RLE), de son architecture de sécurité et de sa connectivité à Internet. A cet effet, nous avons choisi de reprendre l'exemple utilisé dans Firmato [37] de façon à montrer comment les diverses entités et concepts utilisés dans l'architecture de sécurité de ce RLE sont naturellement exprimés avec le modèle Or-BAC. En particulier, nous montrons que le concept de hiérarchie du modèle Or-BAC étendu appliqué aux entités centrales que sont l'organisation, le rôle, l'activité et la vue permet d'éviter l'utilisation d'artifices tels que les groupes "open" et "closed" proposés dans [37].

6.1 Hiérarchie d'organisations

Nous souhaitons modéliser la politique de contrôle d'accès d'un réseau d'entreprise utilisé par une organisation H. H s'est dotée d'une configuration réseau basée sur deux pare-feux (firewalls) comme l'illustre la figure 4. Conformément à la présentation faite dans [37], le firewall externe est chargé de surveiller les connexions Internet de l'entreprise. Il est suivi par une DMZ (zone tampon).

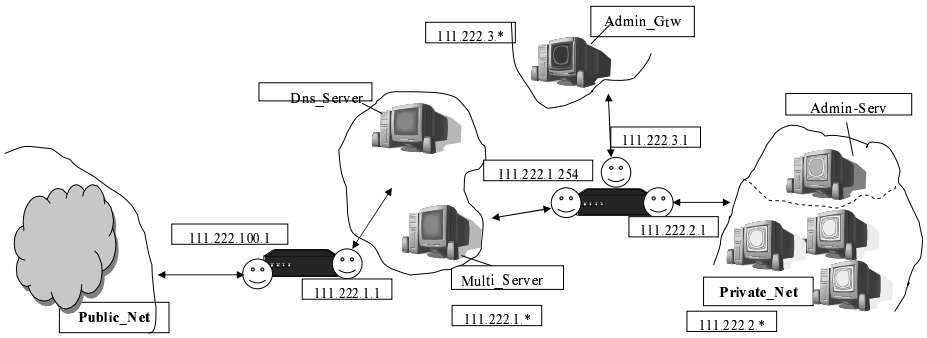


Fig. 4. Exemple d'application

La DMZ comporte les serveurs de l'entreprise visibles de l'extérieur. Dans notre cas, ces serveurs fournissent les services HTTP/HTTPS (Web), FTP, SMTP (e-mail) et DNS. L'entreprise utilise deux hôtes pour permettre l'utilisation de ces services, un premier hôte dédié au DNS et l'autre (appelé multi_server) pour disposer de tous les autres services restants. Quant au firewall interne, il contrôle les connexions intranet de l'entreprise. Ce firewall possède en fait trois interfaces : une pour la DMZ, une autre pour la zone privée de l'entreprise et la troisième pour séparer les hôtes utilisés pour l'administration des firewalls. Enfin, dans la zone privée de l'entreprise, un hôte spécifique, $Admin\_serv$, permet de gérer les serveurs de la DMZ.

Plusieurs organisations peuvent être introduites dans Or-BAC pour modéliser une telle architecture de sécurité réseau. D'abord, nous pouvons définir l'organisation H qui souhaite déployer une politique d'accès au RLE. Par souci de simplification, nous allons assimiler H à son réseau d'entreprise. H possède deux sous-organisations notées H_fw_1 et H_fw_2 correspondant respectivement au firewall externe et au firewall interne. Pour spécifier des politiques devant être mis en œuvre dans une zone particulière du RLE, d'autres organisations peuvent être définies. Ainsi, pour spécifier la politique en vigueur dans le réseau privé de H, l'organisation $H\_private\_net$ peut être introduite. Toutes les entités Or-BAC relatives à cette organisation peuvent alors être spécifiées. Si nous considérons, par exemple, que H est un hôpital, pour modéliser la politique de sécurité de cette partie du RLE on peut introduire des rôles tels que $medecin$, $infirmier$, etc. Cependant, pour simplifier, nous ne raffinons pas davantage cette partie. Signalons au passage qu'une organisation $internet$ peut être définie lorsque la politique de sécurité à mettre en œuvre par l'organisation Internet est explicitement spécifiée (ce qui n'est pas le cas pour le moment).

6.2 Les sujets

Dans cet exemple, les sujets correspondent aux hôtes identifiés par leurs adresses IP. Ainsi, si h est un hôte, le prédicat $address(h, a)$ signifie que l'adresse IP de h est a. Les rôles sont affectés aux hôtes conformément à la stratégie décrite

Nom du rôle	Hôtes affectés au rôle	sous_rôle	Pertinence pour H_fw_1	Pertinence pour H_fw_2
Public_host	Hôtes de la vue *Public_Net*	-	X	
Private_host	Hôtes de la vue *Private_Net*	-		X
Firewall	Interfaces des firewalls	*Ext_firewall* *Int_firewall*		X
Ext_firewall	Interfaces du firewall externe	-	X	X
Int_firewall	Interfaces du firewall interne	-		X
DNS_server	Serveur DNS	-	X	X
Ftp_server	Serveur ftp	*Multi_server*	X	X
Mail_server	Serveur de mails	*Multi_server*	X	X
Web_server	Serveur web	*Multi_server*	X	X
Multi_server	Multi-serveur	-	X	X
Adm_fw_host	Hôtes de la vue *Admin_gtw*	-	X	X
Adm_serv_host	Hôtes de la vue *Admin_serv*	-		X

Fig. 5. Description d'un rôle

dans la section 6.3. Pour ce faire, le prédicat *Empower* permet d'attribuer un rôle à un hôte donné. Toutefois, il est parfois plus simple de former des groupes d'hôtes (appelés *zone* dans Firmato) et d'utiliser le prédicat $G\_Empower$ pour affecter le même rôle à chacun des hôtes appartenant à un groupe. Ainsi, on peut par exemple définir le groupe $Private\_net$ de la façon suivante :

- $\forall h, Use(H, h, Private\_net) \leftarrow$
  $Use(H, h, Host) \wedge address(h, a) \wedge a \in 111.222.2.*$
  $\wedge \neg Use(H, h, Firewall\_interface)$

6.3 Les rôles

Les rôles décrits dans la figure 5 sont affectés aux hôtes. Tous ces rôles sont pertinents (relevants) pour l'organisation H. La figure 5 spécifie les rôles pertinents respectivement aux organisations H_fw_1 et H_fw_2. Signalons que le rôle *Firewall* est pertinent pour H_fw_2 (pour des besoins d'administration) mais ne l'est pas pour H_fw_1. La figure 5 décrit également pour chacun des rôles les sous-rôles correspondants. Dans l'exemple, la hiérarchie de sous-rôles correspond à celle d'une spécialisation des rôles.

6.4 Les activités

Les activités correspondent aux divers services offerts par le RLE de H. Nous définissons d'abord l'activité *all_tcp* à laquelle sont associées différentes sous-activités tcp telles que *smtp*, *ssh* et *https*. Nous définissons de façon similaire l'activité *all_icmp* et les différentes sous-activités sous-jacentes telles que *ping*. Dans le cadre de cet exemple, deux autres activités sont définies. L'activité *admin_to_gtwy* possède deux sous-activités, *ssh* et *ping*. La deuxième activité *gtwy_to_admin* possèdent également deux sous-activités *ssh* et *https*.

L'ensemble de toutes ces activités est pertinent pour les organisations H, H_fw_1 et H_fw_2

La principale différence entre l'approche que nous exposons ici et celle proposée dans [37] est l'utilisation des hiérarchies d'activités alors que Firmato a recours aux services élémentaires et aux groupes de services. Elle est également plus générique dans la mesure où les permissions et les interdictions s'appliquent à la seule entité activité.

6.5 Les vues

Les vues sont utilisées pour structurer les objets auxquels s'appliquent les différents services réseau. Ainsi, nous définissons la vue *target* ayant deux attributs : *content* correspondant aux messages transmis lors de l'utilisation du service et *dest* correspondant au hôte destinataire du service. Cet hôte est identifié par son rôle.

Dans l'exemple traité dans cette section, l'attribut *content* n'est pas utilisé parce que, pour les besoins d'illustration, nous ne considérons que des règles de filtrage sur l'hôte destinataire. Cependant, notre modèle peut aussi être facilement étendu pour spécifier des règles de filtrages sur les contenus des messages.

Nous pouvons ensuite introduire des sous-vues de la vue *target* conformément au rôle affecté à l'hôte destinataire. Nous pouvons, par exemple, définir des sous-vues *to_dns* de la façon suivante :

– $\forall o, Use(H, o, to\_dns) \leftarrow Use(H, o, target) \wedge dest(o, dns)$

Il en résulte la nécessité de définir autant de vues que de rôles. Pour assouplir ce processus, nous suggérons de définir une fonction *to_target* ayant pour domaine les rôles et pour co-domaine les vues. Les vues créées par cette fonction sont définies de la façon suivante :

– $\forall o, \forall r, Use(H, o, to\_target(r)) \leftarrow Use(H, o, target) \wedge dest(o, r)$

Nous considérons que la vue $to\_target(r)$ est pertinente pour l'une des deux organisations définies dans notre exemple si r est un rôle pertinent pour cette organisation en question. Nous considérons également que si le rôle r_1 est un sous-rôle du rôle r_2, alors la vue $to\_target(r_1)$ est une sous-vue de la vue $to\_target(r_2)$.

6.6 La politique de sécurité

Les éléments nécessaires à la spécification de la politique de sécurité ayant été introduits, nous pouvons donc définir les différentes permissions qui doivent être mises en application dans l'organisation H. Les objectifs de cette politique de sécurité sont les suivants :

1. Les hôtes de l'organisation interne peuvent accéder à l'Internet.
2. Les hôtes externes ne peuvent accéder qu'aux serveurs de la DMZ.
3. Les hôtes ayant le rôle *Admin_serv_host* peuvent mettre à jour les serveurs de la DMZ. Les autres hôtes de l'organisation ont les mêmes privilèges que les hôtes externes sur les serveurs de la DMZ.

4. Les interfaces des firewalls ne sont accessibles que par les hôtes jouant le rôle *Adm_fw_host*.

Ces permissions correspondent à la politique de sécurité présentée dans [37]. La figure 6 décrit comment ces permissions sont modélisées dans Or-BAC. Dans un souci de simplification, nous ne spécifions pas d'interdictions et nous supposons que toutes les permissions sont valables dans n'importe quel contexte (le contexte considéré est donc $default$ et est toujours évalué à $true$).

Un avantage significatif de notre approche par rapport à d'autres procédés comme celui de Firmato par exemple, est le fait qu'elle permette de dériver automatiquement des permissions qui s'appliquent respectivement à H_fw_1 et H_fw_2. En effet, il suffit d'appliquer la règle OH_1 pour dériver les permissions relatives aux sous-organisations de H. Les résultats que nous avons obtenus pour H_fw_1 sont décrits dans la figure 7. Pour illustrer le processus de dérivation, considérons la permission suivante :

– $Permission(H, adm\_fw\_host, admin\_to\_gtwy, to\_target(firewall), default)$

Etant donné que le rôle adm_fw_host, l'activité $admin\_to\_gtwy$ et la vue $to\_target(firewall)$ sont pertinents pour la sous-organisation H_fw_2, nous pouvons appliquer OH_1 pour dériver :

– $Permission(H\_fw_2, adm\_fw\_host, admin\_to\_gtwy,$
$to\_target(firewall), default)$

Cependant, puisque la vue $to\_target(firewall)$ n'est pas pertinente pour H_fw_1, nous ne pouvons pas dériver une permission équivalente pour H_fw_1. Mais la vue $to\_target(ext\_firewall)$ est une sous-vue de *to_ target(firewall)* et puisque $to\_target(ext\_firewall)$ est une vue pertinente pour H_fw_1, nous pouvons appliquer les règles VH_1 et OH_1 pour dériver :

– $Permission(H\_fw_1, adm\_fw\_host, admin\_to\_gtwy,$
$to\_target(ext\_firewall), default)$

Signalons que la permission :

– $Permission(H, private\_host, all\_tcp, to\_target(public\_host), default)$

n'est héritée par aucune des deux sous-organisations H_fw_1 et H_fw_2. C'est une conséquence du fait que le rôle $private\_host$ n'est pertinent que pour H_fw_2 alors que la vue $target(public\_host)$ est pertinente uniquement pour H_fw_1. Par conséquent, un seul des deux firewalls ne suffit pas pour gérer cette permission. Dans ce cas, nous proposons d'utiliser cette permission pour configurer les deux firewalls.

Comparativement à Firmato, l'approche que nous proposons est basée sur un ensemble plus réduit de concepts. En particulier, nous n'avons pas besoin d'introduire des notions comme celle des groupes "closed" (fermés). Un groupe fermé n'hérite pas des groupes de niveaux supérieurs de la hiérarchie. L'exemple pris dans Firmato est celui du groupe *firewall* qui ne doit pas hérité de *private_ host*. Nous pensons que cette notion de groupe fermé nécessite une gestion quelque peu complexe et en fait inutile. Dans notre approche, il suffit simplement de spécifier que *private-net* ne comprend pas l'interface *firewall-interface* (se référer à la définition de *private-net* présentée dans la section 6.2). De plus, notre approche peut être utilisée pour traiter des applications plus complexes comprenant des

$Permission(H, adm\_fw\_host, admin\_to\_gtwy, to\_target(firewall), default)$
$Permission(H, firewall, gtwy\_to\_admin, to\_target(adm\_fw\_host), default)$
$Permission(H, private\_host, all\_tcp, to\_target(public\_host), default)$
$Permission(H, adm\_server\_host, all\_tcp, to\_target(dns\_server), default)$
$Permission(H, adm\_server\_host, all\_tcp, to\_target(multi\_server), default)$
$Permission(H, public\_host, smtp, to\_target(mail\_server), default)$
$Permission(H, public\_host, dns, to\_target(dns\_server), default)$
$Permission(H, public\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H, public\_host, https, to\_target(web\_server), default)$
$Permission(H, private\_host, smtp, to\_target(mail\_server), default)$
$Permission(H, private\_host, dns, to\_target(dns\_server), default)$
$Permission(H, private\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H, private\_host, https, to\_target(web\_server), default)$
$Permission(H, dns\_server, dns, to\_target(public\_host), default)$
$Permission(H, ftp\_server, ftp, to\_target(public\_host), default)$
$Permission(H, dns\_server, dns, to\_target(private\_host), default)$
$Permission(H, ftp\_server, ftp, to\_target(private\_host), default)$

Fig. 6. Les permissions dans l'organisation H

$Permission(H\_fw_1, adm\_fw\_host, admin\_to\_gtwy, to\_target(ext\_firewall), default)$
$Permission(H\_fw_1, ext\_firewall, gtwy\_to\_admin, to\_target(adm\_fw\_host), default)$
$Permission(H\_fw_1, public\_host, smtp, to\_target(mail\_server), default)$
$Permission(H\_fw_1, public\_host, dns, to\_target(dns\_server), default)$
$Permission(H\_fw_1, public\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H\_fw_1, public\_host, https, to\_target(web\_server), default)$
$Permission(H\_fw_1, dns\_server, dns, to\_target(public\_host), default)$
$Permission(H\_fw_1, ftp\_server, ftp, to\_target(public\_host), default)$

Fig. 7. Les permissions dans l'organisation H_fw_1

interdictions, des exigences sur les contenus des messages ou encore des règles contextuelles.

7 Conclusion

Dans cet article, nous avons montré comment modéliser des hiérarchies d'héritage dans le modèle Or-BAC. Les travaux précédents considéraient seulement des hiérarchies de rôles (comme c'est la cas dans le modèle RBAC). Ici, nous avons défini des hiérarchies de rôles, d'activités, de vues et d'organisations et étudié l'héritage des permissions et des interdictions dans ces hiérarchies.

Concernant la hiérarchie de rôles, nous avons montré qu'il est important d'introduire deux types de hiérarchies : la hiérarchie de type spécialisation/généralisation et la hiérarchie de type junior/senior. Les permissions sont héritées "vers le bas" dans les deux hiérarchies (le rôle le plus spécialisé héritant du rôle le moins spécialisé et le rôle senior héritant du rôle junior). En revanche, nous considérons que les interdictions sont héritées "vers le bas" dans les hiérarchies de type spécialisation/généralisation (comme pour les permissions) alors qu'elles

sont héritées "vers le haut" dans les hiérarchies de type junior/sénior (le rôle junior héritant les interdictions du rôle senior). Les propositions précédentes ne font pas ce type de distinction. Par exemple, [29] considère que les interdictions sont toujours héritées "vers le haut" dans la hiérarchie de rôles. Notre proposition permet d'éliminer certaines ambiguïtés présentes dans les approches précédentes.

Concernant la hiérarchie d'activités, nous avons seulement considéré la spécialisation/généralisation. On peut en fait considérer d'autres types de "décomposition" d'activités, par exemple décomposer une activité a en $b; c$ représentant l'activité b suivie par l'activité c. Il est clair que si un certain rôle r a la permission de réaliser l'activité a, alors r doit également avoir la permission de réaliser l'activité b. Cependant, il est plus complexe de définir les permissions concernant l'activité c; le rôle r ne devrait avoir la permission de réaliser l'activité c qu'après avoir réalisé l'activité b. Dans Or-BAC, il semble qu'il soit possible de représenter cette contrainte en utilisant la notion de contexte. Modéliser ce type de décomposition dans Or-BAC est un problème intéressant qui a plusieurs applications, en particulier la spécification de politiques de sécurité pour les systèmes de "workflow" [41]. Nous comptons étudier ce problème dans le futur.

Nous avons également proposé une hiérarchie de vues de type spécialisation/généralisation. Nous envisageons d'étudier d'autres associations entre les vues, en particulier l'association d'agrégation (également appelée association "$Part\_of$"). On pourrait ainsi définir comment les permissions et les interdictions se propagent depuis une vue vers ses différentes sous parties. Il y a plusieurs applications à un tel modèle, par exemple la modélisation UML ou la protection de documents XML. Cependant, plusieurs problèmes surgissent, en particulier certaines activités peuvent être pertinentes pour une certaine vue mais ne pas s'appliquer à certaines de ses sous parties. Nous comptons également approfondir ce problème dans le futur.

Enfin, nous avons défini une hiérarchie d'organisations et modéliser l'héritage des permissions et des interdictions dans cette hiérarchie. Nous avons montré que cette hiérarchie est utile pour dériver, de la spécification de politiques de sécurité organisationnelle, les politiques de sécurité technique de composants particuliers tels qu'un firewall. Nous avons implanté un module de traduction qui produit les règles de sécurité pour configurer le firewall NetFilter. Nous envisageons d'appliquer une approche similaire pour produire les règles de configuration d'autres composants tels que des systèmes d'exploitation ou des systèmes de gestion de bases de données. Nous comptons également appliquer Or-BAC pour modéliser les exigences d'interopérabilité entre ces différentes politiques de sécurité.

Remerciements : Les travaux présentés dans cet article sont réalisés dans le cadre du projet RNRT exploratoire MP6 et de l'ACI DESIRS tout deux financés par le Ministère de la Recherche. La participation d'Alexandre Miège à ces travaux se fait dans le cadre d'une thèse financée par France Télécom R&D.

Références

1. B. Lampson, Protection, *5th Princeton Symposium on Information Sciences and Systems, pp. 437-443, mars 1971.*
2. *D. E. Bell et L. J. LaPadula, Secure Computer Systems : Unified Exposition and Multics Interpretation,* The MITRE Corporation, *ESD-TR-73-306, mars 1976.*
3. *K. J. Biba, Integrity Consideration for Secure Computer Systems,* The MITRE Corporation, *MTR-3153, juin 1975.*
4. *R. Sandhu, E. J. Coyne et H. L. Feinstein et C. E. Youman, Role-Based Access Control Models,* IEEE Computer, *Vol. 29, numéro 2, pp. 38–47, 1996.*
5. *S. I. Gavrila et J. F. Barkley, Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management,* Third ACM Workshop on Role-Based Access Control, *pp. 81–90, octobre 1996.*
6. *D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn et R. Chandramouli, Proposed NIST Standard for Role-Based Access Control,* ACM Transactions on Information and System Security, *volume 4 numéro 3, pp. 222–274, 2001.*
7. *N. Damianou, N. Dulay, E. Lupu et M. Sloman, The Ponder Policy Specification Language,* International Workshop, Policies for Distributed Systems and Neworks (Policy 2001), *Bristol, UK, janvier 2001.*
8. *M. A. Harrison, W. L. Ruzzo et J. D. Ullman, Protection in Operating Systems,* Communication of the ACM, *Vol. 19 numéro 8, pp. 461–471, août 1976.*
9. *J. Barkley, K. Beznosoz et J. Uppal, Supporting Relationships in Access Control Using Role Based Access Control,* Proceeding for the ACM workshop on RBAC, *Fairfax, Virginia, USA, octobre 1999.*
10. *E. C. Cheng, An Object-Oriented Organizational Model to Support Dynamic Role-based Access Control in Electronic Commerce Applications,* 32nd Annual Hawaii International Conference on System Sciences (HICSS-32), *Maui, Hawaii, janvier 1999.*
11. *Roshan K. Thomas, TMAC : A primitive for Applying RBAC in collaborative environment,* 2nd ACM, Workshop on RBAC, *FairFax, Virginia, USA, pp. 13–19, novembre 1997.*
12. *C. K. Georgiadis, L. Mavridis, G. Pangalos et R. K. Thomas, Flexible Team-Based Access Control Using Contexts,* ACM Symposium on Access Control Models and Technologies, (SACMAT'01), *mai 2001.*
13. *E. Bertino, F. Buccafurri, E. Ferrari et P. Rullo, A Logic-based approach for enforcing access control,* Journal of Computer Security, *Vol. 8, numéro 2,3, pp. 109–139, 2000*
14. *M. Willikens, S. Feriti, A. Sanna et M. Masera, A Context-related autorisation and access control method based on RBAC : A case study from the Health Care Domain, Seventh ACM Symposium on Access Control Models and Technologies, (SACMAT'02),* Monterey, California, USA, pp. 117–124, juin 2002.
15. L. Zhang, G. Ahn et B. Chu, A Role-Based Delegation Framework for HealthCare Information Systems, *Seventh ACM Symposium on Access Control Models and Technologies, (SACMAT'02),* Monterey, California, USA, pp. 125–134, juin 2002.
16. R. Thomas et R. Sandhu, Task-based Authorization Controls (TBAC) : A Family of Models for Active and Enterprise-oriented Authorization Management, *11 th*

IFIP Working Conference on Database Security, août 1997, Lake Tahoe, California, USA.

17. C. Bettini, S. Jajodia, X. S. Wang et D. Wijesekera, Obligation Monitoring in Policy Management, *International Workshop, Policies for Distributed Systems and Neworks (Policy 2002)*, juin 2002, Monterey CA.
18. R. Viviani, A Type / Domain Security Policy for Internet Transmission Sharing, and Archiving of Medical and Biological data, *International Workshop, Policies for Distributed Systems and Neworks (Policy 2001)*, Bristol, UK, pp. 29–31, juin 2001.
19. E. Bertino, S. Jajodia et P. Samarati, Supporting Multiple Access Control Policies in Database Systems, *IEEE Symposium on Security and Privacy 1996*, Oakland, USA.
20. G. Dinolt, L. Benzinger et M. Yatabe, Combining Components and Policies, *Proc. of the Computer Security Foundations Workshop VII*, Franconia, USA, 1994.
21. F. Cuppens, L. Cholvy, C. Saurel et J. Carrère, Merging Regulations : analysis of a practical example, *International Journal of Intelligent Systems*, Vol. 16, numéro 11, novembre 2001.
22. S. Benferhat, R. El Baida et F. Cuppens, Modélisation des politiques de sécurité dans le cadre de la théorie des possibilités, *Rencontres Francophones de la Logique Floue et ses Applications*, Montpellier, France, octobre 2002.
23. Ravi Sandhu, Bhamidipati et Qamar Munawer, The ARBAC97 Model for Role-Based Administration of Roles, *ACM Transactions on Information and System Security*, Vol. 2, numéro 1, février 1999.
24. A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel et G. Trouessin, Organization Based Access Control, *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Come, Italy, juin 2003.
25. S. Benferhat, R. El Baida et F. Cuppens, A Stratification-Based Approach for Handling Conflicts in Access Control, *8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*, Lake Come, Italy, juin 2003.
26. F. Cuppens et A. Miège, Administration Model for Or-BAC, *International Federated Conferences (OTM'03), Workshop on Metadata for Security*, Catania, Sicily, Italy, novembre 2003.
27. F. Cuppens, L. Cholvy, C. Saurel et J. Carrère, Merging regulations : analysis of a practical example, *International Journal of Intelligent Systems*, Vol. 16 numéro 11, novembre 2001.
28. L. Cholvy et F. Cuppens, Analyzing Consistency of Security Policies, *IEEE Symposium on Security and Privacy*, Oakland, CA, mai 1997.
29. E. Bertino, B. Catania, E. Ferrari et P. Perlasca, A Logical Framework for Reasoning about Access Control Models, *ACM Transactions on Information and System Security*, Vol. 6 numéro 1, février 2003.
30. S. Jajodia, S. Samarati et V. S. Subrahmanian, A logical Language for Expressing Authorizations, *IEEE Symposium on Security and Privacy*, Oakland, CA, mai 1997.
31. E. Bertino, S. Jajodia et P. Samarati, Supporting Multiple Access Control Policies in Database Systems, *IEEE Symposium on Security and Privacy*, Oakland, 1996.

32. S. Jajodia, P. Samarati, M.L. Sapino et V.S. Subrahmanian, Flexible Support for Multiple Access Control Policies, *ACM Transactions on Database Systems*, Vol. 26, numéro 2, juin 2001.

33. Y. Bai et V. Varadharajan, A Logical Based Approach for the Transformation of Authorization Policies, *Proc. of the 10th Computer Security Foundations Workshop*, Rockport, Massachusetts, 1997.

34. G.-J. Ahn et R. Sandhu", Role-Based Authorization Constraints Specification, *ACM Transactions on Information and System Security*, Vol. 3 numéro 4, novembre 2000.

35. J. D. Moffett, Control Principles and Role Hierarchies, *3rd ACM Workshop on Role-Based Access Control*, octobre 1998.

36. J. D. Moffett et E. C. Lupu, The use of role hierarchies in access control, *4th ACM Workshop on Role-Based Access Control* octobre 1999.

37. Y. Bartal, A. Mayer, K. Nissim et A. Wool, Firmato : A novel firewall management toolkit, *20th IEEE Symposium on Security and Privacy*, Oakland, California, mai 1999.

38. Sejong Oh et Ravi Sandhu, A Model for Role Administration Using Organization Structure, *Seventh ACM Symposium on Access Control Models and Technologies, (SACMAT'02)*, Monterey, California, USA, pp. 155–162, juin 2002.

39. F. Cuppens et A. Miège, Modelling contexts in the Or-BAC model, *19th Annual Computer Security Applications Conference*, Las Vegas, décembre 2003.

40. F. Cuppens et A. Miège, Conflict management in the Or-BAC model, ENST Bretagne, décembre 2003.

41. Elisa Bertino, Elena Ferrari et Vijay Atluri, Specification and enforcement of authorization constraints in workflow management systems, *ACM Transactions on Information and System Security*, Vol. 2 numéro 1, février 1999.

42. J. Crampton, On permissions, inheritance and role hierarchies, *10th ACM Conference on Computer and Communication Security*, Washington, novembre 2003.

Une démarche méthodologique pour l'anonymisation de données personnelles sensibles

Anas Abou El Kalam[1], Yves Deswarte[1], Gilles Trouessin[2], and Emmanuel Cordonnier[3]

[1] LAAS-CNRS
7 avenue du colonel Roche
31077 Toulouse Cedex 4
{Deswarte, anas}@laas.fr
[2] ERNST & YOUNG
1 place Alfonse Jourdain
31000 Toulouse
gilles.trouessin@fr.ey.com
[3] ETIAM
20 Rue du Pr Jean Pecker
35000 Rennes
emmanuel.cordonnier@etiam.com

Résumé Cet article présente une nouvelle technique d'anonymisation destinée aux applications où les données personnelles doivent être cachées. Le problème de la protection de la vie privée ainsi que la base terminologique nécessaire à la compréhension de cet article sont d'abord présentés. Une démarche rigoureuse d'analyse des besoins et de choix de solutions est ensuite proposée. En particulier, seront expliqués les différences et les liens entre les besoins, les objectifs ainsi que les exigences d'anonymisation. Une caractérisation des solutions à choisir, à construire ou à mettre en ?uvre sera également proposée. L'analyse nous conduira à montrer l'intérêt majeur de l'utilisation des cartes à puces pour satisfaire les besoins de protection de la vie privée; en particulier pour garder le secret de l'identifiant (ou des variables identifiantes en général) de l'utilisateur et pour exécuter la partie critique de procédure d'anonymisation. Ainsi, en fournissant sa carte, le citoyen (e.g., le patient dans le domaine médical) donne son consentement pour exploiter ses données anonymisées; et pour chaque utilisation, un nouvel identifiant anonyme est générée à l'intérieur de la carte. Par ailleurs, cet article montre comment peut-on lever l'anonymat en respectant certains principes de bases, notamment le consentement du patient (en fournissant sa carte). Cet article montre que la technologie des cartes à puces peut jouer un rôle important pour la protection de la vie privée, ne serait ce que pour stocker un secret (un identifiant anonyme généré localement et aléatoirement à l'intérieur de la carte), et pour garder les procédures d'anonymisation (et donc l'utilisation des données anonymes), et de désanonymisation sous le contrôle permanent de l'utilisateur.

1 Problématique

Les applications informatiques émergentes utilisent des données dont le contenu est souvent sensible : réseaux de soins, recensements démographiques, commerce électronique, vote électronique, etc. D'une part, l'instauration des réseaux facilite le partage et la communication d'informations entre différentes structures ; d'autres part, elle pose des problèmes concrets de respect de la vie privée.

Conscients de ce problème, les législateurs imposent des exigences de confidentialité et de sécurité sur les données personnelles, en particulier par les réglementations internationales [1], européennes [2,3,4] et nationales [5,6]. Pour respecter cette législation, les systèmes (utilisant des données personnelles) ont besoin de mettre en ?uvre des méthodes et des moyens efficaces pour fournir (et justifier) un niveau élevé de sécurité. Mais malheureusement, la protection de la vie privée est souvent négligée ou mal étudiée. En l'occurrence, l'étude analytique des exigences d'anonymisation nous semble peu présente dans la majorité des solutions d'anonymisation actuelles ; plutôt que de se baser sur une méthodologie systématique, ces solutions sont souvent développées empiriquement.

Dans un souci de mener une réflexion bien fondée autour de ce problème plus que jamais d'actualité et d'y apporter des solutions réelles, cet article commence par définir le glossaire des termes élémentaires les plus couramment utilisés, associé au thème "anonymisation" (section suivante). Il propose ensuite une approche méthodologique pour préserver la vie privée, notamment en matière d'anonymisation des identifiants de personnes physiques figurant dans des fichiers informatisés (troisième section).

À cet égard, la démarche progressive présentée est essentiellement fondée sur l'identification des besoins, des objectifs et des exigences de sécurité, avant de définir ou de choisir la solution la plus adaptée à chaque problème lié au respect de la vie privée. Compte tenu des besoins de sécurité de l'application traitée, et en prenant en compte le contexte et la finalité des traitements que subiront les informations, nous identifions les premières questions cruciales à se poser :

- a-t-on besoin de chaînabilité, observabilité, anomynisation ou pseudonymisation ?
- Quel type de mécanisme (anonymisation irréversible ou inversible, appauvrissement de données, brouillage...) est le plus approprié ?
- Quelle forme de chaînage utiliser ?
- Quelle robustesse doit-on avoir et vis-à-vis de quoi ?
- ....

Afin de montrer l'intérêt pratique de notre méthodologie et de faire apparaître toutes ses facettes, nous allons l'appliquer dans un domaine où les données personnelles traitées sont d'une grande sensibilité. La troisième section présente ainsi un ensemble de scénarios représentatifs et en analyse les risques, les attaques, les besoins, les objectifs ainsi que les exigences d'anonymisations.

La dernière section tient compte des besoins déjà identifiés, pour présenter une nouvelle solution générique de génération et de gestion de données personnelles anonymisées. Cette solution montre, selon le cas, les transformations à appliquer aux données personnelles depuis leur collecte (au niveau des hôpi-

taux par exemple), en passant par les centres de traitement (les associations de personnes diabétiques ou les centres des études cliniques, par exemple) jusqu'aux utilisateurs finaux (recherche scientifique, publications, Web, presse, par exemple). Dans un souci de respecter les législations, en particulier les recommandations de la norme européenne [7], le traitement choisi dans notre solution (chiffrement, anonymisation, filtrage de données, ...) tient compte du rôle et de l'établissement de rattachement de l'utilisateur, ainsi que de la finalité de l'utilisation prétendue.

Outre la méthodologie et la procédure globale d'anonymisation présentées dans cet article, une originalité fondamentale de la solution que nous proposons réside dans l'utilisation de la technologie des cartes à puces. Dans l'état actuel des connaissances, une carte à puce constitue un moyen matériel fiable et très difficilement falsifiable[4]. Ainsi, nous suggérons que la partie la plus critique de la procédure d'anonymisation soit exécutée au niveau d'une carte à puce (la carte VITALE, par exemple) appartenant à l'utilisateur (le patient, dans ce cas). Cette procédure est basée sur un secret, l'identifiant anonyme unique et individuel de l'utilisateur, généré aléatoirement (au sein de la carte), détenu par l'utilisateur (sur sa carte) et qui n'est jamais transmis à l'extérieur de la carte. Le secret reste donc sous le contrôle de l'utilisateur. Sauf en cas d'obligation légale, les données personnelles ne peuvent figurer dans une certaine base de données (pour une étude épidémiologique, par exemple) que si l'utilisateur donne son consentement en fournissant sa carte. La transformation cryptographique d'anonymisation s'effectue au sein de la carte ; par conséquent, l'identifiant initial de l'utilisateur n'est jamais connu en dehors de la carte.

En outre, afin de trouver un juste milieu entre l'anonymat et le lever d'anonymat, la désanonymisation doit également être sous le contrôle de l'utilisateur. En ce lieu, le rôle que joue la carte à puce, détenue par l'utilisateur, est incontestable : la désanonymisation n'est possible que si l'utilisateur donne son consentement en fournissant sa carte pour cette opération (la désanonymisation).

La fin de l'article présente une discussion qui montre que la solution proposée assure la sécurité (robustesse aux attaques par dictionnaires, par exemple) sans compromettre la flexibilité (supporter certains changements organisationnels comme le fusionnement de plusieurs établissements).

Après avoir exposé la problématique, et avant de développer davantage notre contribution, nous tenons à préciser les points suivants :

- même si l'étude de cas concerne le domaine santé-social, les préoccupations liées à l'anonymisation ne sont ni dédiées ni spécifiques à ce secteur ; la méthodologie ainsi que les solutions proposées restent applicables à une large gamme d'applications telles que celles citées au début de cet article ;

[4] La falsification demeure très peu probable compte tenu des moyens à mettre en uvre pour réussir lattaque et des résultats obtenus même après lintrusion. La carte peut éventuellement être dotée dun petit programme enregistrant les tentatives daccès illicites à la carte.

– même si nous abordons d'autres moyens (dispositifs) techniques et organisationnelles (comme le contrôle d'accès ou la détection d'intrusion) pouvant compléter notre solution, nous n'allons pas les détailler davantage ; ces dispositifs restent orthogonaux, et en les citant, notre but est tout simplement de montrer un cadre global ou notre proposition peut être intégrée.

2 Définitions

Notre analyse du domaine de la protection de la vie privée nous a permis de constater qu'il existe plusieurs nuances entre les termes utilisés par les différentes communautés scientifiques. À cet égard, citons par exemple la différence entre les législations européennes [2,3,4] qui visent à protéger les données personnelles et les lois Françaises [5,6] qui concernent les données nominatives. Dans notre vision, une donnée peut être "non-nominative" tout en restant "personnelle". En effet, comme il est parfois possible de ré-identifier des données anonymisées, ces données (anonymisées) peuvent perdre ou non leur caractère anonyme ; de même que des données nominatives peuvent perdre leur caractère nominatif. Dans tous les cas, nominatives, anonymes ou anonymisées, ces données restent à caractère personnel. Les anglo-saxons parlent d'ailleurs de "de-identification" au lieu d'anonymisation.

Afin d'éviter toute confusion et de faciliter la compréhension du reste de cet article, commençons tout d'abord par définir la base terminologique associé au thème de l'anonymisation des identifiants de personnes physiques figurant dans les fichiers informatisés.

Un identifiant d'une personne physique peut être défini comme une étiquette de nommage associée grâce à un système ou par une procédure d'identification à toute personne figurant dans une population donnée. Les deux propriétés essentielles à garantir sont l'atomicité ou la fiabilité :

– l'atomicité de l'identifiant est sa capacité à conserver ou à perdre la granularité élémentaire de l'information associée à une personne physique ;
– la fiabilité de l'identifiant est sa capacité à protéger de toutes formes possibles d'ambiguïtés inhérentes au système d'informations ;

Notons que même avec un identifiant atomique, il peut y avoir des doublons d'identifiant pour un même individu (comme des synonymes) ou à, l'inverse, des collisions d'identifiants d'individus distincts (comme des homonymes) : c'est la qualité de l'information amont à l'anonymisation qui présagera en grande majorité la qualité, et donc la fiabilité, du système d'anonymisation.

Par ailleurs, un identifiant peut revêtir deux formes : nominatif ou anonyme. L'identifiant est qualifié de nominatif s'il permet de déterminer sans ambiguïté la personne concernée ; dans le cas contraire, il est considéré comme anonyme.

Les critères communs (en anglais " *Common Critéria for Information Security Evaluation* ") qui sont maintenant une norme internationale [8] détaillent plusieurs classes fonctionnelles liées à la confidentialité, notamment l'anonymat, la pseudonymat, la chaînabilité et l'observabilité.

- L'anonymat peut être définie comme la propriété garantissant qu'un utilisateur peut utiliser une ressource ou un service sans révéler son identité ; autrement dit, l'impossibilité (pour d'autres utilisateurs) de déterminer le véritable nom de cet utilisateur.
- Le pseudonymat ajoute à l'anonymat le fait que l'utilisateur peut être tenu responsable de ses actes ; par exemple, en cas de litige ou d'enquĺte (lutte contre le blanchiment d'argent sale, par exemple), la propriété requise est la pseudonymat (plutôt que l'anonymat) car certaines informations personnelles doivent pouvoir être fournies aux autorités judiciaires.
- La non-chaînabilité représente l'impossibilité (pour d'autres utilisateurs) d'établir un lien entre différentes opérations réalisées par un même utilisateur ; par exemple, en interdisant la fusion ou le croisement d'informations à partir de différents fichiers ou bases de données.
- La non-observabilité garantit qu'un utilisateur peut utiliser une ressource ou un service sans que d'autres utilisateurs soient capables de déterminer si une opération (l'utilisation d'une ressource ou d'un service) est en cours.

3 Démarche d'analyse

L'analyse des solutions d'anonymisation utilisées dans les systèmes de santé des pays européens [9] nous a permis de détecter certaines défaillances, souvent liées à une absence d'une démarche analytique préalable des risques, des besoins, des exigences ainsi que des objectifs de sécurité. Comme pour beaucoup de solutions d'ordre sécurité, nous pensons que l'anonymisation recouvre deux grandes catégories de concepts :

- la demande sous forme de besoins d'anonymisation à satisfaire ;
- la réponse sous forme de fonctionnalités et solutions pour anonymiser.

Une fonctionnalité d'anonymisation peut être exprimée selon un des trois niveaux d'attente suivants classés par ordre croissant de force :

- le besoin d'anonymisation, représente les attentes de l'utilisateur ; généralement sous une forme qui n'est pas toujours très bien explicité ni aisé à formaliser ;
- l'objectif d'anonymisation, spécifie le niveau de sécurité à atteindre ou les menaces à éviter (comment satisfaire les exigences ?) ;
- l'exigence d'anonymisation, représente la façon d'exprimer le besoin ; dans la mesure du possible, très proche d'un formalisme clair et d'une sémantique non-ambiguë.

3.1 Besoins d'anonymisation

Les besoins de protection de la vie privée peuvent être d'ordre général, comme ceux identifiés dans la première section (problématique), mais aussi et surtout spécifiques au système étudié. Dans les systèmes de santé par exemple, une liste non exhaustive des besoins d'anonymisation pourrait être :

- outre le nom, le prénom et le numéro de sécurité social, les données les plus sensibles sont la date de naissance (parfois, seulement l'année de naissance est nécessaire), l'adresse (parfois, seulement la région est intéressante à connaître), la nationalité (il est probablement plus judicieux dans certains cas d'effectuer des regroupements)....
- Les données personnelles à cacher correspondent non seulement aux données directement nominatives (comme le nom, le prénom, le numéro de sécurité sociale, le sexe et l'adresse), mais aussi aux données indirectement nominatives (qui caractérisent la personne). En effet, il est souvent possible d'identifier un individu par un simple rapprochement de données personnelles de nature médicale ou sociale. Par exemple, l'âge, le sexe et le mois de sortie de l'hôpital, permettent d'isoler le patient dans une population restreinte ; la donnée de deux dates (voire de deux semaines) d'accouchement pour une femme permet de l'isoler dans une population plus grande (typiquement, la population d'un pays comme la France).
- Conformément à la législation en vigueur, certaines données collectées doivent être détruites après une durée limitée. Actuellement le règlement des archives hospitalières, impose des délais de conservation de 70 ans pour les dossiers de pédiatrie, de neurologie, de stomatologie et de maladies chroniques.
- Les données d'un patient n'apparaissent dans une certaine base de données (pour une étude médico-commerciale, par exemple) que si c'est obligatoire ou si ce patient donne son consentement ; de la même manière, il nous paraît évident d'avoir le consentement du patient pour lever l'anonymat.
- L'anonymisation est fondée sur un secret, l'identifiant anonyme du patient, qui n'est utilisé que pour chaîner les données médicales du patient, tout en respectant sa vie privée.

Certains des besoins identifiés montrent que les cartes à puces peuvent être une alternative innovante contribuant à la protection de la vie privée. En effet :

- avec la technologie actuelle des cartes à puces, il est possible d'anonymiser toutes les données identifiées comme personnelles[5].
- L'utilisation de la carte est le moyen idéal pour matérialiser le consentement du patient, du moins implicitement.
- La carte est un moyen supposé fiable pour protéger un secret (l'identifiant anonyme du patient) contre toute attaque visant sa lecture ou sa modification illicites. Si en plus, la transformation cryptographique (l'anonymisation ou la désanonymisation) est exécutée au sein de la carte, on peut garantir que le secret (l'identifiant) n'est jamais transmis (connu) en dehors de la carte.

La section suivante commence par identifier ce que nous entendons par objectifs et exigences d'anonymisation. La fin de la section revient en détail sur l'ensemble des besoins à travers un ensemble de scénarios, et identifie, pour chaque scénario, ses objectifs et ses exigences d'anonymisation. La dernière section de cet article

[5] Les cartes à puces actuelles sont capables de stocker les identifiants et supportent la réalisation de traitements simples comme MD5 ou SHA (pour lanonymisation).

achève l'application de notre méthodologie en proposant une nouvelle procédure d'anonymisation.

3.2 Objectifs d'anonymisation

Un objectif d'anonymisation est défini en fonction de l'une des trois propriétés suivantes applicables à la fonction d'anonymisation [10] :

- *réversibilité* : cacher les données par un simple chiffrement des données. Dans ce cas, il y a possibilité de remonter depuis les données chiffrées jusqu'aux données nominatives originelles.
- *irréversibilité* : c'est le cas réel de l'anonymisation ; une fois remplacés par des identifiants anonymes, les identifiants nominatifs originels ne sont plus recouvrables ; cependant, avec les techniques d'attaques par inférence[6], les identifiants anonymes, s'ils sont trop universellement utilisés, risquent de permettre la découverte d'identités mal cachées ; pour ce type d'anonymisation, la technique communément utilisée est une fonction de hachage ;
- *inversibilité* : c'est un cas où il est impossible en pratique de remonter aux données nominatives, sauf en appliquant une procédure exceptionnelle sous surveillance d'une instance légitime (médecin-conseil, médecin inspecteur) garante du respect de la vie privée des individus concernés ; il s'agit cette fois-ci d'une pseudonymisation au sens des critères communs [8] (cf. section 2).

3.3 Exigences d'anonymisation

Des informations sur l'environnement informatique (utilisateurs, attaques, etc.) du système étudié permettent de compléter l'analyse du besoin. En l'occurrence, même si les données sont anonymes, un utilisateur malveillant peut construire divers types de raisonnement pour déduire des informations confidentielles. Les exigences d'anonymisation sont exprimées en terme de chaînage (continuité de l'anonymisation) et de robustesse (sûreté de l'anonymisation).

Le chaînage permet d'associer un ou plusieurs identifiants anonymes à une même personne physique. Comme indiqué sur la figure 1, un chaînage peut être temporel (toujours, parfois, jamais) ; géographique (international, national, régional, local) ; ou spatio-temporel (par exemple, "toujours et partout", "parfois et partout", "local et jamais") [11].

La robustesse d'un système d'anonymisation est constituée de l'ensemble des caractéristiques à satisfaire vis-à-vis d'attaques ayant pour but de lever l'anonymat de façon non-autorisée. Il peut s'agir d'une robustesse à la réversion concernant la possibilité d'inverser la fonction d'anonymisation, mais il peut aussi s'agir d'une robustesse à l'inférence qui consiste à déterminer des informations nominatives à partir d'éléments d'informations purement anonymes. En général, une inférence peut être :

[6] Une inférence est la découverte de données confidentielles non directement accessibles, rendue possible par la mise en correspondance de plusieurs données légitimement accessibles, révélant des informations relatives à une personne.

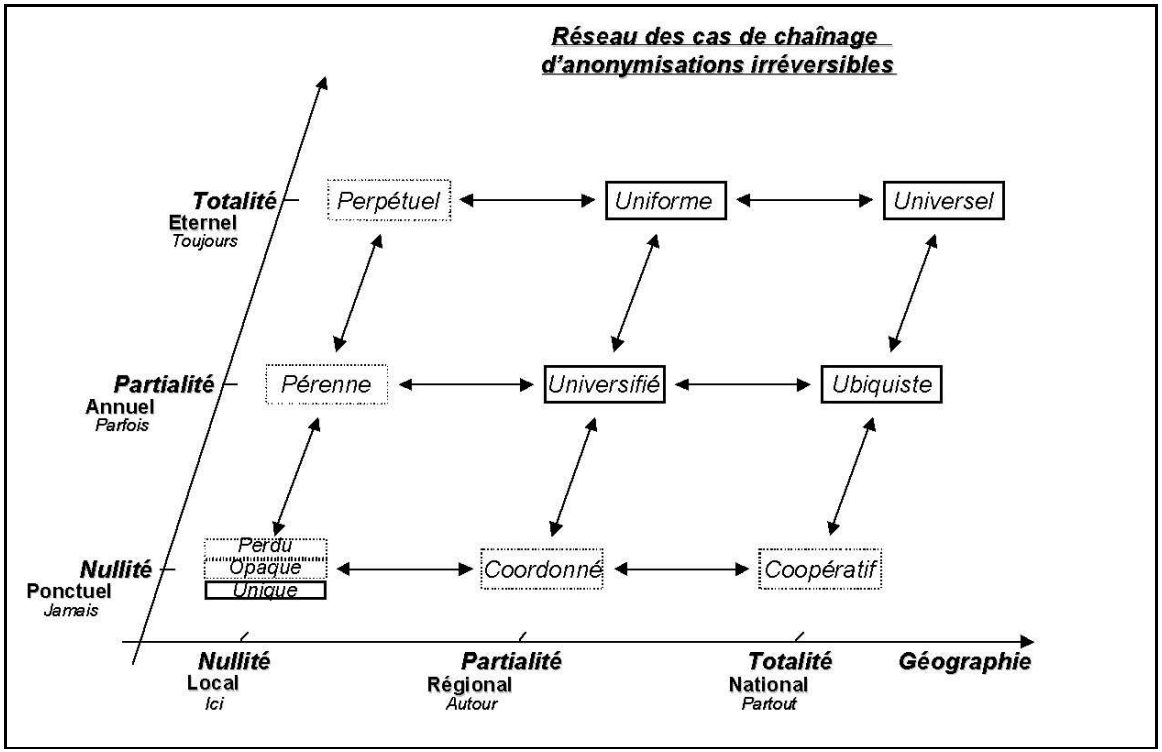


FIG. 1. Anonymisation en cascade : de l'universalité jusqu'à l'unicité

- *déductive* : elle utilise la logique du premier ordre (valeurs : oui, non ; opérateurs : et, ou) pour déduire des informations confidentielles non accessibles ; par exemple, si un certain patient fait un test de dépistage puis dans les quelques jours qui suivent, fait un test de dosage, alors le résultat du dépistage était positif ;
- *inductive* : s'apparente souvent à des raisonnements de type loi des grands nombres sans forcément l'appliquer sur de grandes échelles ; cela consiste par exemple, à induire qu'un tel patient est très certainement atteint de telle pathologie compte tenu du fait qu'il lui est prescrit tels médicaments comme il est d'usage pour cette pathologie ;
- *abductive* : lorsqu'un raisonnement classique utilisant les informations explicitement stockées dans le système d'informations ne permet pas d'inférer d'informations, mais ce raisonnement pourrait être complété en faisant des hypothèses sur certaines informations, par exemple, "et s'il avait un cancer, cela expliquerait pourquoi il s'absente du Conseil des Ministres pour se rendre à l'hôpital Paul Brousse de Villejuif..."
- *probabiliste* (ou adductive) : elle parvient à estimer la vraisemblance d'une information sensible en utilisant les informations accessibles, par exemple, "puisque P est traité à l'hôpital H, et puisque H est spécialisé dans les maladies M_1 et M_2, et puisque à son âge, la probabilité d'avoir M_1 est très faible (10%), alors on peut déduire qu'à 90%, P est atteint de M_2".

Cette liste n'est pas exhaustive et on peut naturellement imaginer d'autres types de canaux d'inférence fondés sur d'autres types de raisonnement, tel que le raisonnement par évidence ou par analogie.

3.4 Choix de solutions

Compte tenu du système, les sections précédentes tentent de donner sens aux bonnes questions à se poser :

- le type de réversibilité : anonymisation irréversible, réversible ou bien inversible ;
- le type de chaînage : chaînage spatial, temporel ou bien spatio-temporel ?
- la forme du chaînage : chaînage nul, partiel ou total ?
- la robustesse à la réversion : réversion directe (inversion) ou bien à indirecte (reconstruction) ;
- la robustesse à l'inférence : inférence déductive, inductive, abductive ou bien adductive ?
- ...

En réponse aux questions posées supra, il est possible de définir une politique d'anonymisation, au travers des choix suivants qui parfois dérivent naturellement en fonction des fonctionnalités quand elles sont correctement exprimées ou bien restent ouvertes s'il n'y a pas de recommandation :

- type de solution : organisationnelle, mécanisme cryptographique ou fonction à sens unique ?
- pluralité de la solution : mono-anonymisation, bi-anonymisation ou bien multi-anonymisation ?

- interopérabilité de la solution : par transcodage (manuel), translation (mathématique) ou bien transformation (automatique) ?
- ...

4 Analyse de scénarios du domaine santé-social

4.1 Lors du transfert des données médicales

La sensibilité des informations échangées entre professionnels de santé (par exemple, le laboratoire d'analyses et le médecin) met en évidence le besoin de confidentialité et d'intégrité des données transitant sur le réseau de soins. La figure 2 schématise une des solutions qui consiste à utiliser un chiffrement asymétrique. Ainsi, en supposant que le destinataire légitime est le seul à disposer de la clé privée, personne d'autre ne peut déchiffrer le message transitant par le réseau et ainsi accéder aux données personnelles en clair. Si les données sont volumineuses, il est préférable d'utiliser un chiffrement hybride.

Selon la classification donnée précédemment, l'objectif est une anonymisation réversible, tandis que l'exigence est la robustesse à l'inversion. Notons que le

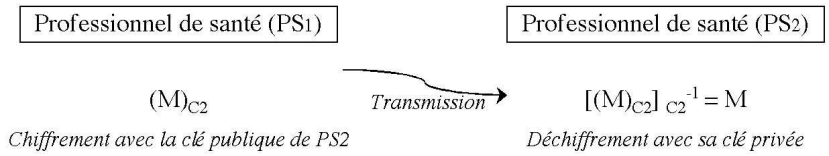


Fig. 2. Échange de données chiffrées entre professionnels de santé

chiffrement des données médicales transitant sur le réseau est actuellement une pratique de plus en plus répandue entre les professionnels de santé, notamment en utilisant S-MIME

4.2 Pour les unions professionnelles

Le transfert des données relatives aux activités des médecins vers les unions professionnelles se fait à des fins d'évaluation de l'activité des médecins. Une première exigence consiste donc à cacher les identités du patient et du médecin. Toutefois, l'anonymat des médecins doit pouvoir être levé pour l'évaluation de leurs comportements en vue de la qualité de soins. En effet, l'article L4134-4 du code de la santé publique ainsi que l'article 81 de la loi 94-43 [14] précisent que " les médecins conventionnés exerçant à titre libéral dans la circonscription de l'union sont tenus de faire parvenir à l'union les informations mentionnées à l'article L.161-29 du code de la sécurité sociale relatives à leur activité, sans que ces informations puissent être nominatives à l'égard des assurés sociaux ou de

leurs ayants-droit ou, à défaut, à condition qu'elles ne comportent ni leur nom, ni leur prénom, ni leur numéro d'inscription au Répertoire national d'identification des personnes physiques. Ces informations ne sont pas nominatives à l'égard des médecins ". Ces textes ajoutent que " L'anonymat (des médecins) ne peut être levé qu'afin d'analyser les résultats d'études menées dans le cadre de l'évaluation des comportements et des pratiques professionnelles en vue de la qualité des soins ". Le scénario décrit dans cette section est résumé dans la figure 3. En tenant

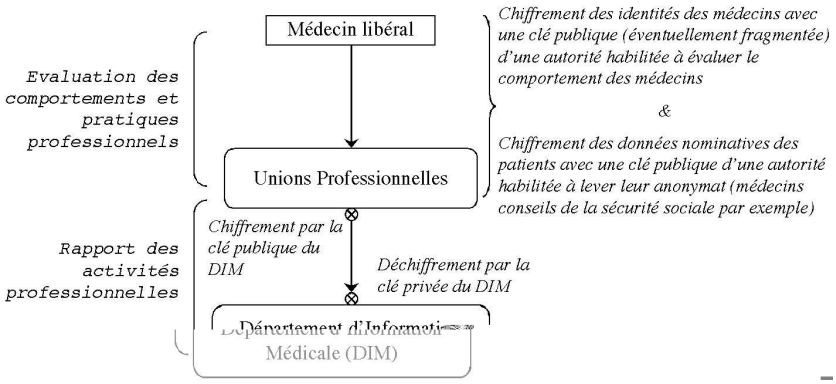


Fig. 3. Manipulations des identités au niveau des unions professionnelles

compte des besoins exprimés à travers cette législation, il conviendrait d'utiliser des pseudonymes pour les médecins et pour les patients. Les objectifs sont alors :

- l'anonymisation inversible de l'identité du médecin ; seule une autorité habilitée à évaluer les comportements des médecins pourrait rétablir les identités réelles des médecins ;
- l'anonymisation inversible des données nominatives du patient, seuls les médecins-conseils de la sécurité sociale pourront lever cet anonymat ; en effet, l'article L161-29 du code de la sécurité sociale ajoute : " seuls les praticiens-conseils et les personnels sous leur autorité ont accès aux données nominatives (des patients) issues du traitement susvisé, lorsqu'elles sont associées au numéro de code d'une pathologie diagnostiquée ".

Cette manière de faire évite les risques suivants (au niveau des unions professionnelles) :

- un utilisateur malhonnête qui tente d'avoir plus de détails sur les activités d'un médecin alors que la finalité de son traitement ne le justifie pas ; par exemple, dans le cadre d'une étude relative au fonctionnement du système

de santé, il n'est pas nécessaire d'accéder aux identités (respect du principe du moindre privilège[7]) ;
- atteinte à l'intimité des patients dans la mesure où ceux-ci peuvent confier des informations à certains professionnels de santé, sans pour autant avoir forcément envie de les communiquer aux autres professionnels de santé ou personnes en charge des traitements au sein des unions.

4.3 Dans le cadre du PMSI

Le Programme de Médicalisation des Systèmes d'Information (PMSI) est un système d'analyse de l'activité des établissements de santé dont la finalité est l'allocation des ressources tout en diminuant les inégalités budgétaires. Le PMSI a été expérimenté depuis 1983, et généralisé dans les hôpitaux publics et privés participant au service public par la circulaire du 24 juillet 1989 [13] pour l'activité de MCO (Médecine, Chirurgie, Obstétrique). Son utilisation à des fins budgétaires a été formalisée par la circulaire du 7 décembre 1996 [14]. Il a été étendu aux établissements privés par les ordonnances du 24 avril 1996. La circulaire du 9 mars 1998 [15] a généralisé le PMSI aux établissements publics ayant une activité de Soins, de Suite et de Réadaptation. Une multitude de textes ont été élaborés pour réglementer le fonctionnement du PMSI. Citons à titre indicatif, la loi du 31 juillet 1991 [16], le décret du 27/07/94 [17] ainsi que les arrêtés des 20/09/1994, 22/07/1996 et 29/07/1998 [18].

Dans la pratique, chaque séjour d'un patient donne lieu à un recueil standardisé de données de nature administrative (dates d'entrée et de sortie, date de naissance, nom et prénom, par exemple) et de nature médicale (diagnostics, actes codés). Les séjours sont ensuite classés selon l'indicateur médico-économique "Groupe Homogène de Malades" (GHM). Les patients d'un GHM donné sont considérés comme ayant mobilisé des ressources de même ampleur. Chaque année une échelle des coûts affecte un coût relatif à chaque GHM, mesuré en points ISA pour "Indice Synthétique d'Activité". Les données du PMSI des établissements publics sont anonymisées, puis transmises semestriellement aux Agences Régionales de l'Hospitalisation (ARH) qui les utilisent pour l'allocation budgétaire. Celles des établissements privés sont transmises trimestriellement à la CNAM-TS, en attendant de devenir un outil d'allocation de ressources. Plus précisément, tout séjour hospitalier effectué dans la partie court séjour d'un établissement fait l'objet d'un Résumé de Sortie Standardisé (RSS), constitué d'un ou plusieurs Résumés d'Unité Médicale (RUM). Le RUM contient des données (administratives et médicales) concernant le séjour d'un patient dans une unité médicale donnée. À partir des RUM récupérés et validés, le Département d'Information Médicale (DIM) construit le fichier des Résumés de Sortie Standardisés (RSS) à l'aide d'un logiciel regroupeur. Les services des statistiques et des études épidémiologiques reçoivent du médecin du DIM, les données médicales et admi-

[7] Le principe du moindre privilège impose que tout utilisateur ne doit pouvoir accéder à un instant donné quaux informations et services strictement nécessaires pour laccomplissement du travail qui lui a été confié.

nistratives figurant sur les Résumés de Sortie Anonymisés (RSA). La procédure générale est donnée sur la figure 4.

Étant donné que la finalité du PMSI est purement médico-économique (et non pas directement épidémiologique),

- le besoin est de pouvoir effectuer des trajectoires de soins par le biais d'une anonymisation (au sens des critères communs [8]) ;
- l'objectif est une anonymisation irréversible ; et
- les exigences sont un chaînage universel (toujours et partout le même identifiant pour un patient donné) ainsi que la robustesse à la réversion et aux inférences (déductives, inductives, abductives, etc.).

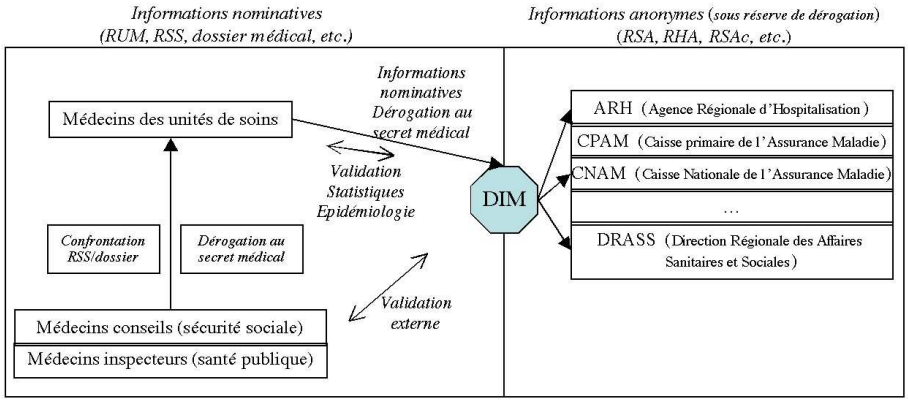


Fig. 4. Frontières des données nominatives, anonymes et anonymisables.

4.4 Traitement des maladies à déclaration obligatoire

Les maladies dont la surveillance est nécessaire à la conduite et à l'évaluation de la politique de santé publique (le SIDA, par exemple), ou qui nécessitent une intervention urgente locale (méningite, choléra, rage) sont des maladies à déclaration obligatoire. À l'origine, les fichiers des patients séropositifs sont nominatifs, mais ils sont anonymisés (anonymisation irréversible) avant toute transmission.

Les besoins sont divers : prévention, production de soins, veille sanitaire, analyses épidémiologiques, etc. L'objectif principal est l'irréversibilité de la fonction d'anonymisation. Le chaînage universel et la robustesse à la réversion et aux attaques par inférence constituent les principales exigences.

À cet égard, le type de protection doit dépendre des objectifs. En effet, s'agit-il d'obtenir, année par année, un état exhaustif du nombre de séropositifs pour connaître l'évolution de l'épidémie, ou d'évaluer, de façon globale, l'impact des actions de prévention ? S'agit-il encore d'instituer une véritable surveillance épidémiologique de l'évolution des cas d'infection par le VIH, du stade de la décou-

verte de la séropositivité, à l'apparition éventuelle du SIDA avéré ? Dans ce cas, l'objectif est de mesurer de façon fine l'impact des actions thérapeutiques et de prévention nécessitant un suivi des cas.

Ce choix d'objectifs comporte des conséquences importantes tant sur la nature des données susceptibles d'être collectées que sur leur durée de conservation et les liens éventuels avec d'autres systèmes de surveillance. Il implique en conséquence des choix en terme de protection de données.

Appliquer l'anonymisation à la source et disposer de mesures de sécurité adéquates ne dispensent pas de s'interroger sur la pertinence des autres informations figurant sur la déclaration de séropositivité. Il s'agit en particulier, du code postal de résidence, la profession et l'origine géographique.

- Le code postal de domicile : si l'objectif est de mieux cibler les actions de prévention locale, sa collecte semble nécessaire. Néanmoins, la pertinence du recueil de cette donnée n'est pas à ce jour réellement démontrée. En outre, sa collecte et son expiration pourraient être de nature à permettre une localisation géographique précise surtout dans les petites communes. Dès lors le recueil sous une forme aussi détaillée que le code postal du lieu de résidence des personnes séropositives peut paraître excessif au regard des objectifs recherchés et il est probablement plus judicieux d'utiliser le code du département au lieu du code postal.
- La profession : ne paraît pas nécessaire de disposer de la profession précise ; une simple mention des catégories socio-professionnelles selon la nomenclature de l'INSEE paraît être pertinente.
- L'origine géographique : il serait peut-être suffisant de mentionner si la personne est originaire d'un pays où la transmission hétérosexuelle est prédominante ou si elle a eu des relations sexuelles avec une personne ayant vécu dans un pays où la contamination hétérosexuelle est prédominante.

Actuellement en France, il semble que le but est d'assurer un suivi des cas et de mesurer l'évolution du SIDA. Il est donc nécessaire de collecter des données individualisées afin de permettre de détecter les doublons, de vérifier les données auprès des professionnels de santé, etc. En ce qui concerne l'appauvrissement des données, l'Institut de Veille Sanitaire a montré que l'utilisation d'informations indirectement nominatives figurant sur les déclarations du SIDA (initiales du nom et prénom, date de naissance, département de domicile) permet de repérer plus de 99% des doublons. Un appauvrissement trop important des données peut donc fausser les statistiques et remettre en cause la fiabilité scientifique de la surveillance épidémiologique.

4.5 Traitement des données statistiques

En aucun cas, les données médicales à caractère personnel ne peuvent être manipulées pour des traitements à des fins non-épidémiologiques, en l'occurrence, des traitements purement statistiques ou à des fins de publications scientifiques. À cet égard, non seulement ces données doivent être anonymisées, mais il doit être impossible de les ré-identifier. Ainsi, s'imposent l'irréversibilité de l'anonymisation ainsi que la robustesse aux inférences. En effet, même après

anonymisation, les identités peuvent être déduites par un statisticien en combinant plusieurs requêtes ou en complétant son raisonnement par des hypothèses ou par des informations externes au système.

Le domaine de l'inférence d'information dans les bases de données a été étudié depuis de nombreuses années, et il a fait l'objet d'une littérature abondante [19,20,21]. La référence [22] explique que la sécurité dans les bases de données statistiques est un problème réel, que plusieurs suggestions apparaissent dans la littérature, mais qu'il est difficile de décider si l'une d'entre elles est vraiment satisfaisante. Par exemple, une solution serait de permuter les valeurs des attributs des n-uplets (lignes) de chaque table de la base de données de sorte que la précision globale de la statistique est conservée, alors que les réponses précises (concernant des personnes identifiées) seront fausses. La difficulté inhérente à cette approche réside dans la recherche des ensembles d'entrées dont les valeurs peuvent être permutées de cette façon. Une autre solution pourrait être le brouillage, qui consiste à modifier les réponses aux requêtes statistiques en y ajoutant du "bruit" aléatoire pour rendre plus difficile le recoupement entre requêtes.

4.6 Études épidémiologiques focalisées

Le PMSI traite des informations médico-administratives, économiques et statistiques, afin de réaliser des analyses pertinentes des bases de données régionales et nationales. Les données traitées sont anonymes, et même si elles sont souvent chaînables, il n'y a généralement aucun moyen de lever l'anonymat. À l'inverse, dans d'autres types d'études, il est souvent souhaitable de revenir à l'identité réelle des patients afin d'améliorer la qualité des soins. Prenons à titre d'exemple, certaines études épidémiologiques focalisées : protocoles de recherche en cancer, maladies génétiques, rares...

Si, par exemple, ces études mettent en évidence la situation suivante : les patients de la catégorie C ayant subi certains traitements T_{avant} ont une espérance de vie considérablement réduite s'ils ne suivent pas le traitement $T_{\text{recouvrement}}$. Dans de telles situations, il faudra remonter aux identités réelles pour que les patients puissent profiter de ces résultats. Il s'agit ainsi d'une anonymisation inversible (pseudonymisation dans le sens des critères communs) : seules des personnes habilitées peuvent lever l'anonymat (*médecins conseils, médecins inspecteurs, médecins traitants*) et seulement quand c'est nécessaire (principe du moindre privilège).

Dans le cas des protocoles de recherche sur le cancer, le processus commence par un typage (stade de la maladie), puis par une identification du protocole correspondant au patient (s'il existe), enfin, selon le protocole, le patient est enregistré dans un registre régional, national, voire international. Les études épidémiologiques et statistiques faites sur ces registres peuvent dégager de nouveaux résultats concernant les patients d'un certain protocole. Dans le but de raffiner les études et faire avancer la recherche scientifique, il est parfois utile de remonter aux identités réelles des patients pour les identifier, faire des recoupe-

ments entre plusieurs données déjà recueillies, et les compléter a posteriori. Là encore, la pseudonymisation semble nécessaire.

5 Une nouvelle solution générique

5.1 Schéma général

La section précédente préconise que toute anonymisation nécessite une étude préalable judicieuse, identifiant de manière claire et explicite les besoins, les objectifs ainsi que les exigences. Par ailleurs, l'application de cette démarche à un certain nombre de scénarios identifiés nous a permis de proposer une nouvelle solution générique qui satisfait les exigences soulevées.

Afin de décider quelle vue (forme spécifique de données) est accessible par quel utilisateur, notre solution prend en considération le rôle que joue cet utilisateur, son établissement de rattachement ainsi que la finalité du traitement que subiront les données de cette vue. Bien entendu, ceci respecte le principe du moindre privilège et met en oeuvre les recommandations de la norme européenne [7].

Pour cela, et comme indiqué sur la figure 5 et détaillé dans la suite de cette section, plusieurs traitements et transformations cryptographiques sont effectués au niveau des hôpitaux, en amont et en aval des centres de traitements (avant la distribution aux utilisateurs finaux).

Transformations au niveau des fournisseurs de données sensibles À l'hôpital, trois types de bases de données peuvent être distinguées :

- une base de donnée administrative accessible par les personnels administratifs, chacun selon ses fonctions,
- une base de donnée médicale dont l'accès est restreint aux personnels soignants en charge des patients,
- des bases de données anonymes, dont chacune contient les informations nécessaires et suffisantes pour un projet donné (un centre de traitement).

Le passage de la base de données médicale à une base anonyme (destinée à un certain projet) nécessite l'application de deux transformations, T1 et T2, aux données à transférer.

La transformation T1 : consiste à obtenir "$\mathrm{IDA}_{\mathrm{pat}|\mathrm{Proj}}$", un identifiant anonyme par personne et par projet, à partir des deux identifiants :

- "$\mathrm{ID}_{\mathrm{proj}}$", l'identifiant du projet, qui est détenu par les établissements de soins (hôpitaux, cliniques) ;
- "$\mathrm{ID}_{\mathrm{pat}}$", l'identifiant anonyme détenu par le patient sur la carte VITALE (rappelons que $\mathrm{ID}_{\mathrm{pat}}$ est généré aléatoirement et n'a aucun lien avec le numéro de sécurité sociale) ; une longueur de 128 bits nous paraît suffisante pour éviter des collisions (risque que deux personnes différentes aient le même identifiant).

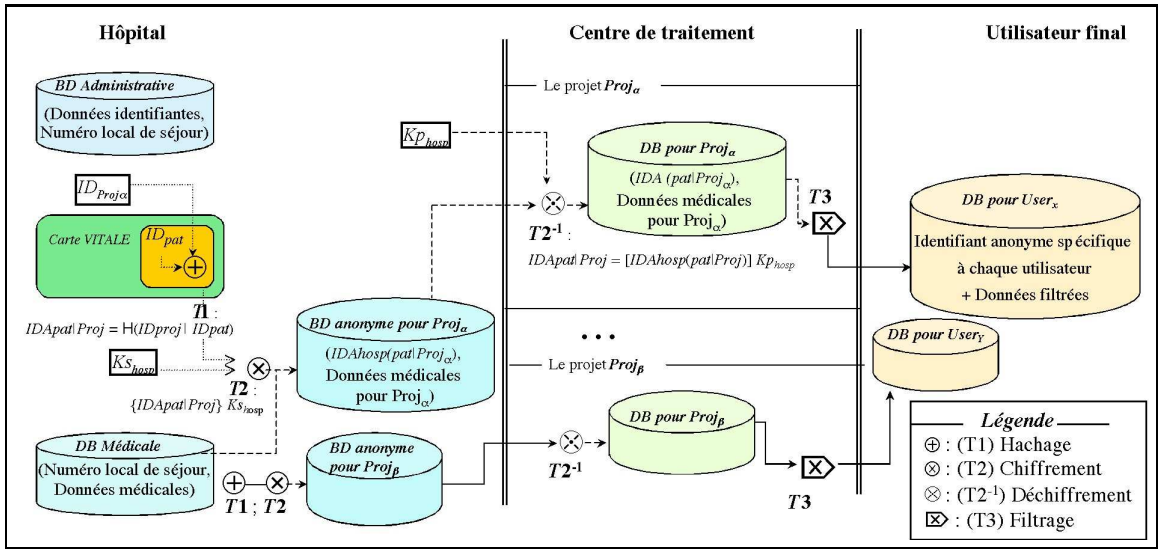


Fig. 5. Procédure d'anonymisation proposée.

Au niveau de l'hôpital, et lors de l'alimentation des bases de données anonymes (par projet), l'utilisateur (employé de l'hôpital par exemple) envoie $\mathrm{ID}_{\mathrm{proj}}$ (l'identifiant du projet concerné par la base de donnée) à la carte ; celle-ci contient déjà $\mathrm{ID}_{\mathrm{pat}}$ (l'identité du patient donnant son consentement pour l'exploitation de ses données médicales par le projet). La procédure T1 consiste à appliquer une fonction de hachage (MD5 ou SHA par exemple) à $(\mathrm{ID}_{\mathrm{proj}}|\mathrm{ID}_{\mathrm{pat}})$, la concaténation de $\mathrm{ID}_{\mathrm{proj}}$ et $\mathrm{ID}_{\mathrm{pat}}$:

$$\text{T1} \qquad \mathrm{IDA}_{\mathrm{pat}|\mathrm{proj}} = H(\mathrm{ID}_{\mathrm{proj}}|\mathrm{ID}_{\mathrm{pat}})$$

La transformation T1, réalisée au sein de la carte VITALE[8] du patient, et produisant l'empreinte $H(\mathrm{ID}_{\mathrm{proj}}|\mathrm{ID}_{\mathrm{pat}})$, vise les objectifs suivants :

- un patient n'apparaît dans une base de donnée anonyme que si cela est obligatoire (par exemple pour le PMSI) ou s'il donne son consentement à travers la fourniture de son identifiant (pour une étude de nature médico-commerciale, par exemple) ;
- l'identifiant anonyme $\mathrm{IDA}_{\mathrm{pat}|\mathrm{proj}}$ n'utilise aucun secret dont la divulgation porterait atteinte à la vie privée des autres personnes (contrairement à l'utilisation d'une clé secrète commune pour tous les patients). De plus, puisque le calcul de l'empreinte $\mathrm{IDA}_{\mathrm{pat}|\mathrm{proj}}$ s'effectue au niveau de la carte, $\mathrm{ID}_{\mathrm{pat}}$ reste toujours au sein de la carte ; il n'est jamais stocké isolement, et il n'est utilisé qu'afin de créer une entrée dans la base anonyme pour un projet donné (au niveau de l'hôpital) ;
- puisque $\mathrm{ID}_{\mathrm{proj}}$ est spécifique à chaque projet, les risques de rapprochements non-autorisés des données de deux projets différents sont écartés, ou du moins sont peu vraisemblables ; de plus, les bases de données anonymes (par projet) sont isolées de l'extérieur de l'hôpital et sont soumises à des mesures strictes de contrôle d'accès ;
- sachant que l'empreinte $\mathrm{IDA}_{\mathrm{pat}|\mathrm{proj}}$ est toujours la même pour un patient et un projet donnés, il est possible que chaque projet puisse faire des rapprochements de données concernant un même patient.

Néanmoins, la transformation T1 ne permet pas de se prémunir contre certaines attaques où les intrus essayent de faire des rapprochements d'informations (concernant un projet donné) détenus par deux hôpitaux différents. En effet, supposant que le patient Paul a été traité à Rangueil et à Purpan, et que dans chacun de ces deux hôpitaux, Paul est consentant de l'utilisation de ses données pour un projet "Projα". Supposant qu'un employé de Purpan, nommé Bob, sait que l'empreinte X (= $\mathrm{IDA}_{\mathrm{Paul}|\mathrm{Proj}_\alpha}$) correspond à Paul. Supposant en plus, que Bob arrive à s'emparer de la base de donnée anonyme concernant Projα, mais détenue par Rangueil. Dans ce cas, l'utilisateur malveillant Bob peut facilement établir le lien entre le patient Paul et ses données médicales (concernant

[8] Il sagira probablement dune nouvelle génération de cartes VITALE qui supporteraient la réalisation de traitements simples comme MD5 ou SHA.

Projα) détenues par Rangueil (en plus de celles détenues par Purpan, puisque Bob travaille à Purpan).

Afin de faire face à ce type d'attaques, nous introduisant la *transformation asymétrique* T2 au niveau de l'hôpital. Ainsi, avant de stocker les données dans les bases de données anonymes spécifiques à chaque projet, l'hôpital chiffre (chiffrement asymétrique) l'identifiant $\mathrm{IDA}_{\mathrm{pat|proj}}$ avec une clé $Ks_{\mathrm{hôp}}$ spécifique à l'hôpital ; ("$\{\}_K$" désigne un chiffrement avec K) :

$$\text{T2} \qquad \mathrm{IDA}_{\mathrm{hôp}}(\mathrm{pat|Proj}) = \{IDA\}_{Ks_{\mathrm{hôp}}}$$

Si on reprend le scénario précédent, l'utilisateur malveillant Bob ne peut guère revenir aux identités des personnes car il ne dispose pas de la clé de déchiffrement Kp_{Purpan}. En effet, chaque hôpital détient sa clé $Ks_{\mathrm{hôp}}$, tandis que $Kp_{\mathrm{hôp}}$ n'est détenue que par les projets.

Il est facile de constater que les deux transformations (T1 et T2) effectuées au niveau des hôpitaux permettent d'avoir une grande robustesse vis-à-vis d'attaques ayant pour but de lever l'anonymat (ou de faire des rapprochements) de façon non autorisée. Pour autant, la procédure proposée reste assez flexible. En effet, si deux hôpitaux ($\mathrm{hôp}_a$ et $\mathrm{hôp}_b$) décident de fusionner un jour, il est tout à fait possible de relier les données concernant chaque patient ; que ces données proviennent de $\mathrm{hôp}_a$ ou $\mathrm{hôp}_b$.

En effet, il suffit que chaque hôpital déchiffre ses données avec sa clé $Kp_{\mathrm{hôp}}$, puis chiffre le résultat avec la clé privée $Kp_{\mathrm{hôp}_{ab}}$ du nouvel hôpital. Ainsi, si $\mathrm{IDA}_{\mathrm{hôp}_a}(pat|Proj)$ (respectivement $\mathrm{IDA}_{\mathrm{hôp}_b}(pat|Proj)$) désigne un identifiant anonyme au sein de l'hôpital $\mathrm{hôp}_a$ (respectivement $\mathrm{hôp}_b$) ; $[]_K$ désignant le déchiffrement avec K :

– Le traitement effectué sur les anciennes données de l'hôpital $\mathrm{hôp}_a$ est :

$$\{[\mathrm{IDA}_{\mathrm{hôp}_a}(\mathrm{pat|Proj})]Kp_{\mathrm{hôp}_a}\}Ks_{\mathrm{hôp}_{ab}}$$

– Le traitement effectué sur les anciennes données de l'hôpital $\mathrm{hôp}_b$ est,

$$\{[\mathrm{IDA}_{\mathrm{hôp}_b}(\mathrm{pat|Proj})]Kp_{\mathrm{hôp}_b}\}Ks_{\mathrm{hôp}_{ab}}$$

Remarquons que les codes de liaisons obtenus seront les mêmes dans les deux établissements (pour chaque base de donnée anonyme associé à un certain projet).

Pour les utilisateurs internes aux établissements de soins, les mécanismes de contrôles d'accès doivent interdire tout accès non-autorisé, tandis que des mécanismes de détection et de tolérance aux intrusions doivent renforcer les autres mesures de sécurité.

Transformations en amont des centres de traitements Les données contenues dans les bases de données anonymes (au niveau des hôpitaux) subissent des transformations qui dépendent de l'identifiant anonyme $\mathrm{IDA}_{\mathrm{pat|proj}}$ et de la clé

$Ks_{\text{hôp}}$. Pour retrouver les données qui lui sont destinées, chaque centre de traitement (correspondant à un projet) déchiffre les données qui lui sont envoyées par la clé $Kp_{\text{hôp}}$ de l'hôpital transmetteur (d'après (T2)) :

$$[\text{IDA}_{\text{hôp}(pat|Proj)}]Kp_{\text{hôp}} = [\{\text{IDA}_{\text{pat|proj}}\}Ks_{\text{hôp}}]Kp_{\text{hôp}} = \text{IDA}_{\text{pat|proj}}$$

Le centre de traitement retrouve ainsi les informations suffisantes et nécessaires aux traitements qu'il effectue. Ces informations sont associées aux identifiants anonymes $\text{IDA}_{\text{pat|proj}}$, ce qui permet à chaque projet de chaîner les données de chaque patient.

Transformation avant la distribution aux utilisateurs finaux Avant leur distribution aux utilisateurs finaux (recherche scientifique, publications, Web, presse...), et afin de respecter le plus possible le principe du moindre privilège, les informations transférées peuvent éventuellement subir un traitement de filtrage ciblé pour chaque catégorie d'utilisateurs. Il peut, par exemple, s'agir d'une agrégation, d'un appauvrissement des données, etc.

Si de plus, l'objectif de sécurité est d'interdire à deux (ou plusieurs) utilisateurs finaux de recouper les informations, il convient d'appliquer une autre anonymisation (MD5, par exemple) avec une clé secrète $K_{\text{util|proj}}$.

$$\text{IDA}_{pat|util} = H(\text{IDA}_{pat|Proj}|K_{util|proj})$$

En fait, selon le besoin, ce dernier cas peut correspondre à deux situations (et donc procédures) différentes :

- si le but est de permettre à l'utilisateur de faire des chaînages dans le temps (par projet), la clé $K_{util|proj}$ doit être stockée au niveau du centre de traitement, de façon à pouvoir la réutiliser, à chaque fois que celui-ci souhaite transmettre d'autres informations à cet utilisateur ; à l'inverse,
- si le centre souhaite empêcher le chaînage dans le temps par les utilisateurs, la clé est générée aléatoirement à chaque distribution.

Désanonymisation L'analyse des scénarios (cf. Section 4) montre qu'il est parfois souhaitable, voir nécessaire de lever l'anonymat. En outre, l'étude des besoins préconise le consentement du patient pour la réalisation de cette procédure (désanonymisation). Afin de satisfaire ces besoins, nous proposons que si un utilisateur final (chercheur dans le domaine des maladies orphelines par exemple) découvre une information qui nécessiterait de remonter aux identités des patients, il doit d'abord renvoyer ses résultats aux hôpitaux participant au projet concerné (probablement via les projets). En se présentant à un de ces hôpitaux, et en fournissant sa carte VITALE, le patient donne son consentement pour lever l'anonymat, et associer ainsi les nouvelles informations (résultat de la recherche scientifique, par exemple) à l'identité réelle du patient.

Celui-ci pourrait ainsi bénéficier de ces résultats. ID_{pat} figurant sur la carte, ainsi que ID_{Proj} et $Ks_{\text{hôp}}$ fournis par le système de l'hôpital, permettraient de

calculer

$$\mathrm{IDA}_{pat|Proj} = H(\mathrm{ID}_{proj|}\mathrm{ID}_{pat)}$$

et

$$\mathrm{IDAhÙp}_{(}pat|Proj) = \{\mathrm{IDA}_{pat|Proj}\}_{Ks}\mathrm{hôp}.$$

C'est donc la seule façon d'établir le lien entre le patient, ses identifiants anonymes et ses informations médicales. Une comparaison entre l'identifiant anonyme du patient et la liste des inversions (envoyée par l'utilisateur final) permettrait de déclencher une alarme demandant au patient s'il souhaite consulter l'information transmise.

Discussion La solution que nous proposons vise à garantir les points suivants :

- L'identifiant anonyme du patient est un secret qui est protégé de tout accès illicite (en lecture ou en modification). Cette donnée sensible est générée aléatoirement au sein de la carte, dispositif supposé fiable et très difficilement falsifiable ; de plus, les identifiants anonymes spécifiques aux projets sont calculés au sein de la carte. Le secret (ID_{pat}) n'est donc jamais transmis en dehors de la carte, ni altéré illicitement.
- L'utilisateur doit donner explicitement son consentement pour toute utilisation non-obligatoire, mais souhaitable, de ses données. De cette manière, tout chaînage de données personnelles ainsi que toute procédure destinée à lever l'anonymat, sont strictement contrôlés par l'utilisateur. La solution résiste aux attaques par dictionnaire et à tous les niveaux : établissements fournisseurs de données sensibles, centres de traitements et utilisateurs finaux.
- La séquence d'anonymisation (anonymisation en cascade) que nous proposons à différents niveaux, combinée avec des mécanismes de contrôles d'accès, permet de garantir, en toute robustesse, l'exigence de non inversibilité ainsi que le principe du moindre privilège.
- Les identifiants anonymes générés étant spécifiques à un secteur particulier (projet, domaine d'activité, centre d'intérêt, branche professionnelle, établissement, etc.), il est possible d'adapter la solution à chaque secteur (par exemple lorsque le centre de traitement est le seul utilisateur) ;
- Il est possible de fusionner les données de deux (voir de plusieurs) établissements sans compromettre la flexibilité et la sécurité.
- La manière selon laquelle l'information est distribuée et utilisée par l'utilisateur final est importante. Notre solution peut être adoptée pour tenir compte de la finalité du traitement.

Actuellement, les hôpitaux français utilisent l'algorithme de hachage SHA (Standard Hash Algorithm) pour transformer, d'une manière irréversible, les variables d'identification : nom, prénom, date de naissance et sexe. Le but est d'obtenir un identifiant strictement anonyme, mais toujours le même pour un patient donné [?]. Afin de chaîner les informations concernant le même patient, le code anonyme obtenu (après anonymisation) est toujours le même pour un individu donné.

deux clés ont été ajoutées à l'algorithme de hachage SHA. La première clé k_1, utilisée par tous les émetteurs des données (hôpitaux et médecins), est concaténée à l'identité. Une fonction de hachage est ensuite appliquée au résultat :

$$\text{empreinte}_1 = H(k_1|\text{identité}).$$

Cette opération produit une empreinte qui varie d'une identité à l'autre, mais qui est toujours la même pour un patient donné. Les informations transmises au centre de traitement des fichiers (DIM) en vue de leur rapprochement sont ainsi devenues strictement anonymes et les personnes qui assurent les traitements centralisés ne peuvent pas lever l'anonymat à l'aide d'une attaque par dictionnaire puisqu'elles ne connaissent pas la clé k_1. De l'autre côté de la communication, les informations reçues par le DIM sont hachées par le même algorithme mais avec une seconde clé k_2, qui n'est pas communiquée aux hôpitaux (voir figure 6) :

$$\text{empreinte}_2 = H(k_2 + \text{empreinte}_1).$$

Il est évident que ce protocole s'avère complexe et risqué. En effet, il nécessite

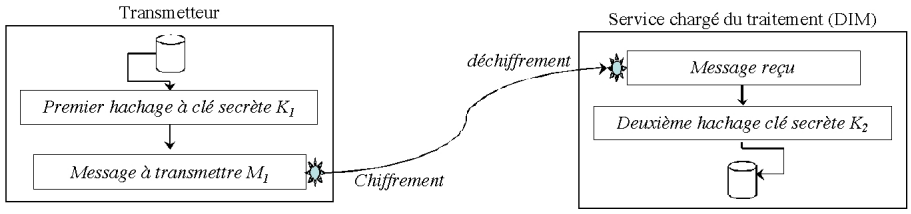


Fig. 6. Les grandes lignes de la procédure de hachage utilisée actuellement dans les hôpitaux français

une distribution de la même clé secrète à tous les fournisseurs d'informations (médecins libéraux, hôpitaux, cliniques...), tout en supposant que cette clé doit rester secrète. Si une clé est corrompue, le niveau de sécurité est considérablement réduit. De même, si un jour il s'avère que l'algorithme (ou la longueur de la clé) n'est plus efficace, comment faire le rapprochement entre les identifiants avant et après changement de l'algorithme ou de la clé (sachant que les empreintes dépendent de la clé supposée être toujours la même et chez tous les fournisseurs d'informations) ? Si ce problème survient, la seule solution envisageable consiste à appliquer une autre transformation à toute la base de données, solution qui n'est guère aisée.

À l'inverse, dans notre solution, les identifiants (ID_{pat}, ID_{Proj}, $\text{IDA}_{pat|Proj}$ et $\text{IDA}_{pat|Util}$) utilisés dans les diverses transformations sont situés dans des endroits différents, et les clés ($Ks_{\text{hôp}}$, $Kp_{\text{hôp}}$) sont détenues par des personnes différentes. En outre, ID_{Proj} est spécifique à un seul projet ; la paire de clés ($Ks_{\text{hôp}}$, $Kp_{\text{hôp}}$) est relié à un seul hôpital ; $\text{IDA}_{pat|Util}$ est destinée à un seul

utilisateur ; etc. Il est donc pratiquement impossible de lever illicitement l'anonymat ou de réussir des inférences non autorisées. De plus, puisque ID_{pat} est spécifique à un seul patient (et ne figure que sur sa carte), sa divulgation (qui reste une tâche très difficile) ne compromet guère la sécurité totale du système.

Par ailleurs, nous pensons que la solution idéale n'existe pas, et nous suggérons de compléter notre solution, selon le cas étudié, par une combinaison de solutions techniques et organisationnelles :

- l'accès aux données doit être parfaitement contrôlé. Une politique de contrôle d'accès doit être définie et mise en place pour que les données ne soient accessibles qu'aux seuls utilisateurs habilités ;
- la spécification du système d'information et de l'architecture du réseau doit obéir à une politique globale de sécurité, et donc doit être adaptée aux besoins ;
- La définition de la politique de sécurité doit inclure une analyse des risques d'abduction ;
- la constitution de sous-bases de données régionales ou thématiques doit être contrôlée.
- Il convient d'utiliser (si cela est possible) des anonymisations thématiques, de sorte que même si un utilisateur parvient à casser l'anonymisation, les risques d'abduction soient limités à un thème donné ;
- Il faut séparer les données d'identité des renseignements proprement médicaux. Bien entendu ce mécanisme ne peut être appliqué que dans des contextes particuliers ;
- Il est parfois souhaitable de renforcer la surveillance des utilisations qui sont faites des données, notamment en définissant et en mettant en oeuvre des outils de détection d'intrusion ; en particulier, ces outils doivent permettre de détecter les requêtes, voire les enchaînements de requêtes, ayant un but malveillant (inférence de données, abus de pouvoir...) ;
- nous préconisons également l'utilisation d'autres techniques comme le brouillage ou le filtrage, de façon à ne pas répondre à des requêtes statistiques si l'information demandée est trop précise ; ....

6 Conclusion

Cet article propose d'utiliser la technologie des cartes à puces pour répondre à l'un des soucis récents, mais majeur, engendré par les nouvelles technologies de l'information et la communication : le respect de la vie privée et la protection de l'individu, dans une dimension électronique qui devient désormais omniprésente. Dans ce cadre, il analyse le problème d'anonymisation, identifie et étudie un certain nombre de scénarios représentatifs, et présente une démarche d'analyse mettant en correspondance des fonctionnalités d'anonymisation avec les solutions d'anonymisation adéquates. Enfin, il propose des procédures génériques, flexibles et adaptées aux besoins, objectifs et exigences de protection de la vie privée.

La solution proposée est en phase finale d'implémentation à travers un scénario complet du domaine médical, allant de l'enregistrement des données au

niveau des hôpitaux jusqu'aux utilisations finales (recherche scientifique, Web, presse...).

Nous envisageons de poursuivre ce travail en étudiant la complexité et en adaptant la solution à d'autres exemples comme les recensements démographiques, le commerce électronique ou le vote électronique.

Références

1. Résolution A/RES/45/95 Assemblée générale des Nations Unies, Principes directeurs pour la réglementation des fichiers personnels informatisés, 14 Décembre 1990.
2. Directive 95/46/CE du Parlement Européen, adoptée par le Conseil Européen le 24 juillet 1995, On the protection of individuals with regard to the processing of personnal data and on the free movement of such data, 1995.
3. Directive du Parlement Européen n° 2002/58/EC concernant "le traitement des données à caractère personnel et la protection de la vie privée dans le secteur de télécommunications électroniques", 12 juillet 2002, Journal Officiel L 201, 31-7-2002, pp. 37-47.
4. Recommandations du Conseil de l'Europe, R(97)5, On The Protection of Medical Data Banks, Council of Europe, Strasbourg, 13 février 1997.
5. Loi 78-17 du 6 janvier 1978 relative à l'Informatique, aux fichiers et aux libertés, Journal officiel de la République française, pp. 227-231, décret d'application 78-774 du 17 juillet 1978, pp. 2906-2907.
6. Loi 2002-303 du 4 mars 2002 relative aux droits des malades et à la qualité du système de santé, article L. 1111-7.
7. CEN/TC 251/WG I, Norme prENV 13606-3 : Health Informatics - Electronic Healthcare Record Communication, n° 99-046, Comité Européen de Normalisation, 27 mai 1999.
8. Common Criteria for Information Technology Security Evaluation, Part 1 : Introduction and general model, 60 p., ISO/IEC 15408-1 (1999).
9. A. Abou El Kalam, Y. Deswarte, G. Trouessin, E. Cordonnier, "Gestion des données médicales anonymisées : problèmes et solutions",2ème conférence francophone en gestion et ingénierie des systèmes hospitaliers (GISEH'04), 9-11 septembre 2004, Mons, Belgique, 2004, (à paraitre).
10. G. Trouessin, G. "L'évolution des normes de sécurité vers plus d'auditabilité des systèmes d'information". Colloque AIM à l'HEGP : « Présent et avenir des systèmes d'information et de communication hospitaliers », 23-24 mai 2002 Springer-Verlag.
11. AFNOR, document de normalisation française, Fascicule de Documentation FD S 97-560.
12. Loi 94-43 du 18 janvier 1994 relative à la santé publique et à la protection sociale, article 8.
13. Circulaire DH/PMSI n° 303 du 24 juillet 1989 relative à la généralisation du Programme de médicalisation (BOMS n° 89/46), Ministère de l'emploi et de la solidarité, France.

14. Ordonnance n° 96-346 du 24 avril 1996 portant réforme de "l'hospitalisation publique et privée des systèmes d'information et à l'organisation médicale dans les hôpitaux publics".
15. Circulaire n° 153 du 9 mars 1998 relative à la généralisation dans les établissements de santé sous dotation globale et ayant une activité de soins de suite ou de réadaptation d'un recueil de RHS, ministère de l'emploi et de la solidarité, France.
16. Loi 91-748 du 31 juillet 1991 portant réforme hospitalière.
17. Décret n° 94-666 du 27 juillet 1994 relatif aux systèmes d'information médicale et l'analyse de l'activité des établissements de santé publics et privés sous compétence tarifaire de l'État, modifié par le décret n° 98-63 du 2 février 1998.
18. Arrêté du 29 juillet 1998 relatif au recueil et traitement des données d'activité médicale par les établissements de santé publics et privés financés par dotation globale visées à l'article L. 710-16-1 du même code et à la transmission aux agences régionales de l'hospitalisation et à l'État d'informations issues de ce traitement (JO, 26 août 1998).
19. D. Denning et P. Denning, "Data Security". ACM Computer Survey, vol. 11, n° 3, septembre 1979, ACM Press, ISBN : 0360-0300, pp. 227-249.
20. F. Cuppens, "Sécurité des bases de données", in Sécurité des réseaux et des systèmes répartis, (Yves Deswarte & Ludovic Mé, eds), Traité IC2, Hermès, ISBN : 02-7462-0770-2, 264 pp, octobre 2003.
21. A. Abou El Kalam, "Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales", Thèse de doctorat, Institut National Polytechnique de Toulouse, 183 pp., 04 décembre 2003 (Rapport LAAS 03578).
22. C. Quantin, H. Bouzelat, FA. Allaert, AM. Benhamiche, J. Faivre et L. Dusserre, "How to ensure data security of an epidemiological follow-up : quality assessment of an anonymous record linkage procedure", International Journal of Medical Informatics 49 (1998) 117-122.

GQ2
un preuve zero-knowledge de connaissance de la factorisation complément essentiel à RSA

Sophie Boutiton[1], François Daudé[2], and Louis Guillou[3]

[1] Doctorante à France Télécom R&D
[2] Ingénieur de recherche à France Télécom R&D
[3] Expert Emérite à France Télécom R&D,
4 rue du Clos de Courtel
35 512 Cesson Sévigné, France

Résumé Nous présentons un protocole d'authentification et de signature électronique inventé par L. Guillou et J.J. Quisquater et appelé GQ2. Nous caractérisons de manière riguoureuse les conditions suffisantes pour que le protocole GQ2 soit une preuve zero-knowledge de connaissance de la factorisation et complétons les éléments de preuve apportés dans [8]. Nous évaluons alors ses performances en le comparant aux principaux protocoles d'authentification existants.
Nous montrons ainsi qu'il offre le meilleur compromis actuel en terme de sécurité et de performance. En particulier, nous montrons qu'à sécurité égale (c'est à dire pour un même module RSA-1024), l'authentification GQ2 est 40 fois plus rapide que l'authentification RSA.

1 Introduction

La découverte, faite en 1977 par Rivest Shamir et Adleman, de l'algorithme RSA [15] a révolutionné le monde de la sécurité, en proposant une réponse efficace au problème de l'authentification. Il faut attendre 1985 pour voir Golwasser, Micali et Rackoff [3] introduire le concept de «zero-knowledge» et apporter une réponse nouvelle au problème de l'authentification. Depuis, de nombreux algorithmes de type «zero-knowledge» ont été proposés dans la litterature scientifique, et dans le but de les comparer, deux critères de base peuvent être pris en considération :

- **la sécurité :** Dans ce cas, on cherche à établir une relation logique entre la sécurité de l'algorithme et la difficulté pour résoudre un certain problème mathématique. Cela suppose au préalable de disposer d'une formalisation mathématique précise de la notion de sécurité de l'algorithme.
- **La performance :** Dans ce cas, on peut la décliner par l'intermédiaire de différentes caractéristiques : le temps de calcul des différentes ressources, le nombre de bits échangés entre les différentes ressources, ou encore la taille mémoire utilisée par chaque ressource.

Dans ce papier, nous nous proposons d'étudier la sécurité et les performances du mécanisme zero-knowledge GQ2 et de montrer qu'il réalise le meilleur compromis actuel de ces deux critères.

1.1 Résultats précédents

En 1987, A.Fiat et A.Shamir [2] propose le premier schéma d'authentification zero-knowledge. Sa sécurité est basée sur le problème de la factorisation, plus précisément le problème du calcul d'une racine carrée modulo n (où n désigne un entier de Blum). D'un point de vue performance, sa faiblesse réside dans le fait que le protocole exige un très grand nombre de bits échangés entre le prouveur et le vérifieur.

En 1988, L.Guillou et JJ.Quisquater [6] propose un protocole appelé GQ1. Ce protocole réduit considérablement le volume de données échangées. Ce protocole est prouvée zero-knowledge, mais sa sécurité n'est plus directement liée à la factorisation mais seulement au problème RSA.

En 1990, H.Ong et C.P.Schnorr [12] proposent un protocole qui, comme GQ1, réduit le volume de données échangées. Sa sécurité est basée sur le problème difficile du calcul des racines carrées successives. Cependant, bien que prouvé sûr contre les attaques actives dans le cas des entiers de Blum [18] puis dans le cas d'un module quelconque [17], il n'est pas «zero-knowledge» dans le sens donnée par [3].

En 2001, L.Guillou et JJ.Quisquater [8] présentent un nouveau protocole GQ2. Il accroit les performances du schéma de Ong-Schnorr et de GQ1. Quand à sa sécurité, des éléments de preuve sont apportés concernant son caractère «zero-knowledge» et sa relation avec le problème de la factorisation.

En parallèle, grâce en particulier aux travaux de O.Goldreich [20], une formalisation mathématique précise du concept de protocole «zero-knowledge» a été effectuée. Cette formalisation fournit le matériel mathématique nécessaire pour établir les preuves de sécurité des protocoles zero-knowledge.

1.2 Nos résultats

En se basant sur le formalisme mathématique introduit par [20] et précisé par [14], nous caractérisons rigoureusement les conditions suffisantes pour que le protocole GQ2 soit une preuve zero-knowledge de connaissance de la factorisation au sens de [3].

En particulier, nous montrons que s'il existe une attaque permettant de tromper le vérifieur avec une probabilité non négligeable, alors il existe un algorithme de factorisation en temps polynomial.

Par ailleurs, nous comparons les performances du protocole GQ2 avec différents protocoles de sécurité selon plusieurs critères : la complexité de calcul du prouveur puis du vérifieur, la taille mémoire nécessaire au prouveur pour le stockage de sa clé privée et le volume des données qui transitent entre le prouveur et le vérifieur. Nous montrons ainsi que GQ2 offre des performances bien

supérieures aux principaux mécanismes d'authentification, exception faite du volume de stockage de la clé privée. En particulier, nous établissons que pour un même module RSA-1024, l'authentification GQ2 est 40 fois plus rapide que celle obtenue à partir du RSA.

2 Présentation du protocole d'authentification GQ2

2.1 Elaboration des bi-clés GQ2

Cette description est conforme aux spécifications de GQ2 présentes dans la norme ISO/IEC 9798-5 [9]. GQ2 fait appel aux trois paramètres suivants :

- Un paramètre noté k appelé **paramètre de sécurité**.
- Un paramètre noté m appelé **paramètre de multiplicité**.
- Un paramètre noté v appelé **exposant de vérification** et défini par $v = 2^{k+1}$.

La construction d'une bi-clé GQ2 s'effectue selon les étapes suivantes :

- On choisit aléatoirement deux nombres premiers p_1 et p_2 congrus à 3 modulo 4
- On calcule le module n égal au produit de p_1 par p_2
- La clé publique GQ2 se compose du module n et de m nombres publics notés $(G_1, .., G_m)$, chaque G_i étant le carré d'un petit nombre premier noté g_i et appelé **nombre de base**.
- La clé publique GQ2 doit vérifier la propriété suivante :
  Pour au moins un nombre de base noté g, nous avons : $\left(\frac{g}{p_1}\right) = -\left(\frac{g}{p_2}\right)$ où $\left(\frac{\cdot}{\cdot}\right)$ désigne le symbole de Jacobi ([11], §2.4)
- La clé privée GQ2 se compose des nombres premiers p_1,p_2 et de m nombres secrets notés $(Q_1, .., Q_m)$ reliés aux nombres publics par la relation suivante :

$$\forall i \in \{1, .., m\}, G_i Q_i^v = 1 \bmod n$$

2.2 Le protocole d'authentification GQ2

Le protocole GQ2 s'effectue entre un prouveur et un vérifieur. Le vérifieur connaît la clé publique GQ2 $(n, G_1, .., G_m)$ et le prouveur connaît la clé privée GQ2 $(p_1, p_2, Q_1, .., Q_m)$. Ils possèdent en commun l'exposant de vérification $v = 2^{k+1}$ et le paramètre de multiplicité m.

Le prouveur GQ2 réalise alors systématiquement les étapes suivantes :

1. Sélection d'un nombre aléatoire positif et inférieur à n, noté r,
2. Calcul de $W = r^v \bmod n$ appelé **témoin** et noté W,
3. En réponse à un défi émis par le vérifieur, consistant en m nombres aléatoires de k-bits notés $(d_1.., d_m)$, calcul du nombre $D = r.Q_1^{d_1}..Q_m^{d_m}$ appelé **réponse** et noté D,

4. Effacement du nombre aléatoire r.

Le vérifieur réalise systématiquement les étapes suivantes :

1. Réception de la part du prouveur du témoin W,
2. Sélection de m nombres aléatoires de k-bits notés $(d_1, .., d_m)$,
3. En réponse à un nombre D émis par le prouveur, calcul du nombre $W' = D^v.G_1^{d_1}...G_m^{d_m} \bmod n$ et vérification de la condition $W' = W \wedge W' \neq 0$
4. Si la condition précédente est vérifiée, acceptation du prouveur.

Lorsque ce protocole est effectué un nombre t de fois, on parle du protocole **GQ2 itéré à l'ordre** t. Conformément à la méthode décrite en [11], note 10.30, le protocole d'authentification peut être transformé en mécanisme de signature électronique.

3 Analyse de sécurité du protocole GQ2

3.1 Rappels de concepts mathématiques utiles

Cette section rappelle les principales notions mathématiques nécessaires à l'énoncé des preuves de sécurité. On définit l'ensemble des mots $E^* = \{0,1\}^*$, Pour tout élément x de E^*, on désigne par $|x|$ la longueur du mot x, définie par $|x| = inf\{n/x \in \{0,1\}^n\}$.

On suppose connues les notions de machines de Turing, de complexité d'une machine de Turing et de machine de Turing probabiliste à temps polynomial (que l'on abrège par MT). On se reportera à [14] pour les définitions mathématiques exactes. On utilise par la suite la notation $M_{w_m}(x)$ pour désigner le mot calculé par la MT M à partir du mot x et de l'aléa (on dit encore ruban) w_m. On rappelle , toutefois, les notions suivantes (elles aussi inspirées de [14]) :

- Deux variables aléatoires (v.a.) $\{U(x)\}$ et $\{V(x)\}$ paramétrées par un ensemble de mots E^* (c'est à dire des v.a. à valeurs dans E^*) sont dites **parfaitement indistinguables** si

$$\forall x \in E^*, U(x) = V(x)$$

  Intuitivement cela signifie que pour tout mot x, on ne peut distinguer deux échantillons de U et de V quelque soit leur taille et la puissance de calcul disponible.
- Soit $P(x,y)$ un **prédicat à deux variables** calculable en temps polynomial en $|x|$. On dit que y est **valide** s'il existe P tel que $P(x,y)$. Pour un x fixé, une donnée y vérifiant $P(x,y)$ est appelé un **témoin** de y.
- Une fonction f de E^* dans R est dite **négligeable** si :

$$\forall c \in \mathbb{N}^*, \exists n_0 \in \mathbb{N}/\forall |G| \geq n_0, f(G) < \frac{1}{|G|^c}$$

Mathématiquement, le protocole GQ2 appartient à la classe des protocoles interactifs à 3 passes, dont on donne la définition ci-dessous :

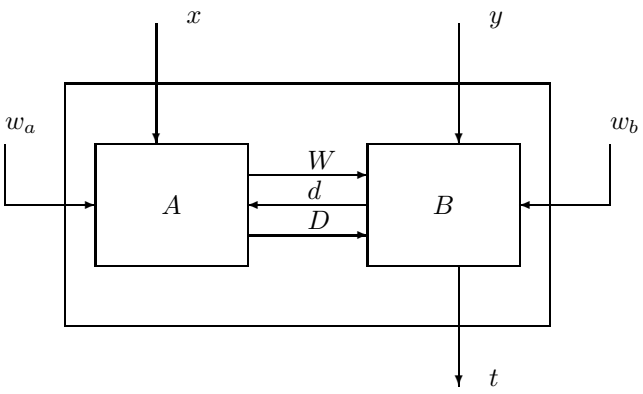
Definition 1. *On appelle* ***protocole interactif à 3 passes*** *le couple, noté* $\langle A(x), B(y)\rangle$*, constitué de deux machines de Turing probabiliste à temps polynomial* A *et* B *et vérifiant la propriété suivante :*

$$\forall(x, y, w_a, w_b)\ , \exists(W, d, D, t)\ /$$
$$A_{w_a}(x, d) = (W, D) \quad et \quad B_{w_b}(y, W, D) = (d, t) \quad et \quad t \in \{0, 1\}$$

avec les notions suivantes associées :
- A *désigne le prouveur et* B *le vérifieur,*
- w_a *et* w_b *désignent les aléas (ou rubans aléatoires) utilisés par* A *et* B *respectivement,*
- (W, d, D) *désigne les données échangées entre* A *et* B *que l'on note encore* $VUE\langle A(x), B(y)\rangle$*,*
- t *désigne la réponse du protocole, que l'on note* $\langle A(x), B(y)\rangle$*,*

Lorsque $\langle A(x), B(y)\rangle = 1$ *on dit que le prouveur* A *est accepté par le vérifieur* B*.*

Schématiquement, un protocole interactif à 3 passes se représente de la manière suivante :



Pour prouver la sécurité du protocole GQ2, nous suivons l'approche élaborée par Goldwasser Micali Rackoff [3], consistant à montrer qu'un protocole interactif à trois passes possède les trois propriétés suivantes :

- **Completeness** : Si le prouveur connait la clé GQ2 (c'est à dire la factorisation du module) alors le vérifieur doit l'accepter avec une probabilité écrasante,

- **Soundness** : Si le prouveur ne dispose pas de clé GQ2 (c'est à dire ne connaît pas la factorisation du module) alors quelque soit la stratégie choisie par le prouveur, le vérifieur doit le rejeter avec une probabilité écrasante.
- **Zero-knowledge** : Quelque soit la stratégie choisie par le vérifieur et quelque soit la quantité d'information qu'il récupère en interagissant avec un prouveur , le vérifieur n'arrivera jamais à récupérer le secret de la factorisation du module détenu par le prouveur.

Mathématiquement, cela se traduit de la manière suivante :

Definition 2. *On dit qu'un protocole interactif $\langle A(x), B(y)\rangle$ est une* ***preuve zero-knowledge de connaissance d'un prédicat P(Q,G)*** *si les trois propriétés suivantes sont respectées :*

1. *Propriété de* ***«consistance»*** *(completeness) :*
   Pour tout G valide, si A connaît un témoin de G, alors A convainc B soit formellement :

$$\forall (G,Q), P(Q,G) \Rightarrow Pr_{w_a,w_b}[\langle A_{w_a}(Q), B_{w_b}(G)\rangle] \geq 1 - e^{-|G|}$$

2. *Propriété de* ***«significativité»*** *(soundness) :*
   Pour tout G valide, si un prouveur $\tilde{A}$ est capable de convaincre B avec une probabilité non négligeable alors il doit nécessairement connaître un témoin de G, soit formellement :

$$\forall c \in \mathbb{N}^*, \exists M \text{ une } MT / \forall \tilde{A} \text{ une } MT, \exists n_0 \in \mathbb{N}, \forall G / |G| > n_0,$$

$$Pr_{w_{\tilde{a}},w_b}[\langle \tilde{A}_{w_{\tilde{a}}}(Q), B_{w_b}(G)\rangle] \geq \frac{1}{|G|^c} \Rightarrow Pr_{w_m}[P(M_{w_m}(G), G)] \geq 1 - e^{-|G|}$$

3. *Propriété de* ***«sans connaissance»*** *(zero-knowledge) :*
   Pour tout G valide, toute information obtenue par un vérifieur $\tilde{B}$ au cours de l'exécution du protocole avec A, peut être obtenue par $\tilde{B}$ à l'aide d'une simulation sans interaction avec A, soit formellement :

$$\forall \tilde{B} \text{ une } MT, \exists M \text{ une } MT \ / \ \forall (G,Q) \ / \ P(G,Q) \ \forall (a,b,c)$$

$$Pr_{w_a,w_{\tilde{b}}}[VUE\langle A_{w_a}(Q), \tilde{B}_{w_{\tilde{b}}}(G)\rangle = (a,b,c)] = Pr_{w_m}[P(M_{w_m}(G) = (a,b,c)]$$

3.2 GQ2 est une preuve zero-knowledge de connaissance de la factorisation

Pour établir que GQ2 est une preuve zero-knowledge de connaissance de la factorisation, nous devons démontrer que

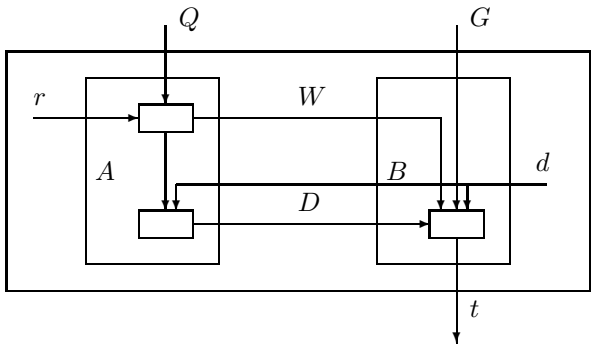
- GQ2 est un protocole interactif à 3 passes au sens de la définition 1,
- Le prédicat associé à GQ2 est équivalent au prédicat associée à la factorisation et défini par $F(p_1, p_2, n) \equiv (n = p_1 * p_2)$,
- GQ2, comme protocole interactif, respecte les propriétés «completeness», «soundness» et «zero-knowledge» au sens de la définition 2.

GQ2 est un protocole interactif à 3 passes

En notant : $G = (n, g_1, .., g_m)$, $Q = (p_1, p_2, Q_1, .., Q_m)$ et $d = (d_1, .., d_m)$ on peut définir les deux MT :

- $A_r(Q, d) = (W, D)$ où $W = r^v \bmod n$ et $D = r.Q_1^{d_1}...Q_m^{d_m} \bmod n$
- $B_d(G, W, D) = (d, t)$ où $t = 1 \Leftrightarrow W = D^v.g_1^{2\times d_1}....g_m^{2\times d_m} \bmod n$

Le schéma ci-dessous permet de vérifier simplement que GQ2 est bien un protocole à 3 passes au sens de la définition 1 :



Le prédicat associé à GQ2 est équivalent au prédicat associé à la factorisation

Le prédicat associé à GQ2 est défini par l'expression suivante :

$$P(Q, G) \equiv \left(\wedge_{i:1..m} g_i^2 Q_i^v \equiv 1 \bmod n\right)$$

Il est clair que la connaissance de G et de la factorisation du module n permet de déterminer le témoin Q de G (algorithme classique d'extraction de racine v-ième). Réciproquement, sachant qu'il existe un nombre de base g tel que $\left(\frac{g}{p_1}\right) = -\left(\frac{g}{p_2}\right)$ en posant $X = g$ et $Y \equiv Q^{-v/2} \bmod n$, on vérifie que $X^2 \equiv Y^2 \bmod n$ mais que $X \neq \pm Y \bmod n$. On en déduit donc la factorisation de $n = gcd(X - Y, n) \times gcd(X + Y, n)$.

De fait, si GQ2 est une preuve zero-knowledge de la connaissance du prédicat $P(Q, G) \equiv \left(\wedge_{i:1..m} g_i^2 Q_i^v \equiv 1 \bmod n\right)$, il est aussi une preuve zero-knowledge de la connaissance de la factorisation du module n.

GQ2 respecte la propriété «completeness»

Par construction de la réponse D, le protocole GQ2 respecte trivialement la propriété «completeness».

GQ2 respecte la propriété «soundness»

Dans [8], on montre que la connaissance de deux triplets entrelacés

$\{(W,d,D),(W,e,C)/d \neq e\}$ permet d'en déduire la factorisation du module. Toutefois, pour en déduire que GQ2 respecte la propriété «soundness», il faut aussi s'assurer que l'on peut produire de tels triplets en un temps polynomial.

Conformément à la définition 2, il nous faut donc construire une MT notée M telle que si $\tilde{A}$ est une MT qui convainc B avec une probabilité non négligeable, c'est à dire formellement $Pr_{w_{\tilde{a}},d}[\langle \tilde{A}_{w_{\tilde{a}}}(G), B_d(G)\rangle] \geq \frac{1}{|G|^c}$ alors la MT M peut calculer un témoin de G c'est à dire formellement $Pr_{w_m}[P(M_{w_m}(G),G)] \geq 1 - e^{-|G|}$. La construction de M repose sur une première MT, notée M', paramétrée par $v = 2^{k+1}$ et m, et définie par :

Entrée : W, d, D, e, E, $G = (n, g_1, ..., g_m)$
$X \leftarrow \left(\frac{E}{D}\right)^{v/2} \bmod n$
$Y \leftarrow g_1^{d_1 - e_1} \times .. \times g_m^{d_m - e_m} \bmod n$
$Q^* \leftarrow NULL$
Si $gcd(X - Y, n) \neq 1, n$ **alors**
$p_1 \leftarrow gcd(X - Y, n)$
$p_2 \leftarrow gcd(X + Y, n)$
$(Q_1, .., Q_m) \leftarrow$ racine v-ième de $(g_1, .., g_m)$ sachant p_1, p_2
$Q^* \leftarrow (Q_1, .., Q_m)$
Fin si
Sortie : Q^*

La MT M , paramétrée par le vérifieur de $GQ2$ noté $B_d(G)$ et par le prouveur $\tilde{A}_{w_a}(G)$, se définit alors de la manière suivante :

Entrée : G, c
Pour i **de** 1 **à** $L = 32|G|^{4c}$ **faire :**
$w_a \leftarrow ()$;$d \leftarrow ()$;$e \leftarrow ()$ /* sélection aléatoire */
$Q \leftarrow NULL$
Si $\langle \tilde{A}_{w_a}(G), B_d(G)\rangle$ **et** $\langle \tilde{A}_{w_{\tilde{a}}}(G), B_e(G)\rangle$ et $e \neq d$ **alors**
$(W,d,D) \leftarrow VUE\langle \tilde{A}_{w_a}(G), B_d(G)\rangle$
$(W,e,E) \leftarrow VUE\langle \tilde{A}_{w_a}(G), B_e(G)\rangle$
$Q \leftarrow M'(W,d,D,e,E,G)$
si $Q \neq NULL$ **alors** sortir de la boucle POUR **Fin si**
Fin si
Fin Pour
Sorter : Q

M' étant une MT, il est clair que M est aussi une MT. Il faut évaluer la probabilité que $M(G)$ soit bien un témoin de G. Pour cela, il faut calculer la probabilité des deux tests SI-ALORS présent dans l'algorithme M. En adaptant un lemme probabiliste donné par [19], on montre que :

Lemma 1. *Soient $\tilde{A}$ et B deux MT. Si on suppose $log(|n|) = o(m.k)$ alors*

$$\forall c \in \mathbb{N}^*, \forall G/|G| > |n|, Pr_{w_{\tilde{a}},d}[\langle \tilde{A}_{w_{\tilde{a}}}(G), B_d(G)\rangle] \geq \frac{1}{|G|^c} \Rightarrow$$
$$Pr_{w_{\tilde{a}},d,e}[\langle \tilde{A}_{w_{\tilde{a}}}(G), B_d(G)\rangle, \langle \tilde{A}_{w_{\tilde{a}}}(G), B_e(G)\rangle, d \neq e] \geq \frac{1}{16|G|^{3c}}$$

Intuitivement, cela signifie que si un prouveur $\tilde{A}$ ne connaissant pas une clé privée GQ2 peut être accepté par un vérifieur B avec une probabilité non négligeable, alors la probabilité d'observer des triplets de la forme

$$\{(W, d, D), (W, e, C)/d \neq e\}$$

dans les échanges entre $\tilde{A}$ et B est, elle aussi, non négligeable.

Par ailleurs, on dispose du lemme suivant qui prolonge le résultat obtenu en [8] :

Lemma 2. *Si A et B désignent respectivement le prouveur et le vérifieur du protocole GQ2 et s'il existe deux triplets $(W, d, D) = VUE\langle \tilde{A}_{w_a}(G), B_d(G)\rangle$ et $(W, e, E) = VUE\langle \tilde{A}_{w_a}(G), B_e(G)\rangle$ tels que $d \neq e$ alors il existe deux entiers X et Y tels que $Pr_{w_{\tilde{a}},d,e}[gcd(X - Y, n) \neq 1, n] = \frac{1}{2}$*

En effet, par définition d'une clé GQ2, il existe $i \in \{1, .., m\}$ tel que $\left(\frac{g_i}{p_1}\right) = -\left(\frac{g_i}{p_2}\right)$. Avec les notations du lemme, on pose $X = \left(\frac{E}{D}\right)^{v/2} \bmod n$ et $Y = g_1^{d_1 - e_1} \times .. \times g_m^{d_m - e_m} \bmod n$. par construction $X^2 = Y^2 \bmod n$ mais $X \neq \pm Y \bmod n$.

Pour conclure, la probabilité que $M(G)$ soit un témoin de G est donc :

$$Pr_{w_{\tilde{a}},d,e}[P(M(G), G)] \geq 1 - \left(1 - \frac{1}{2 \times 16|G|^{3c}}\right)^{32|G|^{4c}} \geq 1 - e^{-|G|}$$

Ainsi, si les clés GQ2 respectent les conditions du paragraphe 2.1 et si les paramètres k et m de GQ2 vérifient la condition «$log(|n|) = o(m.k)$» , alors le protocole GQ2 possède la propriété «soundness».

En adaptant la preuve au protocole GQ2 itéré à l'ordre t, il n'est pas difficile de corriger l'hypothèse sur les paramètres k,t et m et montrer qu'avec la condition «$log(|n|) = o(t.m.k)$» le protocole GQ2 itéré à l'odre t possède encore la propriété «soundness».

La conséquence est que s'il existe une attaque permettant de tromper le vérifieur avec une probabilité de réussite supérieure à 2^{-tkm}, alors il existe un algorithme de factorisation du module n en temps polynomial.

GQ2 respecte la propriété «zero-knowledge»

Il s'agit de montrer que l'observation des échanges entre un vérifieur malhonnête

et un prouveur A connaissant le témoin Q de G apporte la même information que les échanges simulés par $\tilde{B}$ à l'aide d'une MT $M(G)$ ne connaissant pas le témoin de G.
Mathématiquement, il faut donc construire une MT $M(G)$ dont la loi de probabilité est indistinguable de la variable aléatoire $VUE\langle A_r(Q), \tilde{B}_{w_d}(G)\rangle$

Le tableau ci-dessous indique le mode de génération des deux v.a. :

v.a. $VUE\langle A_r(Q), \tilde{B}_{w_b}(G)\rangle$:	v.a. M(G) :
	répéter
$r \leftarrow \{0,..,n-1\}()$ v.a. de loi uniforme	$d_0 \leftarrow \{0,..,2^{km}-1\}()$ v.a. de loi uniforme
$W \leftarrow r^v \bmod n$	$d_1 = (d_{11},...,d_{1m}) \leftarrow \{0,..,2^{km}-1\}()$ v.a. de loi fixée par $\tilde{B}$
$d_0 \leftarrow \{0,..,2^{km}-1\}()$ v.a. de loi fixée par $\tilde{B}$	$D \leftarrow \{0,..,n-1\}()$ v.a. de loi uniforme
$d_1 = (d_{11},...,d_{1m}) \leftarrow d(d_0, W)$: Fonction d fixée par $\tilde{B}$	$W \leftarrow D^v \times G_1^{d_{11}} \times .. \times G_m^{d_{1m}} \bmod n$
$D \leftarrow r \times Q_1^{d_{11}} \times .. \times Q_m^{d_{1m}} \bmod n$	**Jusqu'à** $d_1 = d(d_0, W)$
Renvoie (W, d_1, D)	**Renvoie** (W, d_1, D)
La vue $VUE\langle A_r(Q), \tilde{B}_{w_b}(G)\rangle$ est fonction des v.a. r et d_0 indépendantes.	La v.a. $M(G)$ est fonction des v.a. d_0, d_1 et D indépendantes Il est clair que : Si 2^{km} est polynomial en n alors l'algorithme définissant $M(G)$ est une MT
Remarque : dans les deux cas, la loi de la v.a. d_0 et la fonction d modélisent en fait la caractère «malhonnête» de $\tilde{B}$	

On montre alors que les deux lois de probabilité $VUE\langle A_r(Q), \tilde{B}_{w_d}(G)\rangle$ et de $M(G)$ sont égales. Plus précisément, on a :

$$Pr_{d_0,d_1,D}[P(M_{d_0,d_1,D}(G) = (a,b,c)] = \begin{cases} 0 \text{ si } c^v G_1^{b_1} \times .. \times G_m^{b_m} \neq a \bmod n, \\ \frac{1}{n} Pr_{d_0}[d(d_0,a) = b] \text{ sinon} \end{cases}$$

$$Pr_{d_0,d_1,D}[P(M_{d_0,d_1,D}(G) = (a,b,c)] = Pr_{r,d_0}[VUE\langle A_r(Q), B_{d_0}(G) = (a,b,c)\rangle]$$

Ainsi, si les clés GQ2 respectent les conditions du paragraphe 2.1 et si les paramètres k et m de GQ2 vérifient la condition «2^{km} varie polynomialement en |n|» alors le protocole GQ2 possède la propriété «zero-knowledge».

En adaptant la preuve au protocole GQ2 itéré à l'ordre t, il n'est pas difficile de corriger la condition sur les paramètres k,t et m et montrer qu'avec la condition «$t \times 2^{km}$ varie polynomialement en |n|» le protocole GQ2 itéré à l'ordre t possède encore la propriété «zero-knowledge».

3.3 Conclusion sur la sécurité du protocole GQ2

En supposant que les clés GQ2 respectent les conditions énoncées dans le paragraphe 2.1, on distingue alors deux cas selon les paramètres k,t et m du protocole GQ2 :

- Cas du protocole GQ2 non itéré (t=1) :
  GQ2 possède trivialement la propriété de completeness.
  Sous l'hypothèse $log(|n|) = o(m.k)$, GQ2 possède la propriété de «soundness».
  Sous l'hypothèse «2^{km} varie polynomialement en $|n|$», GQ2 possède la propriété de «zero-knowledge».
  Toutefois, les deux conditions sur les paramètres k et m deviennent contradictoires. Le protocole ne peut respecter à la fois les deux conditions.

- Protocole GQ2 itéré (à l'ordre t) :
  GQ2 possède trivialement la propriété de completeness.
  Sous l'hypothèse $log(|n|) = o(t.m.k)$, GQ2 possède la propriété de «soundness».
  Sous l'hypothèse $t \times 2^{km}$ varie polynomialement en $|n|$, GQ2 possède la propriété de «zero-knowledge».
  En conséquence, si t et 2^{km} varient polynomialement en $|n|$, le protocole GQ2 itéré à l'ordre t respecte les trois propriétés requises et définit bien une preuve «zero-knowledge» de connaissance de la factorisation du module n.

4 Analyse des performances du protocole GQ2

4.1 Comparaison entre les différents protocoles d'authentification

Nous avons comparé les performances du protocole GQ2 aux principaux protocoles d'authentification existants :

- Protocole de Feige Fiat Shamir [2][1],
- Protocole de Schnorr [16],
- Protocole GQ1 [6],
- Protocole GPS (Mode 1) [5],[4],
- Protocole d'authentification RSA unilatéral et mutuel [9].

Les spécifications détaillées de ces différents protocoles sont extraites de la norme ISO/IEC 9798-5 [9]. Les critères de comparaison retenus ont été les suivants :

- CM : Complexité de communication entre le vérifieur et le prouveur évaluée en Kbit
- CPC : Complexité de calcul du prouveur évaluée en nombre de multiplications modulaires
- CPV : Complexité de calcul du vérifieur évaluée en nombre de multiplications modulaires

– CS : Stockage requis par le prouveur évalué en Kbit

Afin de se ramener à une mesure unique pour CPC et CPV, nous avons fait les hypothèses suivantes concernant l'évaluation des opérations arithmétiques en nombre de multiplications modulaires :

– Le calcul de $A^2 \bmod C$, dans l'hypothèse où $|A| \sim |C|$, est évalué à 0,75 fois une multiplication modulo C
– Le calcul de $A^B \bmod C$ est évalué à |B|-1 carrés modulo C et $HW(B) - 1$ multiplications [4] modulo C.
– Le calcul de $A_1^{B_1} \times ... \times A_x^{B_x} \bmod C$ est évalué à $max(|B_1|, .., |B_x|) - 1$ carrés modulo C et $HW(B_1) + ... + HW(B_x) - 1$ multiplications modulo C par A_i.

Le tableau ci-dessous résume les performances comparées des différents protocoles d'authentification, pour chaque critère et en considérant le cas d'un module [5] de 1024 bits pour l'ensemble des protocoles. Les paramètres des différents protocoles sont ceux spécifiés dans l'annexe C.4.3 de la norme ISO/IEC 9897-5 [9].

	CS(K bits)	CPC	CPV	CM(K bits)
Fiat Shamir	5,00	11,00	11,00	8,00
GQ1	2,00	33,50	21,50	2,00
GQ2	5,50	7,75	3,75	2,00
Schnorr	2,31	200,00	208,00	1,17
GPS	1,16	192,00	200,00	1,27
RSA Unilateral Authentication	2,50	320,00	13,00	1,84
RSA Mutual Authentication	2,50	333,00	333,00	2,42

En dépit d'une étape supplémentaire dans le protocole (résultant du caractère zéro-knowledge), GQ2 réduit considérablement le temps de calculs par rapport à RSA. Comme le montre le tableau ci-dessus, pour un module de 1024 bits, le ratio RSA/GQ2 est de l'ordre de 40 pour le prouveur et de l'ordre de 3,5 pour le vérifieur.

Références

1. U.Feige, A.Fiat et A.Shamir, Zero knowledge proofs of identity, Crypto'89, *Lecture Notes in Computer Sciences 435*, pp. 526-544, 1990, Springer.
2. A.Fiat et A.Shamir, How to prove yourself : practical solution of identification and signature problems, Crypto'86, *Lecture Notes in Computer Sciences 263*, pp. 186-194, 1987, Springer.
3. S.Goldwasser, S.Micali et C.Rackoff, The Knowledge Complexity of Interactive Proof Systems, *SIAM journal of computing*, vol.18, pp. 186-208, 1989.

[4] $HW(B_1)$ représente le poids de Hamming de x
[5] ou, le nombre premier dans le cas de Schnorr

4. G.Poupard et J.Stern, Security Analysis of a practical " on the fly " Authentication and Signature generation, Eurocrypt'98, *Lecture Notes in Computer Sciences 1403*, pp. 422-436, 1998, Springer.
5. M.Girault, Self-Certified Public Keys, Eurocrypt'91, *Lecture Notes in Computer Sciences 36*, pp. 437-451, 2001, Springer.
6. L.Guillou et JJ.Quisquater, A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory, Eurocrypt'88, *Lecture Notes in Computer Sciences 330*, pp. 123-128, 1988, Springer.
7. L.Guillou et JJ.Quisquater, How to explain zero-knowledge protocols to yours children, Crypto'89, *Lecture Notes in Computer Sciences 435*, pp. 628-631, 1990, Springer.
8. L.Guillou, JJ.Quisquater et M.Ugon, Cryptographic authentication protocols for smarts cards, *Computer Network Magazine*, vol. 36, pp. 437-451, 2001.
9. , ISO/IEC-9798-5, Information technology - Security techniques . Entity authentication. Part 5 : Mechanisms using zero-knowledge techniques. Publication en cours.
10. ISO/IEC-14888-2, Information technology - Security techniques. Digital signature appendix. Part 2 : Integer factorization based mechanisms. Publication en cours.
11. A.Menezes, P. Van Oorschot et S.Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
12. H.Ong et C.P.Schnorr, Fast Signature generation with a Fiat-Shamir-like scheme, Eurocrypt'90, *Lecture Notes in Computer Sciences 473*, pp. 432-440, 1991, Springer.
13. D.Pointcheval, Les preuves de connaissance et leurs preuves de sécurité, thèse de doctorat, 1996.
14. G. Poupard, Authentification d'entité, de messages et de clés cryptographiques : théorie et pratique, thèse de doctorat, 2000.
15. R.Rivest, A.Shamir et L.Adleman, A Method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 21, pp. 120-126, 1978.
16. C.P.Schnorr, Efficent Identification and Signatures for Smarts Cards, Crypto'89, *Lecture Notes in Computer Sciences 435*, pp. 235-251, 1990, Springer.
17. C.P.Schnorr, Security of 2^t-Root Identification and Signatures, Crypto'97, *Lecture Notes in Computer Sciences 1294* pp. 540, 1997.
18. V.Shoup, On the Security of a practical identification scheme, Eurocrypt'96, *Lecture Notes in Computer Sciences 1070* pp. 340-353, 1996, Springer.
19. D.Pointcheval et J.Stern, Security Proofs for Signature Schemes, Eurocrypt'96, *Lecture Notes in Computer Sciences 1070*, pp. 387-398, 1996, Springer.
20. O.Goldreich, Zero-knowledge twenty years after its invention, *U.S.C. Computer Science Department*, Technical Report 2002/186, pp. 387-398, 2002.

Cryptographie et reverse engineering en environnement Win32

Kostya Kortchinsky

Responsable du CERT RENATER
kostya.kortchinsky@renater.fr

Résumé De façon illustrée et didactique, l'article mettra l'accent sur des résultats que l'on peut obtenir par application de l'analyse et du reverse-engineering dans le contexte particulier de la cryptographie en environnement Win32, sans toutefois s'attarder trop sur les techniques de reverse engineering ni la théorie cryptographique.

1 Introduction

L'un des inconvénients majeurs de logiciels à vocation cryptographique en environnement Windows est sans doute l'absence quasi totale de visibilité de l'utilisateur vis à vis de ce qui lui est vendu. Comment s'assurer que ce que l'on achète fait correctement ce qu'il est sensé faire, et ni plus, ni moins ? Il existe une alternative à une confiance aveugle en ce que nous racontent les éditeurs, et qui permettra à toute personne un tant soit peu motivée et techniquement compétente, de se faire une idée par soi-même : le reverse-engineering. Compliqué en temps normal, l'analyse et la rétroconception gagnent une nouvelle dimension lorsque l'on se trouve confronté à des algorithmes imposants et complexes, mais les résultats sont souvent à la hauteur des efforts investis.

L'actualité fait écho des achèvements de certains dans ce domaine, avec l'exemple parlant du système de génération et de vérification de clés produits Microsoft. Utilisant un algorithme de signature sur courbes elliptiques, les parties privées de ses clés ont récemment été déterminées suite à un travail conséquent sur les binaires concernés.

2 Cryptos, ou pourquoi il ne faut pas croire ce que l'on vous raconte (sauf moi)

Cryptos [1] est un logiciel de chiffrement de fichier simple et ergonomique sous environnement Win32, utilisant en version 1.0 – d'après ses auteurs – un cryptage de type DES de 128 bits (!) pour la version française, jusqu'à 4096 bits (!!) sur la version internationale. Courant 2002, un fichier chiffré "testcryptage.rtf.CR1" (avec la version internationale) a été mis en ligne, et une somme de 1000 euros promise à toute personne arrivant à obtenir le fichier original.

2.1 Analyse

Une analyse sommaire du logiciel en question montre rapidement que :
- le type de chiffrement utilisé n'est en rien du DES (même de loin sous la pluie par temps de brouillard) ;
- l'espace des clés possible est ridiculement faible (100 clés au maximum pour la version française à cause d'une erreur de programmation) ;
- une attaque par texte clair avec une dizaine de caractères connus devrait mener à des résultats rapidement ;
- il parait peu probable que les clés soient uniques pour un même fichier chiffré.

Ce type de renseignements peut être obtenus sans recours à la décompilation, grâce à des manipulations simples susceptibles d'être effectuées dans le cadre d'une utilisation régulière du logiciel sur sa version d'évaluation.

A titre d'exemple, le fait qu'un même fichier, chiffré avec les clés " 12 ", " 123 ", " 1234 " et " 12345 ", donne 4 fichiers chiffrés en tout points identiques est plus que suspect. Quelques essais supplémentaires mettent en évidence que seuls les deux premiers chiffres de la clés sont pris en compte, d'où les 100 clés possibles dans la version françaises.

La version internationale introduit deux paramètres supplémentaires, de type petits entiers (16 bits), augmentant l'espace possible des clés.

Des essais sur des fichiers de petite taille, dans le cadre d'une utilisation légitime du logiciel, permettent de mettre en évidence l'usage d'une opération binaire simple de type XOR (ou-exclusif) pour le chiffrement des fichiers, accompagné d'une opération arithmétique simple sur les paramètres constituant la clé.

2.2 Déchiffrement

Le fichier fournit pour le challenge est de type Rich Text Format (RTF), comme le laisse supposer sa première extension. Ce type de document débute toujours par une séquence de caractères identique : `\rtf1\ansi()`.

En s'aidant des précédentes observations, et ayant à disposition le début du texte clair et la totalité du texte chiffré, l'implémentation d'un programme parcourant de façon exhaustive l'espace des clés ne posera pas de problème majeur.

2.3 Source

Fichier decryptos.c :

```
/*
 * gcc -Wall -O3 -o decryptos decryptos.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(int argc, char *argv[])
{
  unsigned int j;
  unsigned short int i, k, l, m;
  unsigned short int key, tmp;
  unsigned char encrypted_buffer[8],
    decrypted_buffer[8], key_buffer[8];
  unsigned char buffer[8], file_name[64];
  FILE *encrypted_file, *decrypted_file;

  if ((encrypted_file
      = fopen("testcryptage.rtf.CR1", "rb"))
      == NULL)
    exit(EXIT_FAILURE);

  /* Texte clair */
  memcpy(decrypted_buffer, "{\\rtf1\\ansi\\",
    sizeof(decrypted_buffer));
  /* Texte chiffré */
  if (fread(encrypted_buffer, 1,
      sizeof(encrypted_buffer), encrypted_file)
      != sizeof(encrypted_buffer))
    exit(EXIT_FAILURE);

  for (i = 0; i < sizeof(key_buffer); i++)
    key_buffer[i] = encrypted_buffer[i]
      ^ decrypted_buffer[i];

  /* Premier chiffre de la clé */
  for (i = '0'; i <= '9'; i++)
  {
    /* Premier petit entier */
    for (j = 0; j <= 0xffff; j++)
    {
      tmp = ((key_buffer[0] << 8) | i);
      tmp = (tmp + encrypted_buffer[0]) * j;
      k = (key_buffer[1] - (tmp >> 8) - 1) << 8;
      /* Second petit entier */
      for (l = 0; l <= 0x1ff; k++, l++)
      {
        key = tmp + k;
        for (m = 1; m < 8; m++)
        {
          if((unsigned char)(key >> 8)
```

```
              != key_buffer[m])
            break;
          key = (key + encrypted_buffer[m])
            * j + k;
        }
        /* Y a-t-il concordance ? */
        if (m != 8)
          continue;
        /* Oui, on déchiffre tout le fichier */
        printf("[!] Probable good key found: %04x %04x %04x\n",
          (key_buffer[0] << 8) | i, j, k);
        sprintf(file_name,
          "decryptos_%04x_%04x_%04x.rtf",
          (key_buffer[0] << 8) | i, j, k);
        /* Déchiffrement du fichier avec la clé probale trouvée */
        if ((decrypted_file
             = fopen(file_name, "w+b")) == NULL)
          return EXIT_FAILURE;
        fseek(encrypted_file, 0, SEEK_SET);
        key = (key_buffer[0] << 8) | i;
        while (fread(buffer, 1, 1,
          encrypted_file) > 0)
        {
          /* XOR et opération arithmétique */
          decrypted_buffer[0] = (key >> 8)
            ^ buffer[0];
          key = (key + buffer[0]) * j + k;
          fwrite(decrypted_buffer, 1, 1,
            decrypted_file);
        }
        fclose(decrypted_file);
      }
    }
  }

  fclose(encrypted_file);
  return EXIT_SUCCESS;
}
```

2.4 Résultats

Pour ce fichier, sensé être protégé par un chiffrement de type DES, les conséquences sont plutôt dramatiques. Exécution sur un Intel Pentium 4 à 2.26GHz :

```
kostya@icare:~/tools/cryptos$ time ./decryptos
[!] Probable good key found: 3130 0dfc 89a2
```

```
[!] Probable good key found: 3131 0dfc 7ba7
[!] Probable good key found: 3132 0dfc 6dac
[!] Probable good key found: 3133 0dfc 5fb1
[!] Probable good key found: 3134 0dfc 51b6
[!] Probable good key found: 3135 0dfc 43bb

real    0m2.254s
user    0m2.250s
sys     0m0.010s
```

Il nous faut moins de 3 secondes pour parcourir l'ensemble des clés, trouver 6 clés probables, ainsi que les fichiers déchiffrés associés.

```
Fichiers déchiffrés :
kostya@icare:~/tools/cryptos$ md5sum *.rtf
1c401e97695ad4b7bd48b0381f90986f  decryptos_3130_0dfc_89a2.rtf
1c401e97695ad4b7bd48b0381f90986f  decryptos_3131_0dfc_7ba7.rtf
e4b0521ee9e8e05e6c96bf68eea80c27  decryptos_3132_0dfc_6dac.rtf
e4b0521ee9e8e05e6c96bf68eea80c27  decryptos_3133_0dfc_5fb1.rtf
e4b0521ee9e8e05e6c96bf68eea80c27  decryptos_3134_0dfc_51b6.rtf
e4b0521ee9e8e05e6c96bf68eea80c27  decryptos_3135_0dfc_43bb.rtf
```

Le fichier original étant au format Rich Text, il est assez simple de vérifier la validité des fichiers déchiffrés avec nos clés probables. Les deux premières clés mènent à un fichier déchiffré globalement compréhensible, les 4 clés suivantes, au fichier original (pas d'erreur RTF).

3 ASProtect, un cas concret de faille d'implémentation de l'algorithme RSA

ASProtect est un système de protection logicielle d'applications, offrant de nombreuses fonctionnalités, parmi lesquelles le chiffrement et la compres-sion de l'application, l'ajout de contre-mesures variées compliquant débugage et désassemblage, ainsi qu'un système d'enregistrement de l'application utilisant un algorithme à clé publique.

Ce système d'enregistrement a connu de multiples évolutions au fil des versions du produit, l'une d'entre elles ayant été motivée par la sortie de générateurs de clés pour une trentaine de produits protégés avec ASProtect, dont ASPack et ASProtect eux-même. Cet événement laissait suggérer la présence d'une faille dans l'algorithme utilisé, du RSA, dont les paramètres avaient cependant une taille conséquente, 1024 bits. Les travaux effectués pour cet article l'ont été sur une version 1.1 de ASProtect qui pourra être mise à disposition du lecteur.

3.1 Processus de vérification d'une clé d'enregistrement

Une clé d'enregistrement valide pour une application protégée par ASProtect se présente généralement sous forme d'un fichier ".key", pouvant être importé au

Registre de Windows, y ajoutant une nouvelle valeur dont le nom est "Key" et dont le contenu est une chaîne binaire de 1024 bits (au plus) encodée en Base64.

```
Exemple de clé : REGEDIT4

[HKEY_CURRENT_USER\Software\Asprotect]
"Key"="rfk1ksv3o8WSSYZDmYZpjIFpmd8t5b38J
       Q/26lVoGLruotu/tYygUKwpSlPSkoYnc0
       Gj3o2vUiLQaBavC0SvG2xdnkYiEFK3hTT
       Mpgp92p+EAb46ibXK8Eoc78y8ajf/NXj0
       GGJMfV6AXQrUOYFaGB5S7YvkzTL9oyKLh
       rbmPRh="
```

Le traitement appliqué à cette chaîne par le squelette d'ASProtect peut être suivi grâce à un simple debugger de ring3 - OllyDbg [2] pour le citer, néanmoins les quelques trucs et astuces utilisés par ASProtect à l'encontre de ce type d'activité peuvent désorienter des débutants. Le reverse-engineering n'étant pas l'objet premier de cet article, je ne détaillerai pas le procédé mais le décrirai brièvement.

Une fois l'exécutable chargé sous OllyDbg, il nous faut modifier un peu UnhandledExceptionFilter dans kernel32.dll afin que les exceptions ne soient pas passées au debugger, forcer le saut après NtQueryInformationProcess fait l'affaire, puis lancer le programme à proprement parler. La première exception arrêtera l'exécution du code dans le squelette d'ASProtect une fois celui-ci déchiffré, une aubaine. Un breakpoint sur l'API RegQueryValueExA, quelques shift-F9, et nous atteignons la lecture de la valeur de la clé de Registre voulue. Des "breakpoint on access" sur les zones mémoires impliquées nous mènerons jusqu'à la procédure de décodage Base64, puis jusqu'à la première opération arithmétique effectuée sur une partie de la chaîne décodée.

L'instant semble être tout indiqué pour effectuer un dump de la plage mémoire dans laquelle nous évoluons : nous ne disposons pas du code du squelette d'ASProtect puisqu'il est déchiffré dynamiquement à l'exécution, et un deadlisting se révèlera indispensable à l'usage ; de plus il y a de grandes chances que les paramètres de l'algorithme de chiffrement soient présents à ce moment en mémoire. Nous suivons ensuite quelques "ret" en notant les adresses qui nous seront utiles pour le désassemblage, et regardant de temps à autre le contenu des buffers. Un oeil habitué discernera rapidement les paramètres importants, d'autant plus que l'auteur d'ASProtect nous facilite la tache en utilisant des buffers statiques. Un extrait du code (chargement sous IDA [3] du dump de la mémoire en tant que fichier binaire, l'offset peut varier) :

```
; modulo (n)
seg000:0092D0E7 mov     ecx,  offset byte_9326D4
; exposant public (e)
seg000:0092D0EC     mov edx,  offset byte_932654
; message (m)
seg000:0092D0F1     mov eax,  offset byte_9325D4
seg000:0092D0F6     call      sub_92C568
```

Les paramètres sont stockés sous forme d'un tableau de 128 octets, l'octet de poids faible en première position. Le résultat de la fonction est stocké dans le tableau contenant le message initial. Récupérés du dump de la mémoire, les paramètres sont les suivants :

```
m  = 0x183de6b6 868b22a3 fd32cde4 8bed521e
       185a8139 d40a5d80 5e7d4c62 18f47835
       ff376abc ccef1c4a f0cab589 3abe0184
       9fda7d0a a6cc3485 b7521022 469e5d6c
       1baf440b af1668d0 2252af8d dea34173
       278692d2 534a29ac 50a08cb5 bfdba2ee
       ba186855 eaf60f25 fcbde52d df996981
       8c698699 43864992 c5a3f7cb 9235f9ad

e  = 0x 11

n  = 0xeb1d4ead a4815f62 77519791 bffa8b4c
       0b872d1c 436515ab d9572b22 bf6a03fe
       cb4e5cc4 9af1ee35 c3134461 7a12106c
       30569052 9b9ce7f1 3ed2d37c d7034a3e
       dd096853 ec61243b ccac5a58 800b0330
       a4dd85e9 aa237f2f 2ae60ca0 49b1d277
       7b2e0c5f f51e0583 82a86c3e c12f7ab4
       16420227 72ff2a2d 3dba7047 25702199
```

L'utilisation d'un logiciel de calcul sur les grands nombres confirmera le contenu du message déchiffré à la sortie de la fonction.

```
Message déchiffré :

m' = 0x00000000 256453f4 b5e2e2d7 d775aa2c
       b87d6273 64f523a1 14030000 00000000
       00000000 00000000 00000000 00000000
       00000000 00007468 67696c69 77540a0d
       0a0d2d20 65736e65 63696c20 65746973
       202d0a0d 594b534e 49484354 524f4b20
       61797473 6f4b0000 00000000 0000c21c
       f68cadec cc1a8fd2 7072bd9d 839d1002
```

Une séance de débuggage complémentaire contribuera à déterminer le format d'une clé d'enregistrement déchiffrée (je ne rentrerai pas dans les détails) :

0 2 octets identifiant le champ suivant ;
2 24 octets contenant un hash 128 bits complété de caractères nuls ;
26 76 octets contenant le nom d'utilisateur et éventuellement des informations supplémentaires (type de licence, nombre de licences, ...) ;
102 2 octets identifiant le champ suivant ;

104 24 octets contenant un hash 160 bits (?) complété de caractères nuls. Le premier hash se révèlera être un hash MD5 du nom d'utilisateur, dont la validité sera vérifiée par le code de protection ajouté à l'application. La nature du second hash est plus floue pour le moment, on ne peut que constater qu'il est hashé de nouveau et comparé à un hash précalculé.

Voila pour le processus de vérification d'une clé d'enregistrement. Nous disposons à présent d'une quantité d'informations non négligeable pour s'attaquer au système d'enregistrement de ASProtect.

3.2 Processus de génération d'une clé d'enregistrement

Le logiciel ASProtect lui-même inclut un module de génération de clé d'enregistrement : une paire de clé est susceptible d'être générée pour chaque projet, et des clés d'enregistrement produites en fonction des informations fournies. Dans un premier temps, il nous faudra disposer d'un exécutable déprotégé susceptible de fournir une base à tout désassemblage ultérieur. Je vous invite pour cela à appliquer les méthodes décrites par Nicolas Brulez dans son article. Un rapide parcours de l'exécutable (désassemblé ou en hexadécimal) pourrait faire naître en vous un intérêt particulier pour la chaîne "*_rsaKeyGen@16*", typiquement un nom de fonction importée ou exportée. Grâce à un bon désassembleur (IDA Pro [3] pour ne citer que lui), il apparaît que cette fonction appartient à une portion de code chargée dynamiquement depuis l'une des ressources de l'application, TRSA, présente sous forme chiffrée dans l'exécutable.

Afin d'obtenir une image fonctionnelle de la bibliothèque RSAVR.dll et dont une partie est (c) 1986 Philip Zimmermann si l'on en croît deux chaînes de caractères, deux possibilités : extraire la ressource en question et appliquer la procédure de déchiffrement (fastidieux), ou bien dumper la mémoire une fois la DLL déchiffrée, l'extraire et la reconstruire (d'une facilité déconcertante).

La fonction rsaKeyGen est appelée dans ASProtect grâce aux quelques lignes suivantes :

```
; modulo (n)
seg000:0044BA82     push    offset byte_45EC50
; exposant privé (d)
seg000:0044BA87     push    offset byte_45EBD0
; exposant public (e)
seg000:0044BA8C     push    offset byte_45EB50
; 1024bits
seg000:0044BA91     push    400h
; _rsaKeyGen@16
seg000:0044BA96     call    [ebp+var_C]
```

Quelques minutes de reverse-engineering permettent de mettre en évidence l'algorithme de génération de la clé RSA :

1. initialisation du générateur de nombres aléatoires,
2. génération d'un nombre premier P de 512bits :

   - génération d'un nombre aléatoire P de 510bits,
   - mise à 1 des 511ème et 512ème bits (poids fort),
   - détermination d'un nombre premier probable supérieur (immédiatement) au nombre généré grâce à un test de Miller-Rabin [4],
3. génération d'un nombre premier Q de 512bits,
   - génération d'un nombre aléatoire Q de 510bits,
   - mise à 1 des 511ème et 512ème bits (poids fort),
   - détermination d'un nombre premier probable supérieur (immédiate-ment) au nombre généré grâce à un test de Miller-Rabin [4],
4. si P - Q (ou Q - P en fonction du signe) a une longueur strictement inférieure à 505bits, on retourne à l'étape 1.
5. génération de n, e (fixé à (1 « 4) + 1 soit 17), et d grâce aux nombres précédemment calculés.

3.3 Identification de la vulnérabilité

Il convient maintenant de détailler les portions de code pertinentes de la fonction de génération de la clé RSA. Initialisation du générateur de nombre aléatoires :

```
.text:10003358     push    0
.text:1000335A     call    _time
.text:1000335F     add     esp, 4
.text:10003362     mov     esi, eax
.text:10003364     call    ds:GetCurrentThreadId
.text:1000336A     add     esi, eax
.text:1000336C     call    ds:GetTickCount
.text:10003372     xor     esi, eax
; unsigned int
.text:10003374     push    esi
; srand((time(NULL) + GetCurrentThreadId())
  ^ GetTickCount());
.text:10003375     call    _srand
```

Génération d'un nombre aléatoire :

```
.text:1000138B     movsx   eax, bx
.text:1000138E     shr     eax, 5
.text:10001391     push    edi
; byteLength = bitLength / 32;
.text:10001392     mov     edi, eax
.text:10001394     neg     eax
.text:10001396     shl     eax, 5
; remainingBits = bitLength % 32;
.text:10001399     add     ebx, eax
.text:1000139B
```

```
.text:1000139B loc_1000139B:
.text:1000139B     call    generate32bitNumber
; for (i = 0; i < byteLength; i++)
;   bigNumber[i++] = generate32bitNumber();
.text:100013A0     mov     [esi], eax
.text:100013A2     add     esi, 4
.text:100013A5     dec     edi
.text:100013A6     jnz     short loc_1000139B
.text:100013A8     pop     edi
.text:100013A9
.text:100013A9 loc_100013A9:
.text:100013A9     test    bx, bx
.text:100013AC     jz      short loc_100013C1
.text:100013AE     call    generate32bitNumber
.text:100013B3     mov     edx, 1
.text:100013B8     mov     cl, bl
.text:100013BA     shl     edx, cl
.text:100013BC     dec     edx
.text:100013BD     and     eax, edx
; if (remainingBits != 0)
;   bigNumber[i] = generate32bitNumber() & ((1 << remainingBits) - 1);
.text:100013BF     mov     [esi], eax
```

Génération d'un nombre aléatoire de 32 bits :

```
.text:100013D2     xor     esi, esi
.text:100013D4     mov     edi, 4
.text:100013D9
.text:100013D9 loc_100013D9:
; for (result = 0, i = 4; i > 0; i--)
;   result = (result << 8) + rand();
.text:100013D9     call    j__rand
.text:100013DE     shl     esi, 8
.text:100013E1     add     esi, eax
.text:100013E3     dec     edi
.text:100013E4     jnz     short loc_100013D9
.text:100013E6     mov     eax, esi
```

Le constat est simple, l'ensemble des opérations de génération de nombres aléatoires s'effectue grâce aux fonctions standard du C srand et rand. srand se contente d'affecter la valeur passée en paramètre à l'aléa, variable globale.

Contenu de la fonction rand :

```
.text:10003916     mov     ecx, [eax+14h]
.text:10003919     imul    ecx, 343FDh
.text:1000391F     add     ecx, 269EC3h
; globalSeed = (globalSeed * 0x343fd) + 0x269ec3;
.text:10003925     mov     [eax+14h], ecx
.text:10003928     mov     eax, ecx
```

```
.text:1000392A     shr     eax, 10h
; result = (globalSeed >> 16) & 0x7fff;
.text:1000392D     and     eax, 7FFFh
```

Nous voici donc confronté à un cas pratique de génération de clé RSA basée sur l'utilisation d'un générateur d'aléa faible (environ 32 bits) [5].

3.4 Implémentation de l'attaque

Certes, les vulnérabilités théoriques d'implémentation de tels algorithmes existent, mais les possibilités qu'elles offrent à un particulier ne sont souvent pas claires, et les attaques à leur encontre restent généralement l'apanage d'organisations gouvernementales ou de gros centres de recherche ... qu'en est-il dans le présent cas ?

Le principe de l'attaque à mener est simple : un parcours exhaustif de l'espace des graines accompagné des calculs adéquats devrait nous permettre de retrouver les nombres P et Q utilisés pour calculer le paramètre N en notre possession. Cette solution n'est certainement pas unique, mais permettra d'aboutir à un résultat de façon efficace et peu demandant en puissance de calcul et en temps.

Plusieurs éléments de la procédure de génération de la clé RSA méritent d'être regardés en détails.

Initialisation du générateur de nombres pseudo aléatoires La graine de 32 bits initialisant le générateur de nombres pseudo aléatoires est issue d'une opération binaire simple sur les résultats de trois appels à des fonctions standard en environnement Win32 :

- *time(NULL)*, renvoie le nombre de secondes écoulées depuis le 1er janvier 1970 à `00:00:00 UTC`. Aspect particulièrement intéressant pour nous, sa valeur se situe entre *0*x386d3570 pour le 1er janvier 2000 et *0*x3a4fba6f pour le 31 décembre 2000 ;
- *GetCurrentThreadId()*, retourne l'identifiant de la "thread" (processus léger) ayant appelé la fonction, valeur entière guère supérieure à 5000 dans les environnements impliqués ;
- *GetTickCount()*, dont le résultat est le nombre de millisecondes écoulées depuis le démarrage du système, qui, sans jugement de valeur, dépassait rarement *0*x5265C00 (24 heures) sur un système Windows personnel en 2000.

De ces observations, on peut déduire un intervalle approximatif de l'expression

```
(time(NULL) + GetCurrentThreadId()) ^ GetTickCount()
```

dont le résultat devrait être compris entre *0*x38000000 et *0*x3fffffff, soit un espace de 2^27 valeurs, une réduction significative (bien que pas forcément exacte) des graînes possibles.

Travail sur les nombres premiers Bien qu'ayant réduit, de façon plausible, l'espace des valeurs à parcourir, la quantité d'opérations à effectuer reste conséquente, notamment à cause des opérations sur les grands nombres et les tests de primalités.

Dans un premier temps, il est nécessaire de s'affranchir de ces fameux tests. L'écart moyen entre le nombre généré P et ce fameux nombre premier immédiatement supérieur (appelons le P') est grossièrement de l'ordre de log(P) [6], qui vaut environ 355 dans le cas qui nous intéresse ($2^5 12 < P < 2^5 13$). Il est donc raisonnable de penser que la majeure partie des bits de poids fort de P et P' est identique. Il en va bien évidemment de même pour Q.

En conséquence, une très grande partie des bits de poids fort du produit $P * Q$ sera similaire à $P' * Q'$ (largement supérieure à 512 bits).

Le nombre de couples possibles (P, Q) étant de l'ordre de $2^3 2$ et normalement assez bien répartis, il est raisonnable de penser qu'un comparaison du nombre N de ASProtect avec le nombre N que nous allons être amenés à calculer sur leurs 64 bits de poids fort, ou plus, donnera des résultats satisfaisants.

Les tests de primalités en moins, le facteur limitant des itérations devient la multiplication des nombres non premiers P et Q de 512 bits. Un détail va nous permettre d'y remédier. Ces fameux nombres se fait par blocs de 32 bits. En multipliant les 32 bits de poids fort de P à ceux de Q, on obtiendra un nombre de 64 bits dont les 32 bits de poids fort correspondront à ceux du produit complet $P * Q$ aux retenues près.

Cette comparaison sur 32 bits permettra de déterminer si de plus amples calculs sont nécessaires, ou si l'on peut passer directement à la graine suivante.

On en déduit le contenu des itérations à effectuer lors du parcours de l'espace des graines :

1. initialisation de la graine ;
2. génération du nombre P de 512 bits avec 2 bits de poids fort à 1, par 16 appels successifs à la fonction de génération d'un nombre de 32 bits issue du code de RSAVR.dll ;
3. génération du nombre Q de 512 bits avec 2 bits de poids fort à 1, de façon similaire ;
4. si P et Q sont trop proches, retour à l'étape 3 ;
5. calcul du produit des entiers de poids fort de P et Q ;
6. si les 32 bits de poids fort du produit ne correspondent pas aux 32 bits de poids fort de N (aux retenues près), on incrémente la graine et on continue en 2. ;
7. s'il y a correspondance, on calcul le produit complet de P et Q, et on compare davantage de bits ;
8. si il y a de nouveau correspondance, on calcul P' et Q', sinon on passe à la graine suivante.

3.5 Code source

Fichier aspr.c :

```
/*
 * FreeLIP v1.1 is needed
 * gcc -Wall -O3 -lm -o aspr aspr.c lip.c
 */

#include <stdio.h>
#include <stdlib.h>
#include "lip.h"

unsigned long int asprSeed = 0;

/* Modulus de ASProtect */
unsigned long int asprN[32] =
{
  0x25702199, 0x3dba7047, 0x72ff2a2d,
  0x16420227, 0xc12f7ab4, 0x82a86c3e,
  0xf51e0583, 0x7b2e0c5f, 0x49b1d277,
  0x2ae60ca0, 0xaa237f2f, 0xa4dd85e9,
  0x800b0330, 0xccac5a58, 0xec61243b,
  0xdd096853, 0xd7034a3e, 0x3ed2d37c,
  0x9b9ce7f1, 0x30569052, 0x7a12106c,
  0xc3134461, 0x9af1ee35, 0xcb4e5cc4,
  0xbf6a03fe, 0xd9572b22, 0x436515ab,
  0x0b872d1c, 0xbffa8b4c, 0x77519791,
  0xa4815f62, 0xeb1d4ead
};

unsigned long int asprRandom(void)
{
  register unsigned long int r, s;

  /* Réimplémentation des appels successifs à rand(),
     retournant directement l'entier de 32 bits */
  r = asprSeed;
  r = (r * 0x343fd) + 0x269ec3;
  s = (r >> 16) & 0x7fff;
  r = (r * 0x343fd) + 0x269ec3;
  s = (s << 8) + ((r >> 16) & 0x7fff);
  r = (r * 0x343fd) + 0x269ec3;
  s = (s << 8) + ((r >> 16) & 0x7fff);
  r = (r * 0x343fd) + 0x269ec3;
  s = (s << 8) + ((r >> 16) & 0x7fff);
  asprSeed = r;
```

```
  return s;
}

int main(int argc, char *argv[])
{
  unsigned long int i, j;
  unsigned long int p[16], q[16];
  unsigned long int x, y, z;
  verylong vlP = 0;
  verylong vlQ = 0;
  verylong vlN = 0;
  verylong vlAsprN = 0;

  zultoz(asprN, 32, &vlAsprN);
  for (i = 0x3fffffff; i >= 0x30000000; i--)
  {
    /* Notre équivalent de srand(i) */
    asprSeed = i;
    /* Génération de P de 512 bits */
    for (j = 0; j < 16; j++)
      p[j] = asprRandom();
    /* 2 bits de poids fort à 1 */
    x = (p[15] |= 0xc0000000);
    do
    {
      /* Génération de Q de 512 bits */
      for (j = 0; j < 16; j++)
        q[j] = asprRandom();
      /* 2 bits de poids fort à 1 */
      y = (q[15] |= 0xc0000000);
      z = (x > y) ? x - y : y - x;
      /* Tant que P et Q sont trop proches */
    } while (z < 0x01000000);
    /* Multiplication des deux entiers de poids fort */
    z = asprN[31]
      - ((((unsigned long long)x) * y) >> 32);
    if (z <= 2)
    {
      fprintf(stdout, "[!] Probable good seed found : 0x%lx
                       (%lu)\n", i, z);
      fflush(stdout);
      /* Passage en verylong de P et Q */
      zultoz(p, 16, &vlP);
      zultoz(q, 16, &vlQ);
```

```
      /* Multiplication des deux grands nombres */
      zmul(vlP, vlQ, &vlN);
      /* Vérification de la similitude de 4 * 31 bits de
         poids fort du N calculé et du N de ASProtect */
      if (vlN[32] == vlAsprN[32]
          && vlN[31] == vlAsprN[31]
          && vlN[30] == vlAsprN[30]
          && vlN[29] == vlAsprN[29]) {
        fprintf(stdout, "[+] Good seed found : 0x%lx\n", i);
        /* Si concordance, calcul de P et Q */
        if (zodd(vlP) == 0)
          zsadd(vlP, 1, &vlP);
        zmod(vlAsprN, vlP, &vlN);
        while (ziszero(vlN) == 0)
        {
          zsadd(vlP, 2, &vlP);
          zmod(vlAsprN, vlP, &vlN);
        }
        zdiv(vlAsprN, vlP, &vlQ, &vlN);
        fprintf(stdout, "p = "); zwriteln(vlP);
        fprintf(stdout, "q = "); zwriteln(vlQ);

        return 0;
      }
    }
  }
  fprintf(stdout, "[-] Good seed not found\n");

  return 1;
}
```

3.6 Résultats

Exécution un Intel Pentium 4 à 2.26GHz :

```
kostya@icare:~/tools/aspr$ time ./aspr
[!] Probable good seed found : 0x399bacc4 (0)
[+] Good seed found : 0x399bacc4
p = 126288019099230838424474746472287405436197229176055513682050\
    384518153966116669245699179134519091648338851029999841022348\
    052599507362956069915523128660647

q = 130735286696710563710145518127522117222698023352396111601094\
    801959778351833870398894389269026767651645205879297083551553\
    672288089042904491091450944254399
```

```
real    1m46.711s
user    1m46.710s
sys     0m0.000s
```

Le principe est simple, les résultats obtenus tout à fait satisfaisants. Le temps moyen de cassage d'une clé RSA de 1024 bits générée par ASProtect est de l'ordre de 2 minutes.

4 Conclusion

Cas particulier ? Libre à chacun de le penser. Néanmoins ce n'est pas l'avis d'un certain nombre de 'crackers' qui a amplement parcouru le sujet avec un succès pour le moins surprenant : récupération de clés privées, falsification de signatures quel que soit l'algorithme, exploitation de débordement de buffer, et ce avec un unique point commun : une faiblesse d'implémentation trouvée et exploitée. Du petit éditeur de logiciels, au géant du système d'exploitation, personne n'est l'abri du reverse-engineering de ses binaires, dans des conditions plus ou moins légales. La sécurité de l'implémentation du système cryptographique par l'obscurité est alors une bien piètre protection ? La technique d'exploitation de la faiblesse du générateur de nombres aléatoires pour la génération des clés RSA de ASProtect présente l'avantage de s'adapter facilement à d'autres problèmes sous des environnements variés [7].

Références

1. Cryptos, `http://www.palmsoft.fr/`
2. OllyDbg, `http://home.t-online.fr/home/Ollydbg/`
3. Interactive Disassembler Pro, `http://www.datarescue.com/`
4. Miller-Rabin's Primality Test, `http://www.security-labs.org/index.php3?page=5`
5. Randomness Recommendations for Security, `http://www.ietf.org/rfc/rfc1750.txt`
6. How Many Primes Are There ? : `http://www.utm.edu/research/primes/howmany.shtml`
7. Enigme : problème d'aléa en JAVA MISC n°13.

Virus : Détections Heuristiques en environnement Win32

Nicolas Brulez

(SiliconRealms)

1 Introduction

Depuis la première apparition des virus Win32, de nombreuses techniques d'infections ont été inventées. Nombreuses de ses techniques restent utilisées à l'heure actuelle, il est donc important de bien comprendre leur principe de fonctionnement.

Tout comme il existe des detections heuristiques pour les virus DOS,de Macros, VBS etc, il existe aussi des detections heuristiques pour les virus Win32. Les méthodes les plus classiques d'infections sont présentées dans ce papier, ainsi que leurs détections heuristiques.

Les techniques anti heuristiques utilisées par les virus modernes seront aussi présentées.

Finalement, je présenterai mon propre moteur de detection Heuristique et les resultats obtenus par celui ci.

2 Les différents types de virus

Je présenterai ici seulement les virus qui infectent véritablement leur cible. Seront donc écartés, les virus compagnons, ainsi que les virus Prependers, qui écrasent le fichier hote. Il ne s'agit pas de vrais infections, et n'ont donc par conséquent, que très peu d'interet.

2.1 Les virus : Appenders

En général, ce type de virus ajoute son code à la fin du programme infecté. Lors de l'exécution d'un fichier infecté,le virus s'exécute en premier, pour infecter de nouveaux fichiers, puis rend la main au fichier hote qui s'exécute normalement. Ce type de virus est completement invisible à l'utilisateur dans la majorité des cas.

Virus Cryptés Comme leur nom l'indique, ces virus voient leur corps cryptés. Il existe deux types de virus cryptés. Le premier type de virus cryptés est composé d'un algorithme de chiffrement et d'une clé unique. Le virus sera toujours chiffré de la même manière. Le seul intêret de cette technique est d'empêcher l'analyse directe du virus.

L'autre type de virus cryptés utilise une clé différente pour chaque fichier infecté. Cette technique est apparue il y a maintenant de longues années, pour contourner les Anti Virus. Cependant, l'algorithme de dechiffrement restant toujours le même, il est très simple de détecter ces virus.

Oligomorphiques Contrairement aux virus cryptés, les virus Oligomorphiques contiennent plusieurs variations du décrypteur. Ces virus combinent chaque parties différement afin de créer un nouvel algorithme. En modifiant l'odre de ses blocks, il est possible d'obtenir une centaine de variations, ce qui implique une plus grande difficulté de détection de ces virus. Il faut autant de signatures qu'il y a de variations possibles pour détecter le virus à 100% (ou alors avoir recours à l'Emulation).

Polymorphiques Contrairement aux virus cryptés, les virus polymorphiques génèrent un code de decryptage différent, et utilisent des clés différentes. Cela leur permet d'echapper aux détections par signatures. Il existe plusieurs types de virus polymorphes, les polymorphes lents, et rapides.

Un virus polymorphique lent, ne changera de décrypteur qu'une fois par jour par exemple, ou à chaque redemarrage du pc. La récuperation de variantes différentes prends donc du temps. Le virus polymorphique rapide quant à lui, change de décrypteurs à chaque infection.

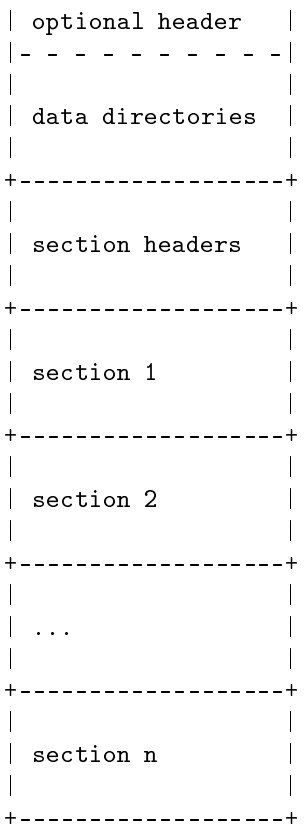
Le corps des virus reste inchangé, ce qui permet une detection du virus, une fois le virus décrypté. Les Anti virus utilisent en général leur émulateur pour décrypter le corps du virus, puis scan l'intérieur du virus decrypté, à la recherche d'une signature dans le corps en clair.

Métamorphiques Le métamorphisme peut être présenté comme un "polymorphisme du virus entier". Contrairement au polymorphisme classique, où seul le decrypteur est modifié, les virus metamorphiques ont une mutation de leur corps, ainsi que du decrypteur. Il est beaucoup plus compliqués de détecter toutes les variations d'un tel virus, car il n'a en théorie aucune partie constante. Les detections par statistiques peuvent être utilisées pour detecter ce genre de virus.

2.2 Présentation du format PE

Le format PE est le format des exécutables windows 32 bit. Je vais présenter rapidement son architecture, ses structures afin d'aider à la compréhension de cette présentation. Pour plus d'information, lire une documentation plus complète sur le format PE. Un fichier peut être représenté sous cette forme :

```
+-------------------+
| MZ Header         |
+-------------------+
| PE file-header    |
+-------------------+
```



Le PE Header Le PE Header débute en 0x00 par un MZ Header pour assurer une compatibilité MS-DOS. En effet, les exécutables win32 affichent un message d'erreur lorsqu'ils sont exécutés sous DOS : This programm cannot be run in DOS mode.

```
00000000 4D5A 9000 0300 0000 0400 0000 FFFF 0000 MZ..............
00000010 B800 0000 0000 0000 4000 0000 0000 0000 ........@.......
00000020 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030 0000 0000 0000 0000 0000 0000 8000 0000 ................
00000040 0E1F BA0E 00B4 09CD 21B8 014C CD21 5468 ........!..L.!Th
00000050 6973 2070 726F 6772 616D 2063 616E 6E6F is program canno
00000060 7420 6265 2072 756E 2069 6E20 444F 5320 t be run in DOS
00000070 6D6F 6465 2E0D 0D0A 2400 0000 0000 0000 mode....$.......
```

On trouve aussi l'adresse du PE file header dans le MZ Header. Celle-ci est placée dans le champ e_lfanew disponible en +3Ch.

Le PE file header

```
00000080 5045 0000 4C01 0600 8D54 F32F 0000 0000 PE..L....T./....
00000090 0000 0000 E000 0E01 ........
```

La structure du PE file header est :

```
PE File Header:
+0 DWORD PE\x00\x00
4 WORD Machine;
6 WORD NumberOfSections;
8 DWORD TimeDateStamp;
C DWORD PointerToSymbolTable;
10 DWORD NumberOfSymbols;
14 WORD SizeOfOptionalHeader;
16 WORD Characteristics;
```

Le PE Optional Header

```
18 WORD Magic;
1a UCHAR MajorLinkerVersion;
1b UCHAR MinorLinkerVersion;
1c DWORD SizeOfCode;
20 DWORD SizeOfInitializedData;
24 DWORD SizeOfUninitializedData;
28 DWORD AddressOfEntryPoint;
2c DWORD BaseOfCode;
30 DWORD BaseOfData;
34 DWORD ImageBase;
38 DWORD SectionAlignment;
3c DWORD FileAlignment;
40 WORD MajorOperatingSystemVersion;
42 WORD MinorOperatingSystemVersion;
44 WORD MajorImageVersion;
46 WORD MinorImageVersion;
48 WORD MajorSubsystemVersion;
4a WORD MinorSubsystemVersion;
4c DWORD Reserved1;
50 DWORD SizeOfImage;
54 DWORD SizeOfHeaders;
58 DWORD CheckSum;
5c WORD Subsystem;
5e WORD DllCharacteristics;
60 DWORD SizeOfStackReserve;
64 DWORD SizeOfStackCommit;
68 DWORD SizeOfHeapReserve;
6c DWORD SizeOfHeapCommit;
70 DWORD LoaderFlags;
74 DWORD NumberOfRvaAndSizes;
```

Cette structure contient des informations très importantes telles que la RVA2 du point d'entrée. Le point d'entrée est l'adresse de la première instruction à exécuter. Nous verrons plus tard qu'elle a une importance capitale dans l'étude des protections PE.

Toutes les adresses contenues dans le PE header sont données sous la forme d'adresses relatives à l'image base (champ ImageBase). Pour obtenir l'adresse en mémoire, il faut ajouter ces RVA à l'image base d'un programme. Pour obtenir l'image base d'un programme, il suffit de lire la valeur contenue dans la structure, en +34h. Il s'agit de l'adresse à laquelle sera chargé l'exécutable en mémoire. En général, celle-ci vaut 0x400000.

Le DataDirectory Il s'agit d'un tableau de structures contenant des informations diverses tel que l'import table, l'export table, etc.

Les Section Headers Ils sont accessibles à l'adresse PE optional header + F8h. Chaque section de l'exécutable est décrite par la structure suivante :

```
+0 8byte ANSI name (Nom de la section)
+8 dword misc (taille de la section en mémoire)
+C dword virtual address (RVA de la section)
10 dword sizeofrawdata (Taille de la section sur le disque)
14 dword pointerToRawData (Offset de la section)
18 dword pointerToRelocations
1C dword PointerToLinenumbers
20 word NumberOfRelocations
22 word NumberOfLineNumbers
24 dword Characteristics (Caractéristiques de la section)
```

2.3 Principes du code relogeable

Un virus doit, pour fonctionner et infecter des programmes hôtes différents, pouvoir être lancé à n'importe quelle adresse mémoire, et être toujours fonctionnel. Pour se faire, les virus utilisent du code relogeable. Ce principe de code relogeable existait déja sous DOS.

Le principe est très simple, et consiste à utiliser un offset de référence pour accéder aux variables du virus.

Les données du virus se trouvent toujours à la même distance de l'offset de référence quelque soit l'adresse ou est chargé le virus, ce qui permet de retrouver les infos nécéssaires au bon fonctionnement du parasite. L'offset de référence est en général appelé "delta offset".

```
Exemple:
--------

virus_start:
```

```
call get_delta

get_delta:

pop ebp
sub ebp,offset get_delta
```

A partir de la, EBP contient un offset permettant d'accéder aux données du virus.

Explications.- L'instruction call pousse sur la pile l'adresse de retour, c'est à dire l'adresse qui précède le call. Les virus se servent de cette caractéristique pour récupérer l'adresse à laquelle ils ont été chargé grace à l'utilisation de l'instruction pop, qui va lire l'adresse de retour sur la pile.

Le virus soustrait ensuite à l'adresse ou il est chargé, l'offset du delta de la première génération. On obtient ainsi un delta offset pour accéder à nos données.

Dans la seconde génération du virus, c'est à dire une fois qu'un fichier est infecté, le virus soustrait à l'adresse poussée sur la pile, l'anciene adresse du pop, ce qui a pour effet de nous donner le déplacement exacte pour accéder à nos données.

Il suffit alors d'utiliser le registre "ebp" pour référencer nos données :

```
mov eax, dword ptr [ebp+kernel]
```

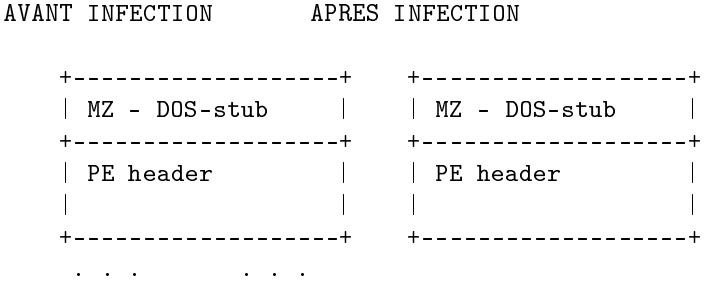
"kernel" est représenté par une adresse une fois compilé. La valeur dans le registre EBP (delta offset) permet d'ajuster l'adresse, et donc de retrouver les données du virus.

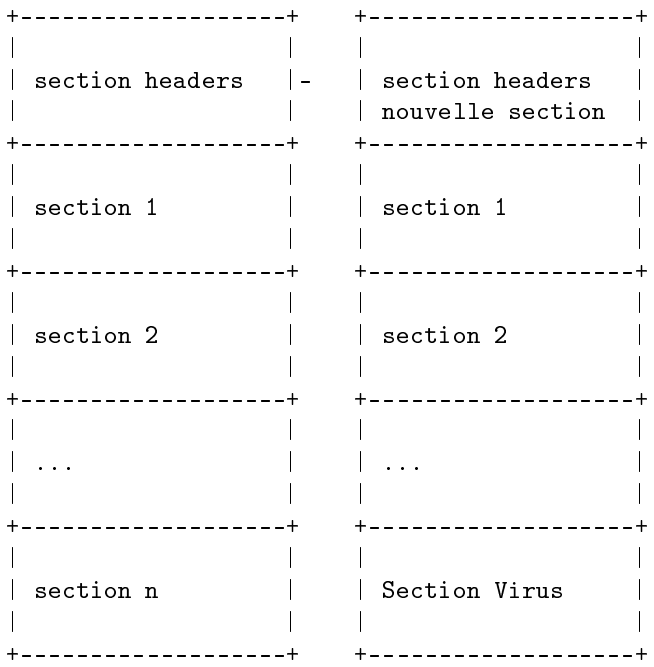
3 Présentation de quelques types d'infections Win32

3.1 Emplacement du virus

Dernière section La plus part des virus se copient à la fin de leur fichier hote, plus précisement dans la dernière section des exécutables PE. Il existe deux types d'infections de ce type.

- **Ajout de Section.**- Les premiers virus ajoutaient une nouvelle section aux fichiers infectés. Le premier virus pour Win95, Boza ajoutait une section du nom de ".vlad".





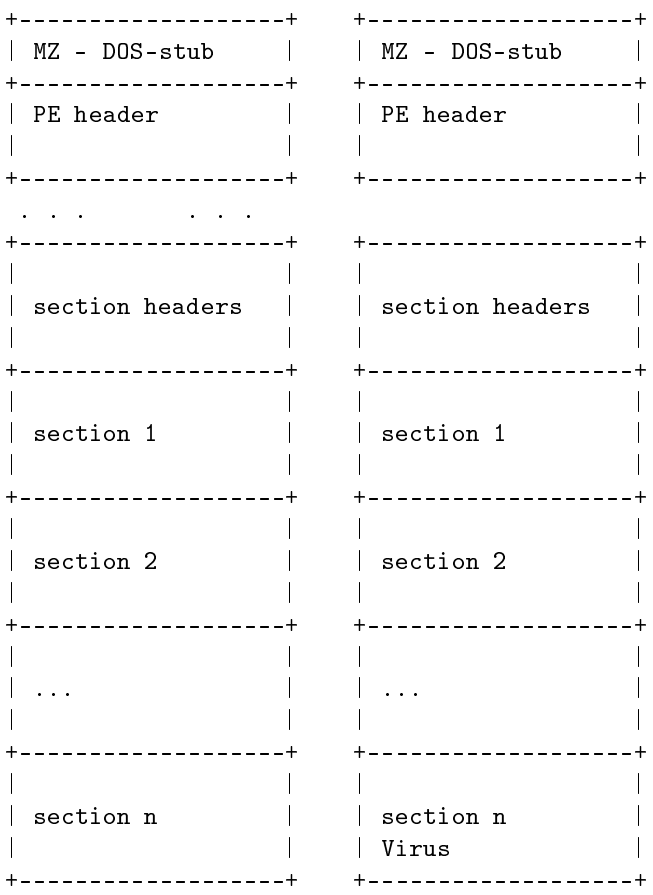
Ces virus modifient le PE Header afin d'ajouter une nouvelle section.

```
.qjxh:004106F6                public start
.qjxh:004106F6 start          proc near
.qjxh:004106F6
.qjxh:004106F6 arg_4          = dword ptr  8
.qjxh:004106F6 arg_8          = dword ptr  0Ch
.qjxh:004106F6
.qjxh:004106F6                jmp     $+5
.qjxh:004106FB
.qjxh:004106FB loc_4106FB:    ; CODE XREF: .qjxh:00407127j
.qjxh:004106FB                mov     ecx, 10000h
.qjxh:00410700                mov     ebp, offset unk_404200
```

etc

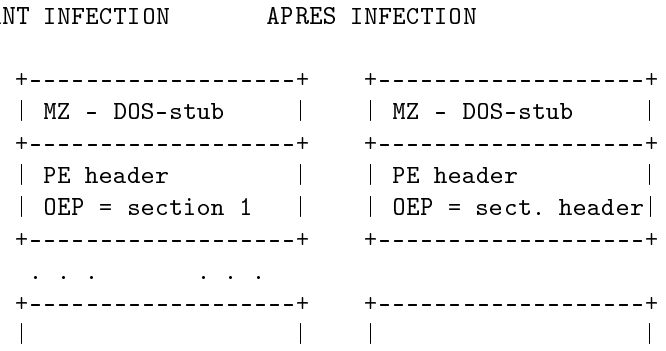
– **Agrandissement de la dernière section**.- Par la suite, de nouveaux virus sont apparus, ceux ci n'ajoutaient pas de nouvelle section, mais agrandissaient la dernière section pour s'y loger. Cette technique a plusieurs avantages.
Elle est plus facile à mettre en oeuvre : pas de grosse modification du PE header. Elle est aussi plus furtive. L'infection n'apparait pas d'un simple coup d'oeil a la vue du nom des sections.

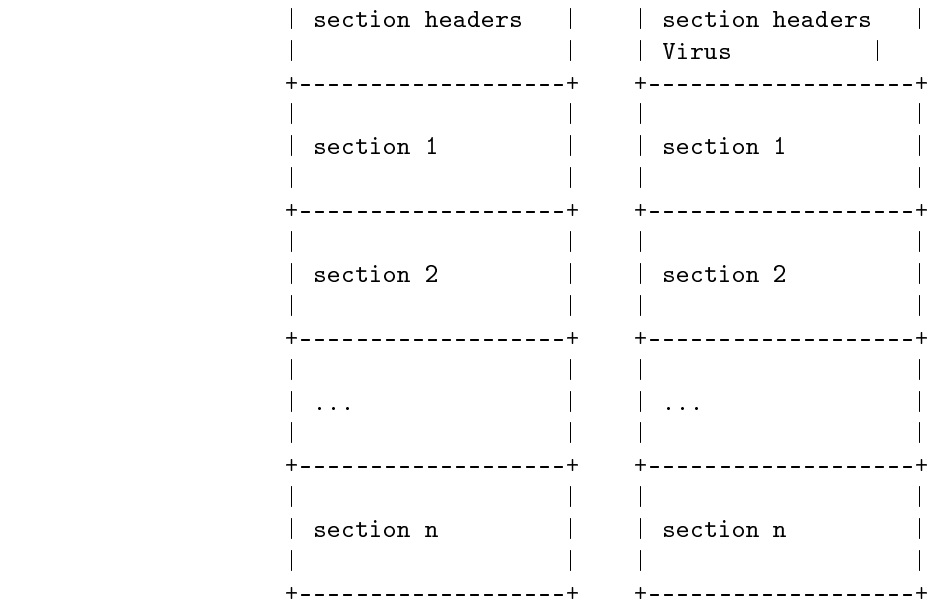
AVANT INFECTION APRES INFECTION



Exemple de virus : Anxiety.

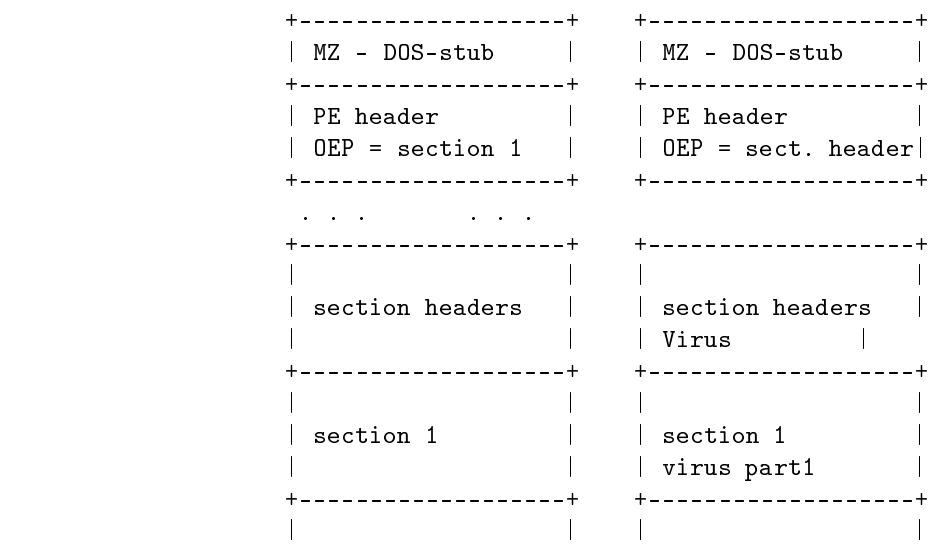
Infection du Header Sous windows 95, certains virus se copiaient juste après le section header, et juste avant la première section. Meme si le code du virus n'était dans aucune section, le loader de windows autorisait l'exécution du virus.

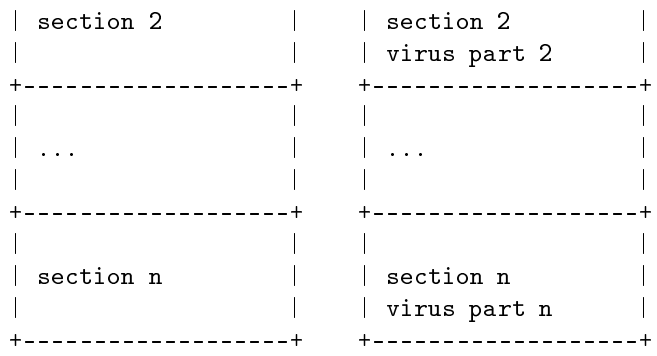




Exemple de virus : Murkry

Cavité Les infecteurs par cavités utilisent le padding des sections pour se loger. Le corps du virus est dissimulé en plusieurs parties dans le padding utilisé pour aligner les sections, et permet d'infecter un fichier sans modifier la taille de celui ci.



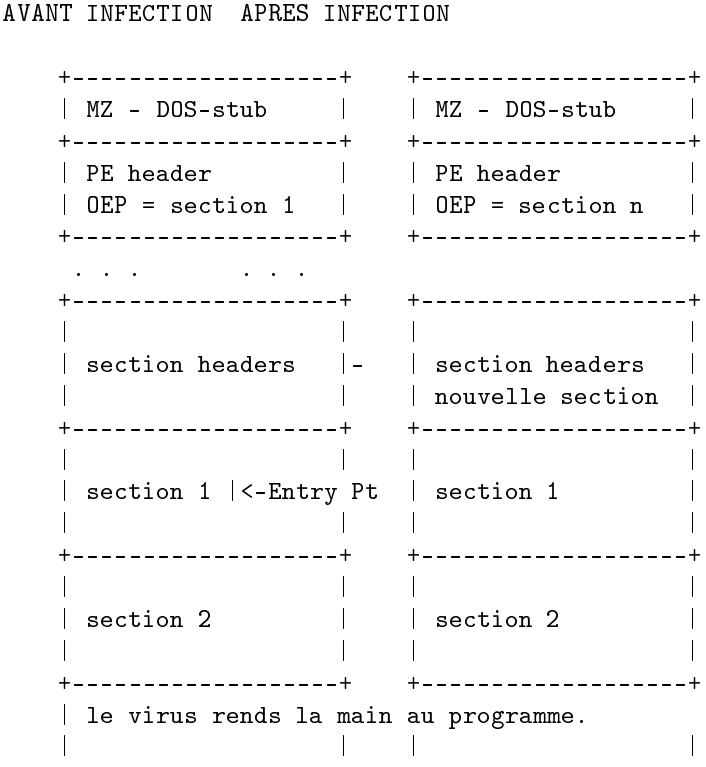


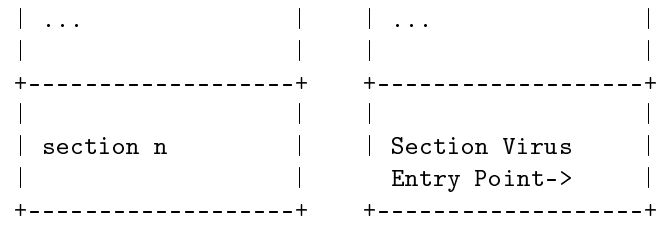
Exemple de virus : CIH

3.2 Point d'entrée

Les virus pour s'activer, doivent modifier le programme afin de prendre la main avant le programme hote. Différentes techniques existent, voici les principales :

Dans la dernière section



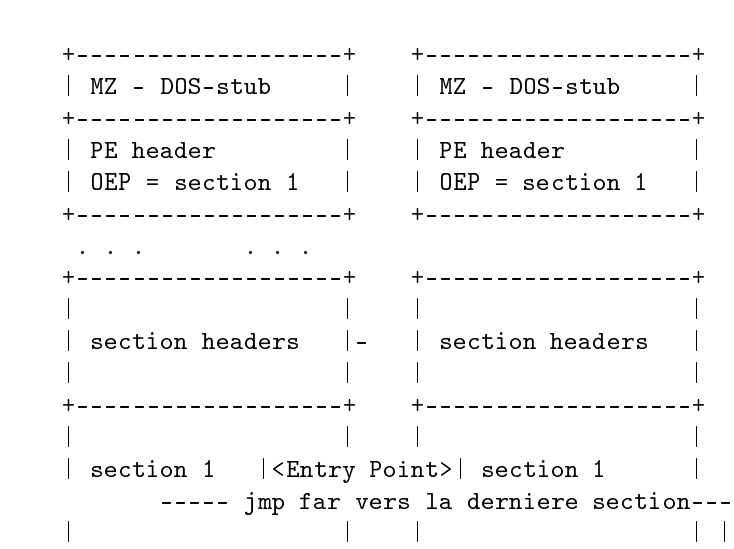


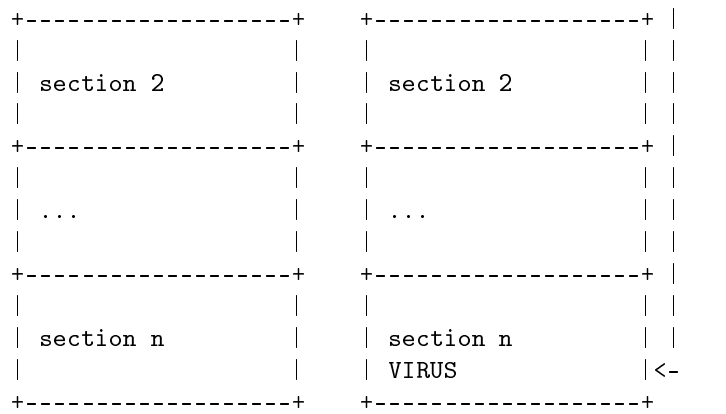
Le point d'entrée d'un fichier infecté pointe dans la derniere section. Le programme commence directement par le code du virus, qui après s'être reproduit, rends la main au fichier hote.

```
.reloc:00409FF4         public start
.reloc:00409FF4 start   proc near
.reloc:00409FF4         and     bl, 99h
.reloc:00409FF7         mov     edx, 1BB51A23h
.reloc:00409FFC         ror     edx, 0BCh
.reloc:00409FFF         movsx   ebp, ah
.reloc:0040A002         mov     ebp, 0EC58F5B4h
.reloc:0040A007         rol     ebp, 2Fh
.reloc:0040A00A         mov     ecx, ebp
etc
```

(Extrait du virus Crypto qui commence dans la dernière section. Ici section .reloc.)

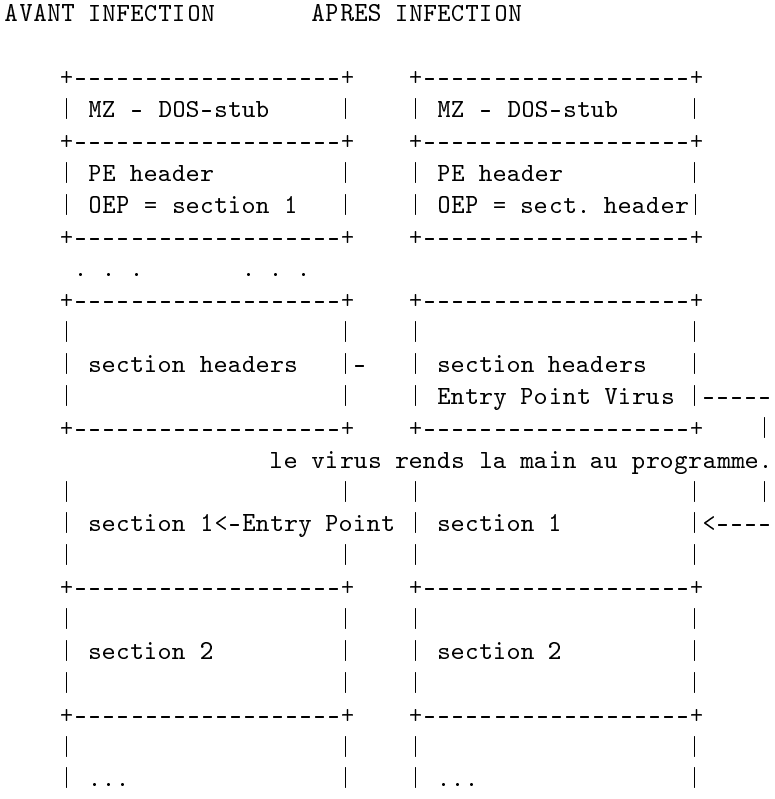
Dans la première section

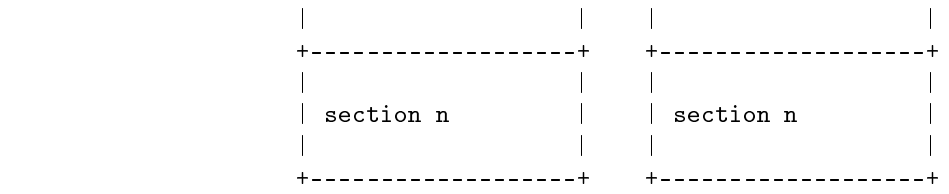




Une technique utilisée certaine fois, est l'insertion d'un saut vers le code du virus à l'entry point du programme hote afin de rediriger le fichier infecté sur le code du virus, d'une magnière plus furtive. Exemple de virus : Marburg.

Avant la première section L'entry point commence avant la premiere section :





Le virus au moment de l'infection modifie le PE header pour qu'il pointe non pas vers le debut du programme original, mais apres la section header. Cette section n'a pas de caracteristiques, mais microsoft a cru bon de la laisser exécutable, donc exploitable par un virus. Le virus se copie donc apres la table des sections et avant la premiere section.

IDA nous avertit d'ailleurs lorsqu'un fichier commence avant la premiere section.

```
HEADER:00400400 ;
HEADER:00400400 ; The code at 400000..401000
            is hidden from normal disassembly
HEADER:00400400 ; and was loaded because the
            user ordered to load it explicitly
HEADER:00400400 ;
HEADER:00400400 ;  <<<< IT MAY CONTAIN TROJAN
    HORSES, VIRUSES, AND DO HARMFUL THINGS >>>
HEADER:00400400 ;
HEADER:00400400 ;
HEADER:00400400
HEADER:00400400 ;
------- S U B R O U T I N E -----------------
HEADER:00400400
HEADER:00400400
HEADER:00400400             public start
HEADER:00400400 start       proc near
HEADER:00400400             xchg    eax, esi
HEADER:00400401             push    edi
HEADER:00400402             sidt    qword ptr [esp-2]
HEADER:00400407             pop     edi
HEADER:00400408             fild    qword ptr [edi]
....
```

3.3 Infection par modification de e_lfanew

L'infection par modification d'elfanew est la méthode d'infection la plus simple à mettre en oeuvre. En effet, ce type d'infection ne nécessite pas de code relogeable. Le virus se copie entiérement à la fin du fichier. Il s'agit d'un exécutable standard, avec ses propres imports. Pour infecter un fichier, le virus modifie le pointeur vers le PE header (elfanew), et le fait pointer sur le PE

Header du virus. Le loader de Windows ne vas donc voir que le code du virus et l'exécuter. Celui ci après avoir infecté d'autres fichiers, effectue une copie du fichier infecté et re-modifie elfanew pour pointer ver le programme hote. Le virus exécute ensuite le fichier "temporairement" désinfecté et attends que celui lui rende la main, puis efface le fichier temporaire, et le virus se ferme.

4 Les detections heuristiques Win32

4.1 Analyse de la structure PE

Les techniques de detections heuristiques en environnement win32 sont basées en grande partie sur l'analyse de la structure des exécutables PE. D'une manière générale, les fichiers sont compilés selon un même modèle.

Point d'entrée dans la dernière section Une fois compilés, les programmes ont pour point d'entrée la première section, soit la section code. Les premiers virus, ou les virus simples, modifient le point d'entrée, et le font pointer dans la dernière section, la ou le virus s'est copié. Un programme par defaut, n'aura jamais un point d'entrée pointant ailleurs que dans la première section.

Point d'entrée avant la première section Comme vu précédement, Le point d'entrée d'une application standard pointe dans la première section, ou du moins en aucun cas avant la première section. Il s'agit d'une caractèristique très suspecte et seulement utilisée par les virus.

Caractéristiques des sections Chaque section a des caractéristiques données, et en régle général, toutes caractéristiques différentes pour un type de section donnée doit être considéré comme suspect.

- **Dernière section exécutable**.- En régle général, seul la première section est exécutable, la dernière section est généralement utilisée pour les relocations, le debug, ou les ressources. Celles ci ne sont pas exécutables. Ce type de caractéristique est donc suspect. Cependant, les packers d'exécutables, utilisent eux aussi régulierement, la dernière section, pour contenir le code de décompression des fichiers packés. Il y a une risque de faux positifs.
- **Première section "writeable"**.- La première section a généralement, les attributs d'exécution et de lecture, mais pas d'écriture. Les virus modifient les caractéristiques de la première section en "writeable", lorsque ceux ci ont cryptés le code du fichier hote, ou alors quand le virus se copie dans la premiere section et qu'il doit mettre à jours ses propres variables, il a besoin des droits en écriture.

Nom des sections et leurs caractéristiques Certains virus renoment la derniere section avant de changer ses caractéristiques pour être moins visibles. Cependant certains ne renoment pas les sections, et se contentent de changer

leurs attributs. Une section ".reloc" ou ".rsrc" par exemple n'aura jamaisle flag "executable", et il s'agit très probablement d'un signe d'infection. Cette caractéristique est donc TRES suspecte.

La "Virtual Size" est incorrect dans le PE header La plus part des virus pour windows 9x n'alignent pas "SizeOfImage" avec l'alignement de section le plus proche. Sous les anciens windows, le loader n'y prete pas attention, et les fichiers s'exécutent, contrairement à Windows NT qui refuse d'exécuter l'application si l'alignement est incorrecte. Cela permet de detecter les anciens virus.

PE header en fin de fichier Dans le cas des virus qui infecent par modification de elfanew, le PE header se retrouve en fin de fichier. Le PE header n'est jamais au dela d'une centaine d'octets dans un fichier normal, il est donc possible de detecter ce type d'infection et de noter l'adresse très suspecte du PE header.

Size Of Code incorrect Beaucoup de virus ne mettent pas à jours le champ "SizeOfCode" lorsqu'ils ajoutent une section avec l'attribut exécutable activée. Si la somme des tailles des sections code est supérieure a la taille définie dans le champ "SizeOfCode", il est fort probable qu'une section ait été ajoutée, probablement par un virus. Suspect.

4.2 Analyse du code

Après l'analyse structurale d'un fichier PE, il est possible d'effectuer une analyse heuristique à partir du code de l'application.

Instruction non standard au point d'entrée. La majorité des applications compilés commencent par un stack frame, ou certaines instructions spécifiques, utilisées pour chaque fichier compilé par un compilateur donné. Si un fichier ne commence pas par ses instructions standards, le fichier peut avoir été infecté par un virus, peut etre packé, ou programmé en assembleur. Il n'est pas inutile de prendre en compte les instructions au point d'entrée du fichier, cela peut apporter une information supplémentaire si d'autres caractèristiques suspectes ont été identifiées dans un fichier. Moyenement suspect.

Calcul du delta offset Comme vu en introduction, les virus ont besoin d'avoir un code relogeable pour pouvoir fonctionner. Le code utilisé est généralement un "call instruction suivante", et sa représentation hexadécimale est la suivante : 0xE800000000. Certains virus utilisent des variations, il est donc possible d'utiliser des régles un peu moins restrictives pour eviter d'être trompé par du code comme celui ci :

```
call get_delta
db 0EBh ; octet inutilisé mais permettant
                ; de tromper une detection trop
                ; simpliste d'un delta offset.
get_delta:
pop ebp
```

Redirection de code Suspect Les virus ne modifiant pas le point d'entrée modifient le code au point d'entrée, pour donner la main au virus d'une manière plus discrete.

- **JMP Far**.- Soit en ajoutant un saut vers une autre section, c'est à dire vers le code du virus, détectable en scannant le point d'entrée à la recherche d'un "jmp far", il est possible d'utiliser un désassembleur pour obtenir un resultat plus précis.
- **PUSH / RET**.- Soit en ajoutant une combinaison d'instructions, tel qu'un "push adresse virus" suivis d'un RET. Ce type d'appel n'est jamais utilisé dans un programme normal, et surtout pas pour appeler le code d'une section différente de la section à laquelle se trouve ce code.
  Ex:

  ```
  406070 Push 414345
  406075 ret
  ```

Un virus peut modifier un programme d'une multitude de manières pour récuperer la main lors de l'exécution du fichier infecté, il peut donc être interressant d'ajouter d'autres type de code assez simple pour appeler le code d'une autre section.

Recherche de fichiers PE Un autre type de detection heuristique par analyse de code, est la detection de la recherche de fichiers PE. Un virus, pour pouvoir infecter un fichier, va d'abord vérifier s'il a bien affaire à un fichier PE, et donc un fichier infectable. On retrouve souvant dans les virus, le code suivant :

```
cmp word ptr [registre],'EP'
cmp word ptr [registre],'ZM'

Exemple dans un virus:

reloc:00404260                  cmp     word ptr [esi], 'ZM'
```

Ce type de code est facilement détectable, surtout à l'aide d'un désassembleur intégré au moteur heuristique.

Chaines de caractères particulieres Certains virus contiennent des chaines de caractères en clair, surtout lors de la première génération, ou en cas de virus simples.

- "***.exe**".- Il n'est pas rare de trouver en clair, la chaine "*.exe" dans la section code d'un virus. Il s'agit du mask de recherche des fichiers qu'infecte le virus. Cette chaine permet de trouver de nouveaux fichiers hotes. Attention aux faux positifs, bien définir les regles de recherche : *.exe dans la derniere section par exemple si celle ci est différente de .data.
- **API et dll windows**.- Les premiers virus avaient dans leur section, le nom des fonctions d'API windows à charger en clair, ainsi que le nom des dlls à laquelle elles appartiennent. Certaines fonctions sont caractèristiques des virus Win32 (*FindFirstFileA, FindNextFileA, CreateFileA, CreateFileMapping, MapViewOfFile...* par exemple). Si ces chaines se retrouvent toutes en même temps dans la section supposée du virus, et que d'autres éléments suspects ont été detectés, il est fort probable qu'il s'agisse d'un virus simple, ou d'une première génération d'un virus qui n'a pas encore chiffré ses chaines par exemple.

Utilisation du PEB pour récupérer des adresses systemes Les virus fonctionnant seulement sous le systeme d'exploitation Windows NT peuvent récuperer l'adresse des fonctions de l'API Windows en utilisant le PEB (Process Environement Block). Il est possible de detecter ce code en désassemblant le point d'entrée d'un fichier.

Exemple concret d'un virus :

```
.rsrc:0040461B     mov     edx, fs:30h
.rsrc:00404621     mov     eax, [edx+0Ch]
.rsrc:00404624     mov     esi, [eax+1Ch]
.rsrc:00404627     mov     eax, [esi]
.rsrc:00404629     mov     esi, [eax+8]
.rsrc:0040462C     xchg    eax, esi
.rsrc:0040462D     mov     [ebp+401900h], eax
```

Détection de code utilisant des adresses systemes "hardcodées" Certains virus contiennent les adresses de certaines dll en dur dans leur code. En cas d'echec de la récupération de l'imagebase d'une dll par exemple, ou par exemple pour accéder au PEB (voir section précédente), en utilisant l'adresse du PEB directement. Celle ci est statique.

Exemple d'adresse hardcodée :

```
.reloc:00404630   mov     ebx, 0BFF70000h
```

L'adresse "BFF70000h" correspond à l'image base de la dll "kernel32.dll" dans les systemes d'exploitation de type Windows 98.

4.3 Emulation simple

Il est possible de coupler l'émulation à l'heuristique pour obtenir de meilleurs resultats.

JMP Far Comme nous l'avons vu plus haut, certains virus ne modifient pas le point d'entrée, et inserent un saut vers la section du virus. Il est possible d'émuler ce saut, et de continuer l'analyse du code dans la section du virus. Une fois au véritable point d'entrée du virus, la recherche d'indices qui nous permettent d'indiquer la présence eventuelle d'un virus peut continuer.

PUSH / RET Certains virus utilisent d'autres instructions pour se rendre au code du virus. Certains utilisent la combinaison PUSH / RET.

L'émulation de cette combinaison permet de trouver le véritable point d'entrée du virus, et de continuer l'analyse du code.

Il existe une multitude de possibilités, c'est pourquoi il est necéssaire d'émuler au minimum les instructions les plus classiques : PUSH, POP, RET, MOV, JMP, CALL, XOR, ADD, SUB, SHL, SHR...

Emulation des décrypteurs Les virus cryptés ou polymorphes utilisent des routines de décryptage, qui empechent l'analyse du code du virus, même si celui ci a pu être trouvé. Un émulateur de bonne qualité permet d'emuler les boucles de décryptages pour ensuite scanner l'interieur du virus, et donc de détecter les caractèristiques virales.

5 Techniques Anti Heuristiques

Les techniques de détections évoluants, les virus ont eux aussi évolués pour continuer les méthodes de détections heuristiques. Je vais vous présenter maintenant, différentes méthodes anti heuristiques.

5.1 Structure PE

Non Modification des caractéristiques des sections Certains virus ne modifient pas les sections de la dernière section. Ils utilisent la fonction de l'API Windows : VirtualProtect. Le virus change les caractéristiques de la "section" en mémoire, et peut donc exécuter du code, meme si sur le disque la section n'a pas l'attribut exécutable. Les moteurs heuristiques ne peuvent donc pas detecter l'exécution dans la derniere section.

Ajout de plusieurs sections En régle général, le virus se copie dans la dernière section. Cependant, certain virus ajoutent deux sections. La première des sections est exécutable et contient le code du virus, alors que la seconde n'est pas utilisée, et ne contient aucun attribut suspect tel que l'exécution ou l'écriture. Les anti virus heuristiques scannant la dernière section ne detectent rien d'anormal.

Ajout d'un bout de code du virus dans la première section (point d'entrée tjs dans la section code) Certains virus copient le code du programme original qu'ils écrasent, dans la derniere section, sans toucher les attributs de celle ci. Ils écrasent ensuite le code sauvegardé, et ajoutent un tout petit bout de code, permettant d'allouer de la mémoire pour copier ensuite le véritable code du virus en mémoire alloué, et ensuite d'exécuter celui ci. Le virus prends donc la main, et replace le code précédement écrasé et infecte d'autres fichiers, avant de rentre la main à l'application qui s'exécutera normalement.

Le principe de cette technique permet d'exécuter le code du virus, sans avoir à changer les attributs des sections, et donc de rester discret et de ne pas se faire detecter par les moteurs heuristiques.

Packing de la section code et ajout du virus dans l'espace non utilisé Cette technique est très similaire à celle précédement présentée, excepté que la section code est compressée pour prendre moins de place. Le virus se copie ensuite à l'endroit précédement utilisé par le code non compressé. Le comportement du virus est ensuite très similaire à l'explication précédente.

Point d'Entrée Obscure La technique du point d'entrée obscure est une évolution du simple saut inséré à l'entry point du fichier hote. De nombreuses techniques ont été inventées pour rendre la main au virus, sans modifier le point d'entrée du programme infecté.

- **Patch des appels aux API windows pour appeler le virus**.- Une des méthodes les plus utilisée est la modification d'un appel vers une fonction de l'API Windows. Les fonctions dans un programme sont appelées de la manière suivante :

```
.text:004012A4
; void __stdcall ExitProcess(UINT uExitCode)
.text:004012A4   ExitProcess     proc near
; CODE XREF: start+Fp
.text:004012A4 FF 25 04 20 40 00
              jmp     ds:__imp_ExitProcess
.text:004012A4
                 ExitProcess     endp
```

  Certains compilateurs utilisent un FF 15 (call dword ptr) pour appeler les fonctions.
  Voici le même appel patché par un virus :

```
.text:004012A4 sub_4012A4 proc near
 ; CODE XREF: start+Fp
.text:004012A4   call    loc_404600
.text:004012A9            nop
.text:004012A9 sub_4012A4 endp
```

  Concraitement, le virus a modifié l'appel à la fonction ExitProcess. Au lieu d'appeler cette fonction, le programme appelera le code du virus. Dans le cas de la fonction ExitProcess, le virus se déclanchera seulement quand on

ferme le programme. Les sandbox n'émulant que le début du programme ne détecteront pas ce type de virus par exemple.
Les moteurs heuristiques doivent scanner les appels aux fonctions d'API pour detecter ce type de détournement, et cela implique un scan plus long car tout le fichier doit être parcouru, surtout si la fonction patchée est choisie aléatoirement.

- **Patch du Stack Frame.-** Certains virus modifient l'initialisation d'une fonction pour y insérer un saut vers le code du virus. Ce type de point d'entrée obscure est plus difficile à detecter que le premier. Le premier peut être détecté en scannant l'espace du fichier contenant tout les appels aux fonctions d'API, il est très simple de trouver un détournement. Dans le cas du patch du stack frame, il n'y a aucun espace contenant tout les stack frames des fonctions.
  L'anti virus doit scanner tout le fichier de façons intelligentes pour detecter le code du virus. Si le virus utilise un code différent pour chaque patch du Stack Frame, il peut être très difficile de détecter ce genre de virus, surtout s'ils sont en plus polymorphiques.

Calcul du Checksum du fichier PE Les virus anti heuristiques mettent à jour le checksum des fichiers PE qu'ils infectent, afin d'éviter le drapeau "checksum du fichier incorrect" dans les moteurs heuristiques.

Renomage des sections existantes Les virus élargissants la dernière section, modifient souvent les caractèristiques de la derniere section pour leur ajouter l'attribut exécutable. Les anti virus detectent ce genre de modification, et la suspicion est encore plus grande si le nom de la section est connue pour ne jamais avoir les attributs exécutables : .reloc par exemple. Certains virus modifient tout simplement le nom de la section après l'avoir modifiée.

A noté qu'il est possible de détecter ce genre de modification si le pointeur vers les relocations dans le PE header n'a pas été modifié. Une section de relocation n'ayant pas pour nom ".reloc" est très suspect.

"Size of Code" est corrigé Les virus anti heuristiques mettent à jour la size of Code pour éviter d'obtenir le drapeau lorsqu'ils sont scannés par les moteurs heuristiques.

5.2 Anti Emulation

Les moteurs utilisant l'émulation étant de plus en plus utilisés, des techniques anti émulations sont apparues.

SEH - Structured Exception Handling. Pour contourner les premiers Emulateurs, les virus utilisaient des Exceptions Handlers (Gestion des erreurs par le programme) afin de déstabiliser les émulateurs. Les virus créaient des erreurs

dans leur code, et s'occupaient eux même de gérer l'erreur pour continuer le code juste après. Certains virus utilisaient ce principe pour effectuer le saut final vers le fichier hote. De nos jours, tout les Emulateurs dignes de ce nom, supportent les Exceptions.

Instructions du Co-Processeur Les premiers Emulateurs ne pouvaient emuler les instructions du co-processeur tel que le FPU. Les auteurs de virus utilisaient donc ces instructions dans le code de leur virus pour déstabiliser les Emulateurs et donc empecher la detection de leur virus.

MMX / SSE Comme pour les instructions FPU, certains virus utilisent les instructions MMX et SSE pour decrypter leur "corps". Les emulateurs ne supportant pas ses instructions ne pouvaient donc pas decrypter le corps du virus, et donc continuer l'analyse.

Tout les Emulateurs actuels supportent ce type d'instructions de nos jours.

Instructions non documentées Les processeurs intel contiennent quelques instructions non documentées et les emulateurs ne supportant pas ses instructions ne pouvaient pas correctement emuler les virus les utilisant intelligement. De nos jours, ses instructions sont correctement émulées.

Code Anti Machine Virtuelle Certains virus utilisent du code pour empecher leur exécution dans une machine Virtuelle telle que Vmware ou Virtual PC. Il est généralement assez simple de detecter ses machines virtuelles, certains virus peuvent ensuite agir différement en fonction du type de machine détectée, ou alors cesser de fonctionner tout simplement dans une machine virtuelle.

Couches de cryptage avec brute forcing Des virus comme Crypto utilisent des couches de cryptage qui se brutent forcent jusqu'a trouver la clé permettant le décryptage de la suite du virus. L'interet de ce type de cryptage/decryptage et qu'il ralentie considérablement le scanning des fichiers, les Emulateurs étant plus lents qu'une exécution réelle d'un programme, l'émulation complete des layers de cryptage d'un virus utilisant ce type d'algorithme peut prendre plusieurs minutes par fichiers.

Threads Certains virus utilisent des threads pour exécuter certaines opérations, comme le decryptage du code du virus. Les Emulateurs ne supportant pas le multi threading ne peuvent pas emuler correctement ce type de virus.

5.3 Code anti heuristique

Le delta offset est obtenu différement Certains virus calculent le delta offset différement afin de ne pas se faire detecter par les anti virus heuristiques

qui effectuent de l'analyse de code. En général le code du delta offset est de la forme : E800000000 ou très similaire. Il est aussi suivit d'un pop registre.

Il est possible de programmer le calcul du delta offset de cette manière :

```
startvirus:

call calculatedelta
delta:
sub ebp, (offset delta1-start)

<code du virus ici>

calculatedelta:
pop ebp
jmp delta
```

De cette manière le code généré est différent des calculs de delta offsets standard, et permet donc d'echapper aux detections simples de calculs de delta offset. Il est possible de modifier ce code en ajoutant du code qui n'a aucune utilité entre chaque instructions, pour rendre les detections encore plus complexes.

Le code pour la recherche de fichiers PE est obscurci Certains moteurs detectent la recherche de fichiers PE. Pour contre carrer ca, les auteurs de virus peuvent employer des méthodes différentes :

```
mov eax, 'EP' xor 46h
; le resultat de "EP" xor 46h est stockée
; lors de la compilation.
xor eax,46h
; au lancement la valeur est mise dans EAX,
; puis ensuite xorée.
cmp byte ptr [edi],al
; on compare le premier octet pointé par EDI
; avec al qui contient "P"
jnz bad_pe_file
; si c'est pas bon, on sort.
cmp byte ptr [edi+1],ah
; sinon si le deuxième octet est égal à "ah",
; c'est à dire "E", alors c'est un fichier PE.
jnz bad_pe_file
; sinon on sort.
```

Une autre variation :

```
mov  eax,dword ptr [edi]
; EDI pointe sur PE\00
shl eax,3
```

```
; 00004550 << 3
cmp ax,2a80h
; si ax = 2a80 alors fichier PE
jnz bad_pe_file
; sinon mauvais fichier.
```

Il existe des centaines de possibilités pour programmer les detections différements, et ce genre de code contourne les analyses de code simples sans difficultés.

Les API ne sont plus référencées directement (checksum) Certains moteurs heuristiques recherchent la présence des chaines de caractères correspondantes au nom de fonctions de l'API Windows souvant utilisées par les infecteurs de fichiers PE, dans la section ou le virus est sensé se trouver. Pour contourner ce problème, la majorité des virus contiennent le CRC de la chaine de caractère pour identifier l'API à charger, et ils utilisent souvent un CRC maison, pour eviter la recherche des valeurs de CRC, si ceux si utilisent des algorithmes standards.

6 Présentation d'un moteur Heuristique Perso

J'ai programmé un moteur heuristique permettant la detection de virus Win32 connus et inconnus en se basant sur l'analyse de la structure PE, ainsi que du code de l'application.

6.1 Analyse de binaires standards : notepad, regedit, calc, Ms Pain, WordPad...

Les applications standards ne déclanchent aucun drapeau.

Notepad

```
Heuristic Engine v0.06
Filename: C:\WINNT\NOTEPAD.EXE

Nb Sections: 7
Size of Code: 38000
Entry Point: 2F9A9
Image Base: 1000000

.text Vsize:65CA RVA:1000 Psize:0
      offset:0 Flags:60000020
.data Vsize:1944 RVA:8000 Psize:0
      offset:0 Flags:C0000040
.text1 Vsize:30000 RVA:A000 Psize:2B000
      offset:1000 Flags:E0000020
```

```
.adata Vsize:10000 RVA:3A000 Psize:D000
      offset:2C000 Flags:E0000020
.data1 Vsize:20000 RVA:4A000 Psize:9000
      offset:39000 Flags:C0000040
.pdata Vsize:30000 RVA:6A000 Psize:27000
      offset:42000 Flags:C0000040
.rsrc Vsize:6000 RVA:9A000 Psize:4000
      offset:69000 Flags:40000040

This file looks normal :-)
```

Regedit

```
Heuristic Engine v0.06
Filename: C:\WINNT\regedit.exe

Nb Sections: 3
Size of Code: AE00
Entry Point: 72E6
Image Base: 1000000

.text Vsize:ADBC RVA:1000 Psize:AE00
      offset:600 Flags:60000020
.data Vsize:48AF0 RVA:C000 Psize:200
      offset:B400 Flags:C0000040
.rsrc Vsize:8000 RVA:55000 Psize:7200
      offset:B600 Flags:40000040

This file looks normal :-)
```

Taskman

```
Heuristic Engine v0.06
Filename: C:\WINNT\TASKMAN.EXE

Nb Sections: 3
Size of Code: 4C00
Entry Point: 28A0
Image Base: 1000000

.text Vsize:4A84 RVA:1000 Psize:4C00
      offset:600 Flags:60000020
.data Vsize:3938 RVA:6000 Psize:2400
      offset:5200 Flags:C0000040
.rsrc Vsize:2000 RVA:A000 Psize:1600
      offset:7600 Flags:40000040
```

```
This file looks normal :-)
```

6.2 Analyse de binaires Infectées : Virus polymorphes, Cryptés, Standard, EPO...

Virus XTC

```
Heuristic Engine v0.06
Filename: C:\reverse engineering\VIRUS\NOUVEAUX\XTC.VIR

Nb Sections: 5
Size of Code: 0
Entry Point: 9000
Image Base: 400000

.AVXencr Vsize:2000 RVA:1000 Psize:E00
         offset:400 Flags:C0000040
.AVXencr Vsize:2000 RVA:3000 Psize:400
         offset:1200 Flags:C0000040
.AVXencr Vsize:2000 RVA:5000 Psize:400
         offset:1600 Flags:C0000040
.rsrc Vsize:2000 RVA:7000 Psize:200
         offset:1A00 Flags:C0000040
.AVXencr Vsize:4000 RVA:9000 Psize:3400
         offset:1C00 Flags:C0000040

Instruction at entry point is not usually used
by a compiler. ASM ? ;-)
Execution starts in last section.
Last section is executable. (Its used by virus
and PE protector)
Suspicious Delta trick (used by injected code)
found at EP+14
Code section is writeable.

This file is probably infected :-/
```

Virus Cocaine

```
Heuristic Engine v0.06
Filename:
C:\reverse engineering\VIRUS\NOUVEAUX\COCAINE.VIR

Nb Sections: 4
Size of Code: 2C00
```

```
Entry Point: 3750
Image Base: 1020000

.text Vsize:2BFE RVA:1000 Psize:2C00
      offset:400 Flags:60000020
.data Vsize:9CC RVA:4000 Psize:A00
      offset:3000 Flags:C0000040
.rsrc Vsize:BAC RVA:5000 Psize:C00
      offset:3A00 Flags:40000040
.reloc Vsize:7000 RVA:6000 Psize:6200
      offset:4600 Flags:C0000040

Instruction at entry point is not usually used
by a compiler. ASM ? ;-)
Suspicious JMP FAR found at Entry point+9.Might
be some OEP obfuscation.
Suspicious difference between OEP RVA and
Section RVA.OEP obfuscation?

This file is probably infected :-/
```

Virus Energy

```
Heuristic Engine v0.06
Filename:
C:\reverse engineering\VIRUS\NOUVEAUX\Energy.vir

Nb Sections: 5
Size of Code: 0
Entry Point: 6000
Image Base: 400000

9286742 Vsize:1000 RVA:1000 Psize:600
        offset:400 Flags:C0000040
5808079 Vsize:1000 RVA:2000 Psize:0
        offset:A00 Flags:C0000040
9079359 Vsize:2000 RVA:3000 Psize:200
        offset:A00 Flags:C0000040
.rsrc Vsize:1000 RVA:5000 Psize:200
        offset:C00 Flags:C0000040
.data Vsize:2000 RVA:6000 Psize:1C00
        offset:E00 Flags:C0000040

Instruction at entry point is not usually used
by a compiler. ASM ? ;-)
Execution starts in last section.
```

```
Last section is executable. (Its used by virus
and PE protector)
Suspicious Delta trick (used by injected code)
found at EP+14
Code section is writeable.

This file is probably infected :-/
```

Virus Crypto

```
Heuristic Engine v0.07
Filename:
H:\reverse engineering\VIRUS\NOUVEAUX\CRYPTO.VIR

Nb Sections: 4
Size of Code: 200
Entry Point: 9FF4
Image Base: 400000

.text Vsize:DD5 RVA:1000 Psize:1000
      offset:1000 Flags:60000060
.idata Vsize:E4 RVA:2000 Psize:1000
      offset:2000 Flags:40000040
.rsrc Vsize:1000 RVA:3000 Psize:1000
      offset:3000 Flags:40000040
.reloc Vsize:8000 RVA:4000 Psize:8000
      offset:4000 Flags:E2000060

Instruction at entry point is not usually used
by a compiler. ASM ? ;-)
Execution starts in last section.
Last section is Writeable and Executable.
A section known not to have any code is executable!
Very suspicious.

This file is probably infected! :-/
```

PEjacky

```
Heuristic Engine v0.06
Filename:
C:\reverse engineering\VIRUS\Souches InfectÚs
   (via vmware)\PEjacky\NO
EPAD.VIR

Nb Sections: 3
```

```
Size of Code: 6600
Entry Point: F800
Image Base: 1000000

.text Vsize:65CA RVA:1000 Psize:6600
      offset:600 Flags:60000020
.data Vsize:1944 RVA:8000 Psize:600
      offset:6C00 Flags:C0000040
.rsrc Vsize:6000 RVA:A000 Psize:5E00
      offset:7200 Flags:E0000040

Execution starts in last section.
Last section is executable. (Its used by virus
and PE protector)
Suspicious code looking for PE files. (could
be a virus or a PE tool
Suspicious difference between OEP RVA and
Section RVA.OEP obfuscation?

This file is probably infected :-/
```

6.3 Analyse de fichiers PE packés : UPX, Aspack, FSG...

PeLocknt

```
Heuristic Engine v0.06
Filename:
C:\reverse engineering\TASM5Plus\EXAMPLE\contemplate/
  \packers\pelocked.EXE

Nb Sections: 5
Size of Code: 600
Entry Point: 5000
Image Base: 400000

PELOCKnt Vsize:1000 RVA:1000 Psize:600
         offset:600 Flags:E0000020
PELOCKnt Vsize:1000 RVA:2000 Psize:600
         offset:C00 Flags:C0000040
PELOCKnt Vsize:1000 RVA:3000 Psize:200
         offset:1200 Flags:C0000040
PELOCKnt Vsize:1000 RVA:4000 Psize:200
         offset:1400 Flags:C0000040
PELOCKnt Vsize:2A00 RVA:5000 Psize:2A00
         offset:1600 Flags:C0000060
```

```
This file is protected by PE-Locknt 2.4
```

PE Protect

```
Heuristic Engine v0.06
Filename:
C:\reverse engineering\TASM5Plus\EXAMPLE\contemplate/
  \packers\pe-prot.EXE

Nb Sections: 5
Size of Code: 600
Entry Point: 5000
Image Base: 400000

CODE Vsize:1000 RVA:1000 Psize:600
     offset:600 Flags:E0000020
DATA Vsize:1000 RVA:2000 Psize:600
     offset:C00 Flags:C0000040
.idata Vsize:1000 RVA:3000 Psize:200
     offset:1200 Flags:C0000040
.reloc Vsize:1000 RVA:4000 Psize:200
     offset:1400 Flags:50000040
.PE-PROT Vsize:1000 RVA:5000 Psize:455
     offset:1800 Flags:C0000040

This file is protected by PE-Protect
```

7 Conclusion

Les techniques heuristiques Win32 permettent de détecter beaucoup de virus Win32 sans avoir recours à des signatures spécifiques. Cependant, des problèmes de faux positifs existent, notament avec les packers d'exécutables PE, qui fonctionnent grossièrement, de la même manière que les virus. (Code Relogeable, Delta Offset, etc) Il est important d'utiliser des réglès minutieusement définie, ainsi qu'un émulateur puissant, pour pouvoir détecter la plus grand majorité des infections d'exécutables Win32 PE.

Les moteurs heuristiques permettent toutes fois de detecter un exécutable "suspect", et ensuite de permettre l'analyse de ce fichier, par une personne compétente.

Quel avenir pour la sécurité Windows ?

Nicolas Ruff

EdelWeb
nicolas.ruff@edelweb.fr

1 Introduction

Les logiciels Microsoft (tels que MS-DOS et Windows 3.1) ont longtemps traîné une réputation d'instabilité chronique plutôt pittoresque (qui ne connaît pas le fameux "écran bleu" ?). Aujourd'hui les dernières générations logicielles (Windows 2000 et XP) doivent faire face à une menace beaucoup plus grande : celle de l'ouverture aux réseaux, Internet en particulier. Les défaillances logicielles ne sont plus alors de simples nuisances pour l'utilisateur, mais des failles majeures dans lesquelles s'engouffrent virus, spammeurs et pirates informatiques.

Après avoir longtemps ignoré ce problème, Microsoft se devait de réagir face à plusieurs attaques de grande envergure (historiquement Melissa et Code Red). Plusieurs annonces fortement médiatisées autour de la sécurité ont alors été faites en plus haut lieu (formation de tous les développeurs, audit de code), sans empêcher pour autant de nouvelles attaques à large diffusion (Slammer, Blaster, Klez, MyDoom, la liste est longue).

Qu'en est-il exactement et qui faut-il croire ?

Cette présentation a pour vocation de retracer un historique de la sécurité chez Microsoft, les grandes campagnes, ce qu'elles ont apportées à la sécurité Windows et les failles qu'elles ont laissées. Cet historique abordera également le présent de Windows, avec les innovations techniques majeures (pourtant mal documentées) de Windows 2003 et de Windows XP SP2.

Ayant passé en revue les imperfections de tous les mécanismes susmentionnés, cette présentation se conclura par quelques éléments de prospective sur les solutions de protection matérielles déjà annoncées et leur efficacité attendue, ainsi que les nouveaux risques auxquels ces solutions devront faire face dans un avenir très proche.

2 Un peu d'histoire

2.1 Lorsque la sécurité n'existait pas...

Le 29 juillet 1996 sort Windows NT4.

Corrigeant la plupart des défauts de jeunesse identifiés dans Windows NT 3.51, Windows NT4 allait marquer le début de la conquête par Microsoft du segment des réseaux locaux d'entreprise, jusque là partagé entre plusieurs systèmes dont principalement des systèmes Unix.

La famille Windows NT se caractérise par une architecture complètement différente des logiciels qui ont fait jusque là le succès de Microsoft (MS-DOS, Windows 3.1 et Windows 95). En effet Windows NT possède un vrai noyau 32 bits, exploitant pleinement les capacités de protection matérielle des processeurs Intel, et ne reposant plus sur une antique couche MS-DOS pour son amorçage.

A l'époque le principal avantage perçu de cette architecture est avant tout la stabilité : le dysfonctionnement d'une application utilisateur n'affecte plus le fonctionnement global du système. Bien que cette assertion soit vraie en théorie, en pratique il s'avère que les nombreux bogues de composants critiques, et surtout des pilotes s'exécutant en mode noyau, font de NT4 un système qui ne tient pas toutes ses promesses. D'ailleurs d'après Microsoft, 90% des "écrans bleus" sont provoqués par des pilotes d'affichage.

Cette nouvelle architecture présente également un intérêt non négligeable du point de vue de la sécurité. La principale préoccupation de l'époque étant les virus, il est vrai que le système NT4 leur porte un coup dur : absence de système DOS pour lequel sont conçus la majorité des virus (avec les techniques bien connues "Interrupt Hijacking" et "Terminate and Stay Resident"), secteur d'amorçage exotique, impossibilité d'accéder directement au disque dur, complexité du format PE, etc. Les concepteurs de virus se tournent alors vers les virus "macro", qui présentent l'avantage d'être multi plateformes.

Toutefois les fonctions réseau avancées et les mécanismes de sécurité robustes[1] de Windows NT4 intéressent des groupes de hackers, tels le célèbre L0pht. Dès lors les premières attaques commencent à apparaître sur Internet : attaque en rejeu sur l'aléa du protocole LM, outil de cassage de mots de passe L0phtCrack, attaque Red Button permettant l'énumération des comptes de manière anonyme....

Face à la grogne des clients et à la multiplication des alertes, Microsoft se voit contraint de réagir et inaugure le 1er bulletin de sécurité le 1er juin 1998 (MS98-001 : "*Disabling Creation of Local Groups on a Domain by Non-Administrative Users*").

Dès lors la question de la sécurité des logiciels va aller croissant, face à des incidents de plus en plus nombreux et étendus (qui vont de pair avec la croissance du réseau mondial).

- 26 mars 1999
  - Propagation mondiale de Melissa (macro-virus Word).
- 19 juillet 2001
  - Propagation du ver Code Red sur les serveurs Web IIS 5.
- 25 janvier 2003
  - Propagation du ver Slammer sur SQL Server et MSDE. De nombreuses infrastructures sensibles sont perturbées.
- 12 août 2003
  - Propagation du ver Blaster, qui exploite une faille RPC présente dans toutes les versions de Windows. Un grand nombre de "home users" sont

[1] Par opposition à Windows 9x.

affectés. Chacun des vers ci-dessus exploite une faille pour laquelle un correctif est disponible depuis plusieurs mois ?

Le nombre de bulletins de sécurité émis annuellement par Microsoft depuis 1999 se situe entre 60 et 100, et ce malgré des efforts notables pour diminuer cet indicateur devenu une valeur de référence dans la compétition entre éditeurs.

2.2 Round 1 : "Microsoft Security Initiative"

Code Red et Slammer sont les plus illustres représentant de la classe de vulnérabilités la plus répandue : le débordement de tampon alias "buffer overflow". Ce type de vulnérabilité est rendu possible par la conjonction de deux facteurs : un langage de programmation bas niveau (ici le C) laissant au développeur le soin de dimensionner ses variables, et un manque de sensibilisation des développeurs à une pratique de programmation "sécurisée". A la décharge de ces pauvres développeurs, il faut bien admettre qu'une grande partie du code des systèmes actuels a été écrit à une époque où le concept même de "buffer overflow" n'avait pas été découvert.

Face à ce constat, Bill Gates annonce personnellement les moyens de réponse que met en oeuvre Microsoft :

- Un arrêt temporaire (2 mois) des nouveaux développements, au profit d'un audit global du code existant. Des outils automatisés, tels que PREFix et PREFast, sont développés en interne par Microsoft à cette fin.
- Une formation des développeurs aux problèmes de sécurité, reprenant l'excellent ouvrage "Writing Secure Code" de Michael Howard, aux éditions Microsoft Press.

Le coût estimé par Microsoft de cette opération est de 200 millions de dollars (une broutille comparativement au chiffre d'affaire de la société).

Au final, il est indéniable que cette opération a profité à la qualité du code source de Windows, compte tenu du nombre de bogues documentés a posteriori et corrigés silencieusement par Microsoft. Le nombre de bogues corrigés lors de cette opération n'a pas été annoncé officiellement par Microsoft, quant au nombre de bogues qui ont échappé à la correction ? il est par définition impossible à connaître !

Ce qui est sûr, c'est qu'au mois de juillet 2003 le groupe polonais LSD annonçait la découverte d'un "buffer overflow" trivial dans le service RPC, avec pour conséquence la propagation du ver Blaster. Ce bogue a été identifié par rétro-ingénierie sans accès au code source.

Cette démarche montre donc ses limites sur un code complexe (plus de 30 millions de lignes de code), dont la conception initiale remonte à plus de 10 ans.

2.3 Round 2 : "Get Secure, Stay Secure"

Malgré l'amélioration de la qualité du code source et la correction (souvent silencieuse) de nombreux bogues, le parc Windows déployé reste attaqué avec succès partout de par le monde. Microsoft identifie deux causes à ce problème rédhibitoire :

"*La configuration par défaut du système n'est pas sécurisée.*"

Cette configuration a été conçue pour fonctionner dans tous les environnements (professionnel et domestique) et maximiser la fonctionnalité ; or le besoin de rétro-compatibilité en entreprise impose l'activation d'options archaïques et dangereuses (tels que les sessions nulles sur SMB). On se souvient que Code Red s'est propagé de manière importante car le serveur Web IIS était installé par défaut avec Windows 2000 Serveur, et que le gestionnaire d'extensions ".IDA" était actif par défaut.

Au final, et d'après le "top 10" du SANS, l'attaque la plus souvent utilisée contre un Windows 2000 consiste à se connecter à distance via un compte administrateur local sans mot de passe ?

Windows XP innove donc en la matière, avec une surface d'attaque réduite par rapport à Windows 2000, et une configuration adaptée à chaque utilisation. Ainsi sur un poste en "Workgroup", il n'est pas possible de se connecter via le réseau avec un mot de passe vide, et les sessions nulles sont interdites.

Le concept n'a toutefois pas été poussé jusqu'au bout, puisque le nouveau service UPnP était démarré par défaut ? et rapidement identifié comme vulnérable à un "buffer overflow" !

Windows 2003 Server diminue encore la surface d'attaque par rapport à ses prédécesseurs : chaque applicatif système doit être installé individuellement, en fonction des besoins.

Philosophiquement il est intéressant de constater que les applicatifs système de Windows 2003 ne sont pas forcément moins bogués, mais font l'objet de moins de tentatives d'attaque à partir du moment où la base installée est plus faible. Après tout, quitte à investir du temps dans la recherche de nouvelles attaques, autant maximiser sa portée potentielle ?

Pour les entreprises, Microsoft publie également à titre gracieux des guides de sécurisation, destinés à remplacer les guides "non officiels" fleurissant un peu partout sur Internet (NSA, SANS...).

On peut reprocher à ces guides d'être volumineux (plus de 400 pages pour le guide Windows 2000), et de nécessiter un gros effort d'interprétation et d'adaptation au contexte de l'entreprise (pas toujours évident avec des guides rédigés majoritairement en anglais). Toutefois ces guides constituent indéniablement une mine d'informations pour les personnes désireuses de s'impliquer dans la sécurité de leur parc Microsoft.

Parallèlement des outils gratuits tels que MBSA assistent l'utilisateur (nécessairement éclairé malgré l'effort de pédagogie du produit) dans la configuration sécurisée de son poste de travail.

Malgré tout, des vulnérabilités restent présentes dans le coeur historique (et archaïque) du système, celui qui échappe à toute démarche de sécurisation proactive. Le meilleur exemple est le ver Blaster, utilisant une faille dans les services RPC. Désactiver un tel service dans Windows est suicidaire, et les options de configuration peu nombreuses ?

"*Les utilisateurs n'appliquent pas les correctifs de sécurité.*"

Effectivement, les efforts de correction de code mettent du temps à se répercuter sur la base installée, y compris en entreprise où les modifications du système d'information sont rarement préventives.

Or la principale cause de propagation des vers est bien la non application des correctifs de sécurité, on citera en particulier Slammer qui est apparu un an après le correctif adéquat ?

On notera au passage que Microsoft a nié publiquement[2] l'utilisation de "0-day"[3] pour Windows, alors que plusieurs exemples patents ont été documenté tels que la compromission d'un site militaire américain via la faille WebDAV, ou la propagation quotidienne de virus via des failles IE.

Toujours est-il que Microsoft s'est concentré sur la mise à jour de ses produits : activation par défaut de Windows Update dans Windows X, ; outils gratuits de gestion des correctifs (MSUS, HFNetChk), envoi de CDs gratuits sur simple demande, changement de la politique de publication (deuxième mardi de chaque mois) afin de permettre aux entreprises de s'organiser.

Le bilan que l'on peut tirer de cette initiative est mitigé : les entreprises qui se sont dotées de moyens de gestion des correctifs arrivent à obtenir des résultats, avec néanmoins les limites suivantes :

- Gestion du nomadisme.
- Gestion des masters (intégration difficile des correctifs dans les images d'installation, suivi de version et compatibilité applicative),
- Gestion de parc (identification des postes concernés et déploiement massif, problème du retour arrière et des postes non gérés).
- Continuité de service (qualification des correctifs, compatibilité applicative et problème du redémarrage).

En ce qui concerne les PME et les utilisateurs domestiques, voire certains grands groupes, tout reste à faire. La meilleure preuve en est la propagation fulgurante du ver Blaster en août 2003. De plus les outils gratuits disponibles chez Microsoft (Windows Update, MSUS) ne gèrent que la plateforme Windows / IE et non les autres applicatifs Microsoft, d'où la propagation du ver Slammer sur SQL Server et MSDE. A la décharge des utilisateurs, il faut également admettre que les virus exploitent aujourd'hui des failles IE plus rapidement qu'elles ne sont corrigées. Il semble toutefois que 2004 s'annonce comme l'année du "patch management", avec du côté de Microsoft la sortie du nouveau "Microsoft Update".

2.4 Round 3 : "La défense périmétrique"

Puisque la sécurisation du coeur historique de Windows semble vouée à l'échec, et qu'il existera toujours une majorité d'irréductibles utilisateurs qui n'appliqueront pas les correctifs de sécurité et les guides de configuration, la nouvelle approche de Microsoft dans le SP2 de Windows XP consiste à isoler

[2] We have never had vulnerabilities exploited before the patch was known `http://news.co.uk/1/hi/technology/3485972.stm`

[3] Vulnérabilité non documentée et *a fortiori* non patchée.

complètement l'utilisateur du réseau, en plus des corrections sensibles apportées aux principaux vecteurs d'attaque du système (Internet Explorer, Outlook Express, RPC, COM/DCOM...).

Cette approche radicale a des conséquences non négligeables sur la compatibilité applicative, comme nous le verrons plus loin dans la présentation du SP2.

Il ne faut pas confondre défense périmétrique et défense en profondeur, un autre concept hérité du domaine militaire, applicable aux systèmes d'information.

La défense en profondeur consiste à opposer aux attaquants plusieurs lignes de défense successives. On peut citer par exemple la séparation des privilèges dans le démon SSH et la notion de "prison" (jail) obtenue grâce à la commande chroot. Lorsque ces mécanismes sont en place, la compromission d'un service réseau ne permet pas d'obtenir immédiatement les droits les plus élevés sur le système. Or actuellement avec Windows XP, même SP2, l'utilisateur reste administrateur local du poste et de nombreux services réseau sensibles (tels que RPC) tournent sous le compte SYSTEM ?

Avec une défense périmétrique, la moindre brèche permet de compromettre complètement le système.

3 Les nouveautés majeures

3.1 Windows XP/2003 : les révolutions silencieuses

Protection de pile Lors de la publication en juillet 2003 de la faille DCOM/RPC ayant donné lieu au ver Blaster, aucun code d'exploitation pour Windows 2003 n'a été rendu public alors que ce système était annoncé comme vulnérable par le groupe LSD et Microsoft.

A contrario le bogue MS04-006 dans le service WINS provoque un déni de service sur Windows 2003 uniquement, alors que toutes les versions de Windows sont annoncées comme vulnérables.

La raison technique profonde de ces comportements est une nouveauté introduite dans le compilateur Visual Studio.NET via l'option de compilation /GS. Il s'agit d'un mécanisme de protection de pile à l'exécution (runtime), basé sur l'insertion d'une valeur aléatoire en pile (appelée canary ou cookie) avant tout appel de fonction manipulant des buffers.

Ce mécanisme n'est pas nouveau, il existe sous Unix dans le produit StackGuard par exemple. Visual Studio 6 intègre également une protection de ce type, via les options /GZ et /RTC, mais celles-ci impactent trop les performances pour être utilisées en mode "Release" (elles avaient été pensées dans un but de déboguage uniquement).

La nouveauté est que l'ensemble de Windows 2003 Server, y compris le code "hérité" (de type WINS, RPC) et IIS 6.0, a été compilé avec cette option, ce qui le rend beaucoup plus robuste vis-à-vis des possibilités d'exploitation malveillante en cas de débordement de buffer. (Cette technologie n'élimine pas les

débordements de buffer, mais bien la possibilité de les exploiter dans la plupart des cas).

D'autres produits majeurs, tels que le ".NET Framework" ou Office 2003, ont également été compilés avec cette option. Microsoft est tellement satisfait des premiers retours d'expérience que l'ensemble du SP2 pour Windows XP sera également recompilé avec /GS.

On notera que le projet Fedora (version communautaire de Red Hat Linux) a également été compilé en grande partie avec la technologie "exec-shield".

Gestionnaire d'exceptions Sans rentrer dans les détails, une technique d'exploitation des débordements de buffer bien connue et mise en ?uvre par Code Red en 2001 consiste à modifier l'adresse du gestionnaire d'exception (située en pile). Cette technique permet de s'affranchir de la protection /GS vue précédemment.

Face à cette menace, plusieurs nouveautés concernant le traitement des exceptions ont été introduites dans Windows XP :

- Le *dispatcher* d'exceptions refuse de transférer le contrôle à un gestionnaire, si celui-ci réside dans la pile.
- Les registres EAX, EBX, ESI et EDI sont effacés avant le traitement de l'exception, afin d'éviter la fuite d'adresses mémoire.
- Les gestionnaires d'exception peuvent être déclarés dans une section spécifique du format de fichier exécutable PE. Aucun autre gestionnaire ne peut alors être appelé par le programme à l'exécution. Cette technologie n'est quasiment jamais utilisée en pratique, car la plupart des gestionnaires d'exception sont dynamiques.

.NET Framework A partir de Windows 2003, le ".NET Framework" est installé en standard avec le système. Certains composants de Windows 2003 sont des assemblies ".NET".

Par conception, et sauf bogue majeur dans le Framework, les applications ".NET" sont immunisées aux problèmes de débordement de buffer pour deux raisons :

- Il s'agit d'un langage interprété, les assemblies contenant en fait du "pseudo-code". Le Framework effectue des contrôles forts à l'exécution.
- Les types tableau ne sont pas gérés par le développeur mais par le langage (la notion de pointeur n'existe pas).

Options de sécurité De nombreuses options de sécurité ont été ajoutées dans Windows XP, et ces options sont activées par défaut. Les mêmes options ont été reprises dans Windows 2003. Parmi les plus significatives on notera :

- Les comptes avec mot de passe vide ne peuvent pas être utilisés pour des opérations réseau (mode workgroup uniquement).
  - L'accès au partage administratif `C$` avec le compte administrateur local et un mot de passe vide a longtemps été l'une des failles de conception les plus exploitées de Windows 2000 (cf. Top 10 du SANS).

- L'utilisateur "anonyme" n'appartient plus au groupe "tout le monde".
  - Ainsi les connexions anonymes ne peuvent pas être utilisées pour se connecter aux partages réseau, dont les permissions par défaut sont "tout le monde : lecture seule".
- La permission par défaut sur les nouveaux partages n'est plus "tout le monde : contrôle total" mais "tout le monde : lecture seule".
- L'énumération via une connexion anonyme des comptes et groupes est désactivée dans la configuration par défaut.
  - Cette option est présente depuis Windows NT4 SP4, mais rarement activée auparavant.

3.2 Windows XP : les nouveautés du SP2

Introduction Avec la sortie du SP2 pour Windows XP (prévu pour cet été), Microsoft compte frapper un grand coup. Ce Service Pack ne corrige pas seulement des bogues, mais ajoute également de nombreuses fonctions de sécurité, et active par défaut des options de sécurité généralement inutilisées. L'esprit de ce Service Pack est de bloquer à la source toutes les techniques utilisées jusqu'ici pour attaquer un système Windows. Par exemple :

- Vers de type Code Red, Slammer, Blaster.
- Firewall intégré en mode "deny all" sur les connexions entrantes.
- Spam via le service d'affichage des messages.
- Désactivation des services "alerter" et "messenger" par défaut.
- Bogues IE non corrigés.
- Restriction de la zone poste de travail (cf. outil QwikFix), blocage des popups....

Le document décrivant les nouveautés du SP2 fait à lui seul plus de 150 pages, on peut le trouver à l'adresse suivante : `http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/winxpsp2.mspx`

La "liste à la Prévert" de ces nouveautés est la suivante (il est bien entendu impossible de rentrer dans les détails ici, le lecteur curieux est invité à se reporter au document Microsoft et aux analyses tierces publiées sur Internet) :

- Protection réseau
  - Services "Alerter" et "Messenger" désactivés par défaut.
  - Support Bluetooth natif.
  - Ajout des fonctions "rechercher..." et "sélectionner des utilisateurs, des ordinateurs ou des groupes" aux outils d'administration
  - Restrictions COM / DCOM / RPC
    - Désactivation de l'accès anonyme par défaut.
    - Journalisation accrue.
    - Granularité des permissions accrue.
  - Redirecteur WebDAV.
    - "Basic auth" interdit sur HTTP.
- Support du flag NX.
  - Processeurs AMD64 et Itanium uniquement.

  - Désactivable dans le panneau de configuration (globalement ou par application).
- API "AES" (Attachment Execution Service).
  - Point d'entrée pour un filtrage antivirus.
  - Commune à IE / OE / Messenger.
- Windows Messenger.
  - Blocage des fichiers "dangereux" envoyés par des inconnus.
  - Nickname obligatoirement différent de l'adresse email.
- Outlook Express.
  - Lecture en texte par défaut (rendu RTF au lieu de HTML).
  - Pas de téléchargement du contenu HTML externe.
- Wireless Provising Service.
  - Envoi de paramètres de configuration par les hotspots...
  - Le Wireless Network Registration Wizard permet de donner son numéro de carte bleue aux opérateurs WiFi OEM...
- Windows Media Player.
  - Installation obligatoire de Media Player 9.
- Maintenance
  - Les correctifs de sécurité n'apparaissent plus dans ajout/suppression de programme.
  - Client "Windows Update v5" / "Microsoft Update".
  - Calcul du RSoP.
  - "Security Center"
    - Alerte l'utilisateur sur les fonctions de sécurité suivantes : antivirus, firewall, mises à jour.
  - Windows Installer 3.0

Les deux composants les plus fondamentalement modifiés par le SP2 sont ICF (le Firewall intégré de Windows XP) et IE.

ICF Le produit ICF devient un Firewall personnel à part entière, capable de rivaliser avec des produits du commerce sauf sur un point (et non des moindres) : ICF est incapable de filtrer les connexions sortantes.

En revanche il bénéficie logiquement d'une meilleure intégration avec Windows que ses concurrents, et peut être configuré soit par script, soit de manière centralisée par GPO.

La principale nouveauté est la configuration par défaut en mode "deny all" sur les connexions entrantes pour les postes en Workgroup. Lorsque le poste est joint à un domaine, seul le sous-réseau local est autorisé à établir des connexions entrantes.

ICF intègre également d'une API permettant aux applications "ICF-aware" d'ouvrir dynamiquement des ports. Cette API comprend des fonctions telles que *INetFwAuthorizedApplication* (s'ajouter à la "white list"), *INetFwOpenPort* (ouvrir un port en écoute), *INetFwProfile* (modifier les paramètres globaux du Firewall).

Ceci représente bien évidemment une voie royale pour le code malveillant, aussi quelques restrictions ont-elles été apportées par Microsoft, dont les suivantes :

- Seuls les processus exécutés sous les comptes LocalSystem, LocalService ou NetworkService pour accéder à l'API *INetFwOpenPort*
- Dans tous les cas, SVCHOST ne peut pas accéder à l'API.

Enfin des problématiques spécifiques ont été traitées avec des "astuces" telles que :

- Traitement spécial des RPC et du problème des ports dynamiques : la clé *PrivilegedRpcServerPermission* permet de lister les applications hors "white list" qui peuvent néanmoins effectuer des appels RPC.
- Traitement des requêtes UDP : la réponse est attendue pendant 90 secondes, avant fermeture du port.
- Traitement des broadcasts et des multicasts UDP : la réponse est attendue pendant 3 secondes avant fermeture du port.

Par défaut, ICF bloque efficacement les attaques réseau "frontales" contre un utilisateur domestique en Workgroup. Il s'avèrera probablement inefficace contre du code malveillant exécuté directement sur le poste de l'utilisateur (ouvrant par exemple une porte dérobée), et les nombreuses possibilités de contournements prévues par conception laissent entrevoir des scénarios d'attaque potentiels.

IE Les nouveautés dans IE sont également très nombreuses et vont bien au-delà de la simple correction de bogues. Rien de révolutionnaire toutefois, puisque Microsoft intègre principalement des technologies déjà éprouvées (ex. outil QwikFix, blocage des popups).

Parmi les principales nouveautés on peut citer :

- La barre d'information, affichant les alertes de sécurité pour la page en cours.
- Une seule popup de sécurité par page et non plus par composant.
  - Popup de sécurité = popup de type "voulez-vous exécuter les contrôles ActiveX sur cette page ?"
- Gestion facilitée des "add-ons". Les add-ons (souvent utilisés par des codes malveillants de type Spyware pour s'installer de manière résidente) comprennent :
  - Les plug-ins de navigation (de type "Google Bar").
  - Les "binary behaviors" (extensions de rendu HTML).
- Blocage des popups.
- Nouvelles "Feature Control" (options de sécurité).
  - "MIME sniffing" : reconnaissance des types de fichier par signature et pas par extension. La base de signatures utilisée, et les moyens de mise à jour, restent inconnus à ce jour.
  - Pas d'exécution dans un contexte plus privilégié que l'URL de base.
  - Limites sur la création et le déplacement des fenêtres par script (permet d'éviter les fenêtres masquées ou affichées hors écran).

Conclusion Il est difficile de tirer des conclusions définitives sur le SP2 de Windows XP car à la date de rédaction de cet article, seule une RC1 est disponible.

On notera toutefois un gros effort d'amélioration de la sécurité Windows de la part de Microsoft, salué par la presse spécialisée américaine.

De nombreuses critiques ont également été formulées par des experts reconnus lors de la conférence RSA 2004, au cours de laquelle Bill Gates en personne s'est déplacé pour présenter le SP2. Ces critiques sont les suivantes :

- Ajout de fonctionnalités (y compris de sécurité) = ajout de failles, le nombre de défauts dans le code étant directement proportionnel à la taille du code.
- Pas de changements fondamentaux dans l'architecture Windows, qui conserve les protocoles RPC, DCOM, SMB... et toutes les failles attenantes (ports dynamiques, accès anonymes, configuration complexe...).
- Fonctionnement en mode "pompier" : les protections apportées se basent sur les failles exploitées, pas sur les failles exploitables. Par exemple rien n'est prévu à l'heure actuelle pour lutter contre l'"API Hijacking", car cette technique commence à peine à être utilisée par les codes malveillants. Ainsi Bill Gates a raison en disant que "les virus et les hackers rendent Windows plus robuste"[4].
- La nouvelle configuration par défaut est très orientée "utilisateurs domestiques", mais inadaptée pour une entreprise.
- Microsoft concurrence clairement des produits commerciaux, malgré les nombreux rappels à l'ordre de la justice pour abus de position dominante. A l'heure actuelle il n'est pas prévu d'intégrer un antivirus dans le SP2, mais on se souviendra que Microsoft a racheté la société GeCAD et son produit antivirus RAV.
- Le gain sera nul si l'impact sur les applications est trop fort, car toutes les nouvelles fonctions de sécurité seront alors désactivées par les administrateurs pressés ou par les éditeurs d'applications.

En conséquence, on peut prévoir à la date de rédaction de cet article et sur la base de la version RC1 que le déploiement du SP2 s'annonce problématique. Il est impératif d'effectuer des tests de compatibilité exhaustifs avant tout déploiement massif et de prévoir la désinstallation du SP2. Le gain en sécurité obtenu par l'installation du SP2 est néanmoins important et justifie ces contraintes.

4 Vers une informatique de confiance ?

Personne ne sort réellement victorieux de la guerre entre l'attaque et la défense des systèmes Windows. A chaque nouvelle annonce de Microsoft succède une nouvelle classe d'attaque ou une nouvelle épidémie virale, qui remet en question les efforts entrepris.

Résultats : la faillibilité du système se banalise, la confiance des utilisateurs s'érode et le développement de la société numérique s'en trouve retardé, sans

[4] `http://www.theregister.co.uk/content/55/35145.html`

parler des conséquences économiques immédiates pour les entreprises infectées ou obligées d'investir une part importante de leur budget informatique dans le maintient de l'existant.

Face à cette situation, voyons maintenant quelles sont les pistes explorées actuellement dans le domaine de la protection des systèmes Windows, censées donner un avantage durable à la défense sur l'attaque.

4.1 Prochaines pistes de recherche

Visual Studio 2005, nom de code "Whidbey" La prochaine version du compilateur phare de Microsoft devrait intégrer, sous une forme qui reste à définir, des technologies permettant (enfin) de développer facilement des applications respectant le principe du moindre privilège, c'est-à-dire ne nécessitant pas que l'utilisateur soit administrateur du poste.

Ceci serait bien évidemment une avancée majeure dans le domaine de la sécurité Windows, puisque pour le moment les applications "héritées" et les services nécessitent des privilèges maximum (administrateur ou SYSTEM) pour fonctionner.

Analyse de code source Dans le cadre de la démarche "Microsoft Security Initiative", Microsoft a développé en interne des outils d'audit de code appelés PREfast et PREfix.

Ces outils, déjà disponibles dans le DDK Windows 2003, seront probablement intégrés en standard dans la prochaine version du compilateur Visual Studio.

D'autre part Microsoft confirme sa volonté d'ouvrir de plus en plus largement son code source aux gouvernements, afin sans doute de ne pas se laisser distancer par son concurrent Linux sur l'important marché des administrations.

Aux Etats-Unis où le gouvernement durcit ses exigences de certification Critères Communs pour les fournisseurs de logiciels, Microsoft va sans doute être obligé de re-certifier Windows à chaque nouveau Service Pack.

Tout ceci concourre à une probable amélioration de la qualité générale du code source, bien que le nombre de vulnérabilités triviales identifiées ces derniers temps par des tiers n'ayant pas accès au code source puisse laisser dubitatif ...

Autres idées On retiendra également les idées suivantes, issues du monde Unix ou du Logiciel Libre, qui pourraient présenter un intérêt certain appliquées au monde Windows. Toutefois à ma connaissance Microsoft ne travaille pas officiellement sur ces sujets.

- *Protection de type PaX / GRSecurity.*- Bien que le support matériel des pages non exécutables soit intégré dans Windows XP SP2, cette protection nécessite un processeur AMD 64 ou Itanium. Compte tenu de la vitesse de renouvellement du parc matériel, de nombreux clients ne pourront pas bénéficier de cette protection avant un certain temps. Or des solutions de protection logicielle existent, implémentées sous Windows dans les produits peu connus SecureStack et Overflow Guard.

GRSecurity intègre également des protections simples à mettre en ?uvre telles que l'allocation d'adresses aléatoires pour la pile et les exécutables. L'efficacité de ces protections contre les attaques simples n'est plus à démontrer.

- *Séparation des privilèges.*- Bien que Windows dispose d'un mécanisme d'impersonation via les API *Impersonate*()* et *RevertToSelf()* qui n'a rien à envier au *setuid()* Unix, ce mécanisme est trop peu utilisé par les développeurs, y compris chez Microsoft. Il s'agit là à mon avis d'une "culture sécurité" différente entre les développeurs Windows et Unix.
  La plus belle illustration de ce risque est une Shatter Attack élégante qui a été documentée récemment sur Internet :
  - En tant qu'utilisateur non privilégié, introduire un virus quelconque sur un poste : le service antivirus se déclenche et affiche une "popup".
  - Faire F1 pour obtenir l'aide.
  - Ouvrir le menu "Aller à l'URL..." et désigner "CMD.EXE".
  - CMD est exécuté avec les privilèges de son parent, c'est-à-dire SYSTEM.
- *Chroot.*- Il n'existe pas aujourd'hui de technologie fiable sous Windows permettant de limiter la visibilité du système de fichiers pour un processus donné.
  C'est ainsi que la première recommandation de tout guide d'installation IIS est de mettre la racine Web sur un disque différent du disque système, afin de limiter les conséquences d'une attaque en Directory Traversal, ce qui reste avouons le un "bricolage".
  Une telle technologie trouverait de nombreuses applications pratiques, par exemple dans les fermes de Terminal Server. A l'heure actuelle les "masquages" de disques ou de répertoires sont basées sur la "bonne volonté" de l'explorateur de fichiers, qui effectue lui-même les contrôles d'accès. N'importe quel explorateur tiers se rie de ces restrictions.

4.2 Sécurisation de l'environnement

Malgré toutes les protections vues précédemment et les efforts de R&D considérables, les utilisateurs continuent de cliquer sur les pièces jointes et de télécharger du contenu pirate sur Kazaa.

Windows reste donc un système exposé et vulnérable, principalement sur le poste client. Ce risque est particulièrement accru par la mobilité des postes (PC portables connectés sur des réseaux domestiques ou WiFi) et des données (clés USB, interfaces avec des téléphones portables).

Partant du constat que le poste client représente un risque, le concept de "défense en profondeur" fait son chemin dans les schémas de protection d'entreprise, et des solutions de contrôle externe commencent à apparaître.

On peut citer les annonces Cisco et Checkpoint sur des équipements réseau "intelligents" qui vérifient la configuration des postes et en particulier la mise à jour de l'antivirus. De son côté Microsoft n'est pas en reste avec une technologie de quarantaine des nomades native dans Windows 2003 Server.

Cette implémentation reste sommaire, puisque basée sur l'exécution d'un script côté client qui renvoie au service de quarantaine une valeur booléenne indiquant la conformité du poste par rapport aux paramètres testés. Ces technologies représentent néanmoins un axe de développement futur important.

4.3 Sécurité matérielle

Le support du drapeau NX dans les processeurs IA-64 et AMD 64 démontre une collaboration étroite entre Microsoft et les principaux fondeurs de microprocesseurs, indispensable au succès des opérations futures que sont TCPA et NGSCB.

Intel a déjà annoncé sa volonté d'intégrer nativement un support TCPA dans ses processeurs. Reste l'inconnue NGSCB, dont le contour reste pour l'instant flou malgré les premières présentations publiques de Microsoft.

Sans rentrer dans la polémique, la date de pénétration sur le marché de ces technologies, le comportement des acheteurs et le gain réel en sécurité pour l'utilisateur final sont bien mal cernés aujourd'hui.

4.4 Les nouveaux risques

Dans ce dernier chapitre, je vais m'exercer au jeu dangereux de la prospective. En effet l'explosion des technologies va de pair avec l'apparition de nouveaux risques pour lesquels personne ne se pose encore la question de protections éventuelles, jusqu'à la première catastrophe. Parmi tous ces risques j'ai choisi de mettre l'accent sur les suivants :

- *La mobilité.*- Bien que le problème des postes nomades commence à être pris en compte après des coups durs (vers Slammer et Blaster entre autres), de nombreux autres aspects de la mobilité du code sont totalement occultés.

  L'exemple le plus frappant est l'explosion des applications pour téléphones portables grâce à la standardisation de la plateforme Java (MIDP). Les capacités des nouveaux téléphones en font des nomades à part entière (augmentation de la capacité de stockage à plusieurs Mo, connectivité Internet via GPRS, synchronisation automatique avec Outlook), quand ils ne sont pas directement équipés de Windows CE.
- *Intégration de plus en forte d'Internet dans les applications Microsoft.*- Il est indéniable qu'une grande partie de la stratégie de Microsoft tourne autour de l'osmose entre le poste de travail et les serveurs Microsoft : activation des produits, aide en ligne, compte Passport, aide et cliparts Office 2003 en ligne...

  Une connexion Internet rapide et permanente devient quasiment un prérequis pour pouvoir bénéficier pleinement des produits Microsoft. Par voie de conséquence, le poste est également de plus en plus exposé aux attaques provenant d'Internet.

- *Encapsulation des flux.*- Parallèlement à l'intégration très forte de la composante Internet dans les produits Microsoft, on notera une fâcheuse tendance à encapsuler les flux sur HTTP y compris à travers des extensions non normatives de ce protocole.
  Compte tenu de l'absence de Firewall applicatif intégrant ces extensions (sauf dans le produit Microsoft ISA Server), il devient difficile de filtrer efficacement les flux entre les postes clients du réseau interne et Internet. L'exemple le plus critique est l'encapsulation RPC sur HTTP (utilisé par OWA), compte tenu des nombreux bogues découverts dans les services RPC ces derniers temps.
- *Chevaux de Troie furtifs.*- Pour faire face au développement des logiciels de protection de type Firewall personnel, tout en répondant à la demande en matière de marketing en ligne, de plus en plus d'éditeurs de Spyware (voire de créateurs de virus) intègrent des techniques jusque là considérées comme offensives ("API Hijacking", masquage de processus, injection dans des processus autorisés...) - voir la longue liste des "Trojan.Downloader.xx" référencés chez les éditeurs d'antivirus.
  Il existe encore peu d'outils de protection efficaces contre ces "malwares" qui sont proches de "rootkits", donc difficiles à détecter par définition. Seule une analyse "à froid" du disque permet de conclure de manière fiable sur l'infection d'un poste.
- *Les risques applicatifs.*- Avec l'explosion des services Web, les bogues applicatifs "classiques" (injection SQL, cross-site scripting, etc.) deviennent une réelle menace et l'une des premières causes de "défiguration" (defacement) de sites. Cette menace va aller croissante compte tenu de l'extension des technologies Web (XML, SOAP, ASP.NET, etc.).

Aucune des technologies précédentes (de type protection logicielle ou matérielle contre les "buffer overflow") ne protège contre ces risques, puisque nous sommes en présence de langages interprétés (Java, .NET, XML, scripts, etc.) et que le défaut se situe dans la logique intrinsèque de l'application - chose beaucoup plus difficile à détecter pour un outil.

5 Conclusion

Depuis quelques années, la sécurité logicielle est devenue un enjeu majeur dans un monde de plus en plus connecté. Windows, souffrant de tares de conception et d'une communauté d'utilisateurs peu sensibilisés aux problèmes de sécurité, a été victimes de nombreuses attaques fortement médiatisées compte tenu de sa pénétration importante du marché.

Face à cet état de fait, Microsoft a lancé plusieurs grands chantiers de sécurisation du système, dont le dernier en date (le SP2 pour Windows XP) va sortir cet été. Les bénéfices de chantiers précédents sont techniquement sensibles, mais n'ont eu qu'un faible impact pour l'utilisateur final submergé par les attaques de grande envergure qui continuent à se succéder (ver Blaster, virus Mimail, Bagle et Netsky).

Bien qu'un réel effort de sécurisation soit entrepris, l'accroissement des fonctionnalités et de la connectivité des systèmes Windows augmente parallèlement les risques.

L'équation à résoudre n'est pas simple puisque le système Windows est à la fois :

- Versatile :
  - Produits quasiment identiques en versions grand public, postes de travail, serveurs Web, contrôleurs de domaine...
- Intégré :
  - Les applications et les services exigent des privilèges important pour fonctionner.
  - La frontière entre le système et les applications est très floue.
  - La frontière entre le système et Internet de plus en plus également.
- Ouvert :
  - Ce système est leader du marché.
  - La diversité applicative énorme.
  - La communauté des développeurs n'est pas sensibilisée aux problèmes de sécurité.
- Hérité :
  - Le besoin de compatibilité protocolaire impose l'utilisation de protocoles peu sûrs.
  - Le besoin de compatibilité avec le parc logiciel impose également une configuration par défaut moins sécurisée.
  - Les nouvelles fonctions de sécurité doivent être comprises pour être utilisées par les développeurs.

Références

1. Interview David Aucsmith, *"We have never had vulnerabilities exploited before the patch was known"*. `http://news.bbc.co.uk/1/hi/technology/3485972.stm`
2. Interview Bill Gates, *"Les virus et les hackers rendent Windows plus robuste"*. `http://www.theregister.co.uk/content/55/35145.html`
3. Outil QwikFix, `http://www.pivx.com/main.html`
4. Produit SecureStack, `http://www.securewave.com/news/pr/pr019.html`
5. Produit Overflow Guard `http://datasecuritysoftware.com/`
6. SP 2 Preview, `http://www.microsoft.com/sp2preview`
7. Documentation officielle du SP2. La page du SP2 : `http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/winxpsp2.mspx`
8. ICF, `http://www.microsoft.com/downloads/details.aspx?FamilyID=4454e0e1-61fa-447a-bdcd-499f73a637d1&DisplayLang=en`
9. Windows XP Service Pack 2 White Paper Overview, `http://download.microsoft.com/download/6/6/c/66c20c86-dcbe-4dde-bbf2-ab1fe9130a97/windows%20xp%20sp%202%20white%20paper.doc`

10. Analyses tierces du SP2 : Paul Thurrott sur http://www.winsupersite.com/reviews/windowsxp_sp2_preview2.asp - Steve Friedl sur http://www.unixwiz.net/techtips/xp-sp2.html et Nicolas Ruff sur http://www.ossir.org/windows/supports/2004/2004-04-05/OSSIR-20040405%20-%20Nouveaut%E9s%20du%20SP2.pdf

Filtrage de messagerie et analyse de contenu

Philippe LAGADEC

DGA/CELAR
Philippe.Lagadec@dga.defense.gouv.fr

Résumé La messagerie est un des services les plus utilisés sur Internet et sur les réseaux d'entreprise. L'ouverture de ce service vers Internet nécessite un filtrage efficace pour se prémunir des nombreux risques inhérents au protocole de messagerie SMTP/MIME : virus, vers, contenus actifs, vulnérabilités des clients de messagerie et des serveurs, chevaux de Troie, usurpation d'identité, relayage, spam, mailbombing...Cet article présente les objectifs d'un filtrage de messagerie sécurisé, ainsi que les différentes techniques généralement utilisées, dont l'analyse de contenu. Il détaille ensuite les problèmes et les limites de ces techniques de filtrage, en apportant quelques exemples concrets comme le filtrage de scripts dans HTML ou de macros dans les documents Office, le "zip de la mort", le filtrage par nom ou par contenu. Le cas particulier des webmails est aussi abordé. En conclusion les techniques de filtrage actuelles sont imparfaites mais nécessaires, et quelques pistes d'amélioration sont proposées. Cet article est complémentaire de celui publié pour SSTIC03 à propos des formats de fichiers et du code malveillant.

1 Introduction

La messagerie est aujourd'hui un des services les plus utilisés sur Internet et sur les réseaux d'entreprise. Elle permet une communication à la fois rapide, asynchrone et bon marché, en complément du téléphone, du courrier postal et du fax. C'est également un moyen simple et universel d'échange de fichiers. Ce service devenu incontournable comporte cependant de nombreux risques en termes de sécurité informatique. Cet article dresse tout d'abord un bilan général des risques encourus, et présente certaines solutions de protection classiques, en particulier l'analyse de contenu. Il a pour but de montrer les limites actuelles de ces solutions techniques. Pour cela, on se place dans le cas général d'un réseau local connecté à Internet, dont la passerelle d'interconnexion offre un service de messagerie. Le réseau interne comprend au moins un serveur de messagerie, ainsi que des postes clients. Les utilisateurs se servent d'un logiciel client de messagerie pour émettre et recevoir des messages en passant par le serveur interne.

2 Les risques liès à la messagerie SMTP

Les risques liés au service de messagerie sont nombreux, et touchent tous les éléments qui constituent l'architecture de messagerie : réseau, serveurs, clients,

systèmes d'exploitation, applications, utilisateurs. Ces risques concernent essentiellement le protocole SMTP [1]), sur lequel s'appuie le service de messagerie standard d'Internet, ainsi que ses extensions (par exemple MIME pour les pièces jointes [2]. Les paragraphes suivants rappellent succinctement la nature de ces différents risques :

2.1 Virus et Vers

Les virus et les vers constituent un des principaux fléaux d'Internet. Selon la définition couramment utilisée, un virus se présente sous la forme d'un fichier, qui infecte la machine du destinataire si jamais celui-ci l'exécute. Il peut alors se camoufler ou s'attacher à certains fichiers présents pour se reproduire, et sa diffusion est essentiellement assurée par l'utilisateur lorsque celui-ci transmet les fichiers infectés à d'autres machines. Un ver est quant à lui capable de se diffuser tout seul, généralement sans action de l'utilisateur, en se servant simplement des mécanismes ou des vulnérabilités de la machine pour en infecter d'autres. Dans le domaine de la messagerie, on voit apparaître chaque semaine de nouvelles infections, qui correspondent souvent à des hybrides virus/vers. Ceux-ci se présentent généralement sous la forme de messages contenant une pièce jointe infectée. Selon les cas, celle-ci peut être automatiquement exécutée à la visualisation du message, ou bien l'utilisateur est simplement invité à l'ouvrir. Une fois lancée, cette pièce jointe peut se comporter comme un virus pour infecter le système, et comme un ver pour se répliquer toute seule par messagerie en envoyant des messages infectés à toutes les adresses collectées sur la machine. Elle peut également déclencher une charge utile portant atteinte à la machine. La messagerie est aujourd'hui un des vecteurs les plus courants pour ce type d'infection, en raison de sa simplicité, de son efficacité, et de ses lacunes en termes de sécurité. Les vers les plus rapides se répandent dans le monde entier en moins d'une journée. La plupart des réseaux uniquement protégés par un antivirus sont donc très vulnérables au début de chaque nouvelle infection, dans l'intervalle de temps ou les signatures de l'antivirus n'ont pas été mises à jour.

2.2 Messages non sollicités, Spam

Un spam est un message commercial reçu par un utilisateur sans qu'il l'ait sollicité, souvent envoyé en masse par des logiciels automatisés. Ces logiciels utilisent des listes d'adresses de messagerie collectées sur les sites web et dans les newsgroups, ou achetées à des fournisseurs d'accès peu scrupuleux. Certains virus récents ont même été conçus pour la collecte de telles adresses ainsi que pour le relayage de spam. De nombreuses études annoncent que début 2004 la proportion de spam dans les messages échangés sur Internet atteint environ 50%, ce qui en fait un problème majeur pour tous les utilisateurs du réseau. Les deux conséquences principales du spam sont la réduction des ressources informatiques (bande passante du réseau et performances des serveurs), ainsi que la perte de temps pour les utilisateurs (lecture et suppression des spams).

2.3 Usurpation d'identité

Le protocole SMTP standard ne fournit pas de moyen d'authentifier de façon sûre l'émetteur d'un message. Il est ainsi très simple de forger un message SMTP en se faisant passer pour n'importe quel émetteur, que celui-ci existe ou non. Cette possibilité est très souvent utilisée par les virus et les spammeurs afin de camoufler la source réelle de l'envoi.

2.4 Pièces jointes, chevaux de Troie, spyware

Le protocole MIME ([2]) est principalement utilisé pour composer des messages riches, pouvant contenir du texte au format HTML ainsi que des fichiers en pièces jointes. Un fichier en pièce jointe à l'apparence anodine peut très bien contenir du code malveillant, qui s'exécute lorsque l'utilisateur ouvre le fichier ([3]). Ce code peut porter atteinte à la sécurité du poste et des données de l'utilisateur sans que ce dernier s'en rende compte. Cela peut aboutir par exemple à la prise de contrôle de la machine à distance (cheval de Troie), ou à la fuite d'informations privées ou confidentielles vers Internet (spyware).

2.5 Contenus actifs

Lorsque le corps d'un message est au format HTML, il est possible d'y inclure des scripts tout comme dans une page HTML sur le Web. Ces scripts peuvent être exécutés par le client de messagerie à la visualisation du message (suivant le logiciel utilisé et son paramétrage), et porter atteinte à la sécurité du système s'il n'est pas correctement protégé.

2.6 Vulnérabilités des clients de messagerie

Comme tout logiciel, un client de messagerie peut comporter des vulnérabilités. Si celles-ci concernent les fonctions de décodage ou d'affichage des messages, on peut aboutir à des vulnérabilités exploitables à distance par l'envoi d'un simple message, avec exécution de code en local. Un exemple simple serait un client de messagerie qui provoquerait un débordement de tampon lorsque le sujet d'un message dépasse 1024 caractères. Un attaquant peut alors forger un message avec un sujet très long, et faire exécuter le code de son choix par la machine vulnérable qui recevrait ce message, dans le contexte de l'utilisateur.

2.7 Vulnérabilités des serveurs de messagerie

Un serveur de messagerie peut aussi souffrir de telles vulnérabilités, comme on l'a vu dans le cas de Sendmail en 2003 [4]. Dans ce cas un attaquant peut parvenir à faire exécuter du code directement sur le serveur à l'aide d'un simple message SMTP.

2.8 Relayage

La plupart des serveurs de messagerie peuvent être configurés pour jouer le rôle de relais de messagerie ("*open relay*"). Dans ce mode de fonctionnement un serveur accepte tous les messages qui lui sont adressés, même si l'adresse du destinataire ne correspond pas à son domaine local. Il renvoie alors ces messages vers le serveur du domaine destinataire (enregistrement MX du DNS de ce domaine). Si un serveur est configuré de cette façon comme relais ouvert sur Internet sans contrôle particulier, il peut être utilisé par les spammeurs ou les créateurs de virus pour envoyer des messages en camouflant leur adresse source. Il est alors beaucoup plus difficile de retrouver leur trace. De plus, le relayage peut être considéré comme une complicité dans le cas d'une attaque basée sur la messagerie.

2.9 Mailbombing

Le mailbombing est une attaque en déni de service qui consiste à émettre un grand nombre de messages afin de bloquer la boîte aux lettres d'un utilisateur ou bien de saturer les ressources d'un serveur.

2.10 Web bugs

Un "*web bug*" consiste à exploiter l'affichage d'un message au format HTML dans un client de messagerie pour tracer à distance l'ouverture du message sans que l'utilisateur s'en aperçoive. Il s'agit généralement d'une référence dans le code HTML du message vers une image qui se trouve sur un serveur Web, ce serveur web étant administré par l'émetteur du message. Lorsque le client de messagerie affiche le message, il télécharge automatiquement l'image indiquée, et le serveur web est ainsi prévenu que le message est en cours de visualisation, avec diverses informations supplémentaires comme l'adresse IP du client ainsi que la nature de son logiciel d'affichage. C'est donc un moyen très efficace pour un spammeur, qui a toujours besoin de valider les adresses de messagerie qu'il utilise. Cela permet également de contrôler l'efficacité d'un mailing commercial, de façon plus sûre et plus directe qu'un accusé de réception.

3 Les objectifs d'un service de messagerie sécurisé

Compte tenu de tous les problèmes cités précédemment, un réseau utilisant la messagerie doit se protéger pour réduire les risques au minimum. Le niveau de protection et les contraintes associées doivent être adaptés aux besoins de sécurité du système protégé, notamment en termes de disponibilité, de confidentialité et d'intégrité. Bien sûr, toute mesure de protection doit aussi être compatible avec les besoins des utilisateurs pour que le service soit opérationnel.

Voici une liste non exhaustive des objectifs à atteindre pour assurer la sécurité "idéale" d'une messagerie connectée à Internet :

- Les virus et les vers doivent être détectés et bloqués avant qu'ils puissent atteindre les machines internes, en minimisant leur impact sur les performances du service.
- Le système doit aussi être protégé lorsqu'une nouvelle infection débute, et que les signatures de l'antivirus ne sont pas encore mises à jour.
- Les messages de spam doivent être détectés et bloqués au plus tôt, en minimisant le temps perdu pour l'utilisateur, ainsi que les ressources utilisées au niveau du serveur.
- Un utilisateur doit être protégé contre les messages ou les pièces jointes contenant du code potentiellement dangereux.
- Un message provenant de l'extérieur ne doit pas pouvoir être pris pour un message interne, afin d'éviter l'usurpation d'identité.
- Le service de messagerie ne doit pas pouvoir être utilisé comme relais par un tiers pour camoufler ses envois.
- Si le besoin en disponibilité est important, le système doit être protégé contre les attaques de type mailbombing.
- Les vulnérabilités découvertes dans les serveurs et les clients de messagerie doivent être corrigées au plus tôt.

La plupart de ces objectifs peuvent être atteints en mettant en place un filtrage de messagerie efficace. Il est souvent nécessaire de compléter ce filtrage par des mesures organisationnelles complémentaires, comme la sensibilisation des utilisateurs et le suivi des correctifs publiés pour les divers logiciels employés.

Pour que le service soit opérationnel, les mesures de sécurité doivent respecter les besoins fonctionnels d'une messagerie connectée à Internet et les besoins des utilisateurs :

- Le service doit accepter les messages conformes aux RFC (cf. [SMTP] et [MIME]), mais il doit souvent aussi être compatible avec les très nombreux serveurs qui ne respectent pas à la lettre ces standards.
- Les différents types de fichiers normalement employés par les utilisateurs doivent pouvoir être échangés comme pièces jointes.
- La politique de filtrage doit rester compréhensible par l'utilisateur pour qu'il puisse la respecter et comprendre la raison d'un éventuel blocage.
- Tout message bloqué ou modifié doit donner lieu à une notification claire pour l'utilisateur.
- La modification d'un message ou d'une pièce jointe pour raison de sécurité ne doit pas rendre le message illisible (perte de mise en forme) ou la pièce jointe inexploitable.
- Les faux-positifs (messages bloqués ou modifiés à tort) doivent être réduits au strict minimum, afin d'éviter la perte de données importantes.
- Le filtrage doit respecter les droits des utilisateurs, étant donné que la messagerie est considérée comme une communication privée.

En définitive, garantir la sécurité d'un réseau utilisant un service de messagerie connecté à Internet est loin d'être une tâche simple.

4 Les techniques de filtrage

Pour protéger un système utilisant un service de messagerie connecté à Internet, la méthode la plus employée est le filtrage des messages, en plaçant une passerelle de messagerie en coupure entre Internet et le serveur interne. Cette passerelle peut être constituée d'un simple pare-feu ou bien d'une DMZ plus évoluée. Suivant le résultat des divers filtres mis en oeuvre, un message peut être accepté tel quel, placé en quarantaine ou bien modifié. Les paragraphes suivants présentent différentes techniques de filtrage envisageables.

4.1 Filtrage sur les champs SMTP/MIME : émetteur, destinataire...

Les différents champs utilisés par les protocoles SMTP et MIME permettent un filtrage simple suivant divers critères. Un des filtrages les plus importants consiste à interdire tout message provenant de l'extérieur avec une adresse émetteur correspondant au domaine interne, afin d'éviter l'usurpation d'identité d'un utilisateur local. Ce filtrage est souvent dénommé "antispoofing". De même, tout message entrant doit avoir une adresse destinataire correspondant au domaine interne, afin d'éviter les problèmes de relayage. Pour les messages sortants, il est nécessaire de vérifier que l'émetteur appartient bien au domaine local, et que le destinataire n'en fait pas partie.

4.2 Filtrage sur le texte et l'objet du message

Les logiciels de filtrage de messagerie fournissent souvent des fonctions de détection de mots-clés portant sur le corps ou le sujet du message. Cette technique est essentiellement utilisée pour bloquer les messages comportant un ou plusieurs mots faisant partie d'une liste noire. Elle est efficace contre des spams ou des virus qui utilisent des mots facilement reconnaissables.

4.3 Filtrage des pièces jointes sur le nom et le "content-type"

Un fichier placé en pièce jointe d'un message grâce au format MIME est précédé d'un entête contenant plusieurs informations. Voici un exemple d'entête MIME :

```
Content-Type: application/octet-stream; name="fichier.xyz"
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="fichier.xyz"
```

On y trouve en particulier le nom du fichier (champs "name" et "filename") et une indication sur son type de contenu (champ "Content-Type"). En effet, certains systèmes d'exploitation comme Windows se basent exclusivement sur le nom d'un fichier (ou plutôt l'extension après le dernier point) pour reconnaître son type de contenu, alors que d'autres stockent cette information à part. Ce type de filtrage permet de reconnaître facilement certaines pièces jointes à

risque, comme les exécutables et scripts Windows, qui comportent des extensions connues : EXE, COM, BAT, CMD, SCR, VBS, JS, VBE, JSE...

Il est cependant impossible de garantir un filtrage parfait à partir de l'entête MIME, car ces champs sont purement déclaratifs, et rien ne garantit que l'indication "Content-Type" correspond effectivement au contenu du fichier. Par exemple, de nombreux virus contournent ce filtrage en envoyant un exécutable tout en le déclarant de type "image/gif" ou "audio/x-wav". (Ce qui pouvait provoquer accessoirement l'exécution automatique de la pièce jointe dans d'anciennes versions d'Outlook Express) De plus, on peut constater qu'une même entête MIME contient deux champs possibles pour indiquer le nom du fichier, alors que celui-ci n'en a qu'un lorsqu'il est extrait du message. Comme les clients de messagerie ne prennent pas tous en compte ces 2 champs de la même façon, il est là aussi possible d'en tirer partie pour contourner un filtrage basé sur un des deux champs uniquement. Il est également important de remarquer que certains types de fichiers peuvent être renommés avec des extensions inconnues ou peu utilisées, tout en restant potentiellement dangereux s'ils contiennent du code malveillant. Un exemple répandu est celui des documents Microsoft Office [3].

4.4 Analyse de contenu

Au lieu de se contenter de vérifier le nom d'un fichier, certains logiciels de filtrage en analysent le contenu intégral. Cette approche a pour avantage de détecter la plupart des fichiers renommés, en reconnaissant correctement leur type. Comme il existe de nombreux formats de documents qui peuvent optionnellement contenir du code sous forme de macro-commandes (documents MS Office) ou de scripts (HTML, PDF), l'analyse de contenu permet également de distinguer les fichiers purement statiques de ceux qui pourraient présenter un risque [3]. L'analyse de contenu est aussi employée pour filtrer le code HTML du corps des messages, afin de détecter les scripts. Cela permet en outre de se protéger contre les vulnérabilités exploitées par les virus pour provoquer l'ouverture automatique des pièces jointes infectées dès la visualisation des messages. (exemple des balises < IFRAME > sous Outlook Express)

4.5 Formats conteneurs et analyse récursive

Certains formats de fichiers peuvent contenir d'autres fichiers, qui sont extraits par l'utilisateur lorsqu'il veut les ouvrir. Les exemples les plus connus sont les archives compressées (ZIP, RAR, TAR.GZ, CAB...), mais il en existe beaucoup d'autres qui offrent cette fonctionnalité (documents Office, RTF, PDF, SHS...), comme le montre [3]. Si l'on se contente de filtrer les pièces jointes uniquement sur le nom de fichier, il est très simple de camoufler un exécutable ou un virus dans un fichier conteneur. Certains virus récents ont utilisé cette méthode, ce qui montre qu'un exécutable infecté dans un simple fichier ZIP est toujours efficace face aux barrières actuelles. Pour obtenir un filtrage complet, il est donc nécessaire que l'analyse de contenu soit récursive, en extrayant tous les fichiers trouvés dans les formats conteneurs.

4.6 Filtrage antivirus

Le couplage d'un serveur de messagerie avec un logiciel antivirus est aujourd'hui quasiment indispensable. Chaque message est analysé avec ses pièces jointes, et une base de signatures connues permet la détection de la plupart des virus. Il est important de noter que suivant les fonctionnalités et le paramétrage de l'antivirus, celui-ci n'effectue pas forcément une analyse complète et récursive de tous les formats de fichiers, et en particulier des formats conteneurs. Par exemple, il est très rare qu'un antivirus seul soit en mesure de détecter un exécutable infecté qui serait inclus dans un document RTF.

4.7 Filtrage antispam

Aujourd'hui le spam est un problème crucial pour la messagerie, cependant il n'existe aucune technique de filtrage capable de se protéger de façon complète et sûre. A l'heure actuelle, la meilleure solution est de combiner diverses techniques complémentaires pour distinguer les messages de spam des messages normaux. A la suite de cette détection, les messages considérés comme spams peuvent être directement supprimés, ou bien il est possible d'annoter le sujet du message pour que l'utilisateur puisse faire le tri lui-même. La seconde solution permet d'éviter de perdre un message normal en cas de faux-positif, au prix d'une perte de temps pour l'utilisateur. Voici une liste des principales techniques mises en oeuvre aujourd'hui pour lutter contre le spam :

Liste noire d'émetteurs : Cette technique consiste à maintenir une liste noire d'émetteurs connus pour avoir déjà envoyé du spam, et à refuser tout nouveau message en provenance de ces émetteurs. Il peut s'agir d'adresses individuelles de messagerie (par exemple *adam.smith@provider.com*) ou de domaines entiers (**@provider.com*). Il est possible de s'abonner à des listes (gratuites ou payantes) régulièrement mises à jour sur Internet, pour profiter automatiquement des spams détectés par des communautés d'utilisateurs. La principale limite de cette technique est que les spammeurs utilisent constamment de nouvelles adresses pour émettre, au fur et à mesure que celles utilisées sont placées en liste noire.

Liste noire de serveurs open relay : Comme indiqué précédemment, les spammeurs emploient souvent des serveurs ouverts au relayage afin de camoufler leurs traces. On peut trouver sur Internet des listes noires tenues à jour comme [5], qui indiquent les serveurs de messagerie open relay qui ont été détectés. Pour l'utiliser, chaque serveur de messagerie doit refuser tout message provenant d'un des serveurs open relay placé dans la liste noire.

Vérification DNS : Sur Internet, chaque domaine doit contenir un enregistrement "MX" (*Mail eXchanger*) dans son DNS, pour indiquer l'adresse IP de son serveur de messagerie principal. Lorsqu'on reçoit un message d'un autre serveur,

il est possible de vérifier si l'adresse IP de ce serveur correspond bien à l'enregistrement MX du domaine de l'adresse e-mail de l'émetteur. Par exemple, si mon serveur reçoit un message envoyé par le serveur 192.168.12.34 dont l'émetteur est *adam.smith@provider.com*, il faut vérifier si l'enregistrement DNS MX du domaine provider.com a bien l'adresse IP 192.168.12.34. Cette technique permet de se protéger contre les messages forgés qui correspondent à un domaine existant, et qui ne seraient pas émis par le serveur officiel. Elle oblige cependant à une gestion stricte du DNS et des enregistrements MX, ce qui n'est pas toujours possible. (cas des serveurs de messagerie secondaires)

Filtrage par mots-clés : Comme indiqué plus haut, il est possible de dresser une liste noire de mots-clés qui sont très souvent employés dans les spams ("viagra", "xxx", "porn"...), et très peu souvent dans les messages normaux. Il s'agit donc d'un indicateur simple pour la détection de spams. Cependant cette technique est de moins en moins efficace, car les spammeurs utilisent diverses méthodes de camouflage qui rendent la détection automatique de mots-clés de plus en plus difficile : remplacement de lettres par des chiffres à la forme similaire (zéro à la place de O), insertion de points entre les lettres, insertion de commentaires HTML, de lettres minuscules colorées en blanc... Les logiciels de filtrage antispam doivent être constamment améliorés pour prendre en compte les nouvelles formes de camouflage. Un autre défaut de cette technique est le nombre élevé de faux-positifs et de faux-négatifs, car les spams tendent à ressembler de plus en plus à des messages normaux.

Filtrage Bayesien : Il s'agit d'une technique statistique qui associe des probabilités aux différents mots-clés rencontrés dans un message, à l'aide d'un apprentissage progressif. En combinant les différentes probabilités obtenues suivant la méthode de Bayes, on calcule la probabilité globale qu'un message soit du spam, ce qui permet de le classer. La méthode décrite par Paul Graham [6,7] permet d'obtenir un taux de détection excellent (au-delà de 99%) avec très peu de faux-positifs, après un temps d'apprentissage suffisant. La difficulté de cette technique est l'apprentissage, qui doit être supervisée en désignant manuellement les messages normaux et les spams. De plus, cet apprentissage doit être adapté aux messages normaux reçus par chaque utilisateur pour que le filtrage soit optimal. Les résultats et les contraintes sont donc différentes selon l'implantation du filtre, qui peut être mis en ?uvre sur le serveur (exemple de SpamAssassin [8]) ou bien dans le client de messagerie (exemple de Mozilla [9]).

Filtrage comportemental : En analysant le trafic de messagerie dans le temps, il est possible de distinguer les envois normaux des envois de spam, généralement massifs et automatisés. Cette méthode est par exemple employée dans le logiciel j-chkmail [10], en complément d'autres techniques.

Autres méthodes : Etant donné qu'aucune technique de filtrage ne donne pleine satisfaction, la recherche de nouvelles solutions est très active dans le domaine de la lutte antispam.

5 Les limites du filtrage

Malgré toutes les techniques disponibles pour le filtrage de messagerie, il est actuellement difficile de se protéger de façon complète et sûre contre tous les risques présentés au début de cet article. En effet, chaque technique de filtrage possède des faiblesses et peut être contournée. Voici quelques exemples de problèmes connus.

5.1 Filtrage des types de fichiers par nom ou par contenu

Comme indiqué précédemment, le simple filtrage par nom de fichier ne suffit pas pour détecter toutes les pièces jointes qui pourraient contenir du code potentiellement dangereux. Par exemple, un document Word avec une macro auto-exécutable peut être renommé avec une extension inconnue ".xyz", et contourner la politique de filtrage. De même, l'analyse du contenu des fichiers n'est pas strictement suffisante pour détecter le type d'un fichier. Par exemple, il est parfois difficile d'identifier à coup sûr un fichier XML ou HTML en ne se basant que sur son contenu. D'autre part, certains formats contiennent des marqueurs en début de fichier, d'autres à la fin. Il est ainsi possible de forger des fichiers dont le contenu s'apparente à deux formats distincts. La détection des fichiers "à risque" doit donc obligatoirement combiner la vérification du nom du fichier et l'analyse de son contenu. Il peut être utile de corréler cette analyse avec le champ MIME "Content-Type" pour déceler une éventuelle combinaison anormale.

5.2 Récursivité de l'analyse de contenu

Les formats de fichiers conteneurs permettent souvent de contourner les logiciels de filtrage et les antivirus, par exemple en incorporant un fichier exécutable dans un document RTF ou Word (objet OLE Package). Si l'on veut obtenir une détection complète de tout code exécutable dans les pièces jointes, le filtrage doit se faire de façon récursive dans tous les formats conteneurs autorisés. Il est difficile voir impossible d'obtenir ce résultat à l'aide des logiciels de messagerie disponibles sur le marché actuel, car leur support des formats conteneurs est partiel. La seule possibilité est de limiter les types de conteneurs autorisés (ZIP, TAR.GZ, CAB...), au détriment des utilisateurs, qui ont parfois besoin des formats conteneurs moins connus (par exemple les objets OLE Package).

5.3 Performances

Il est bien sûr évident que l'analyse de contenu récursive est beaucoup plus lourde en termes de ressources qu'une simple analyse sur le nom des fichiers.

Pour certains systèmes cela constitue un problème bloquant, car il faut disposer de ressources matérielles adéquates, et le temps de transit d'un message dans le logiciel de filtrage n'est pas négligeable.

5.4 "Zip of Death"

Le "zip de la mort" est une vulnérabilité qui touche certains logiciels d'analyse de contenu et certains antivirus, et qui aboutit généralement à un déni de service. Pour effectuer l'analyse récursive d'un format conteneur, ceux-ci extraient sur disque les fichiers inclus dans un conteneur, et effectuent alors l'analyse de chaque fichier jusqu'à extraction complète. Si l'on prend un grand fichier rempli de caractères identiques (par exemple des espaces ou des zéros), et qu'on le place dans une archive compressée, on obtient un taux de compression élevé, de l'ordre de 1000 avec le format ZIP voire 1000000 avec RAR. En répétant l'opération plusieurs fois (archives imbriquées), on obtient un fichier de quelques kilo-octets, qui occupe un grand nombre de giga-octets une fois décompressé. Un fichier d'exemple connu et diffusé sur Internet fait ainsi 42 Ko, et contient plusieurs millions de fichiers de 4 Go chacun ! Si le logiciel de filtrage ou l'antivirus effectue la décompression sur disque sans contrôle particulier, cette décompression prend beaucoup de temps en occupant le serveur, et le disque utilisé peut être saturé. Afin d'éviter ce problème classique, il est indispensable de contrôler la taille d'un fichier inclus avant de l'extraire, et de limiter le temps et l'espace disque alloués à l'analyse d'un fichier. Ce contrôle doit être effectué quel que soit le format conteneur analysé.

5.5 Détection et nettoyage de scripts dans HTML, camouflage

De nombreux logiciels de filtrage proposent la détection de scripts dans le corps HTML des messages, ainsi que dans les pièces jointes au format HTML. En présence d'un script, il est possible de bloquer le message, ou bien de le nettoyer en supprimant le code du script tout en maintenant le reste de la mise en forme. Ce nettoyage n'est pas toujours parfait lorsqu'on teste les logiciels du marché, pour diverses raisons :

Emplacement des scripts : Un script peut apparaître sous diverses formes et à divers emplacements dans le code HTML : balises < SCRIPT >, évènements onXXX (par exemple "onLoad" ou "onMouseOver"), URL d'un lien (par exemple `<A HREF="javascript:?">` ou encore `<BASE HREF="javascript:?">`)... Tous ces emplacements ne sont pas toujours correctement pris en compte.

Texte au format Unicode : Un fichier HTML est généralement au format texte ASCII (1 caractère est codé sur 1 octet soit 8 bits), mais Internet Explorer et certains autres navigateurs acceptent d'autres formats comme Unicode UCS-2 (1 caractère sur 16 bits) ou encore UTF-8 (caractères courants sur 8 bits, caractères étendus sur 16 bits ou plus). Certains logiciels de filtrage HTML peuvent être facilement contournés par cette méthode.

Caractères codés : Dans une URL, le format HTML autorise le codage de caractères sous la forme "&#" suivi de leur code ASCII en décimal ou en hexadécimal, puis d'un point-virgule optionnel. Par exemple, le caractère "j" peut être représenté sous les formes "&#106", "j", "&#6A", "&#6a", "&#6A;", "&#00106"... i Comme tous les logiciels de filtrage ne sont pas capables de décoder ce type d'URLs, il est souvent possible de camoufler un script de cette façon :

```
<A HREF="&#106;avascript:alert('ceci est un script')">
```

Scripts imbriqués : Un script peut être nettoyé suivant plusieurs méthodes, par exemple en supprimant simplement son code, ou en le mettant dans un commentaire HTML. Si le logiciel de filtrage supprime simplement le code des scripts en une seule passe, il peut être vulnérable aux scripts imbriqués. Ce problème apparaît lorsque le code suivant est nettoyé :

```
<SCRI<SCRIPT> alert('script 1'); </SCRIPT>PT> /
alert('script 2'); </SCRIPT>
```

Si le script 1 est simplement supprimé, le script 2 apparaît sous une forme correcte et peut alors être exécuté par le client de messagerie ou le navigateur :

```
<SCRIPT> alert('script 2'); </SCRIPT>
```

Cela montre que le nettoyage de scripts par suppression doit être effectué en plusieurs passes, ou qu'un script supprimé doit être remplacé par un code neutre.

Détection et nettoyage de macros Certains logiciels proposent la détection et le nettoyage des macros dans les documents Office. Cependant cette détection ne concerne pas toujours tous les types de documents (Word, Excel, Powerpoint, Access, Project, ?), et elle n'est pas toujours efficace vis-à-vis des versions successives de ces formats de fichiers (Word 6, 95, 97, 2000, XP, 2003...).

Fichiers ou messages découpés Certains formats de fichiers comme les archives compressées (ZIP, RAR, ARJ...) offrent la possibilité d'être découpés en plusieurs parties, afin de ne pas dépasser une taille donnée, par exemple la taille d'une disquette. De même, le standard MIME [2] permet le découpage d'un message SMTP en plusieurs messages partiels, pour ne pas dépasser un seuil donné si la messagerie employée est limitée. Dans le cas d'un message partiel ou d'une pièce jointe découpée, l'analyse de contenu n'est pas toujours possible, car les structures ou les motifs recherchés peuvent être répartis sur des messages indépendants.

Fichiers chiffrés Il existe de nombreux formats de fichiers chiffrés : PGP, GPG, archives compressées protégées par mot de passe (ZIP, RAR, ARJ...), documents Office protégés en lecture par mot de passe (Word, Excel...), ou de

nombreux autres formats moins connus. Dans tous les cas, l'analyse de contenu est impossible. Il est donc nécessaire d'analyser le risque encouru pour tous les formats chiffrés que l'on autorise. Il existe un risque uniquement dans le cas où les utilisateurs disposent des logiciels nécessaires au déchiffrement.

Stéganographie La stéganographie est une technique apparentée au chiffrement, qui consiste à camoufler un fichier (ou toute information) dans les données d'un autre fichier en apparence anodin. Il est par exemple possible de camoufler un exécutable sous forme codée dans un commentaire d'un fichier HTML, ou dans les données d'un fichier image (PNG, JPEG, GIF...) ou d'un fichier audio (WAV, MP3, AU...). Tout fichier de données peut ainsi être utilisé comme conteneur. Néanmoins, le destinataire d'un tel fichier doit être en mesure d'extraire le contenu camouflé pour que cela présente un risque. Cet utilisateur doit donc effectuer une action volontaire voire disposer d'un logiciel spécifique, puisque les systèmes d'exploitation et les applications classiques ne sont pas conçus pour cela. D'un point de vue technique, un logiciel de filtrage de messagerie ne peut détecter le camouflage par stéganographie, à moins que les structures de codage soient connues.

Webmail Lorsque les utilisateurs du réseau interne ont accès à Internet via les protocoles HTTP et/ou HTTPS, ils peuvent utiliser les interfaces Web de nombreux fournisseurs pour envoyer et recevoir des messages personnels. Ces messages sont émis et reçus à partir d'un serveur situé sur Internet, en dehors du réseau interne. Ils ne sont donc jamais analysés par la passerelle de filtrage de messagerie. Lorsqu'un utilisateur reçoit un message avec pièce jointe dans une interface Webmail, il peut extraire les fichiers joints et les importer sur le réseau interne. Le téléchargement s'effectue alors grâce à un des protocoles HTTP, HTTPS ou FTP. Pour obtenir une protection homogène, il est donc nécessaire d'appliquer le même type de filtrage sur les fichiers téléchargés par ces protocoles que sur les pièces jointes des messages reçus par SMTP.

6 Conclusion

Le service de messagerie SMTP présente de nombreux risques pour un réseau connecté à Internet. Si le système contient des données sensibles, il est nécessaire de mettre en place des mécanismes de filtrage efficaces afin de se protéger, après avoir clairement analysé les menaces et défini une politique de sécurité. Il existe sur le marché de nombreux logiciels proposant des fonctions de filtrage de messagerie, avec diverses protections contre les virus, le spam et le code malveillant. Cependant le filtrage de messagerie est relativement complexe en raison des nombreux standards, protocoles et formats de fichiers mis en jeu. Les divers problèmes et limites des techniques de filtrage esquissés dans cet article montrent qu'il est difficile d'obtenir une protection complète et sûre d'un service de messagerie à partir des logiciels disponibles sur le marché. La recherche de nouvelles solutions plus efficaces est donc une necessité.

Références

1. RFC 2821 et 2822 (remplacent les RFC 821 et 822), *Simple Mail Transfer Protocol et Internet Message Format*, `http://www.rfc-editor.org`
2. RFC 2045 à 2049, *Multipurpose Internet Mail Extensions*, `http://www.rfc-editor.org`
3. P. Lagadec, Formats de fichiers et code malveillant, *SSTIC 2003*, `http://www.sstic.org/SSTIC03/interventions03.shtml` Une version mise à jour pour l'OSSIR est disponible sur `http://www.ossir.org/windows/supports/liste-windows-2003.shtml`
4. BUGTRAQ, Sendmail Header Processing Buffer Overflow Vulnerability, `http://www.securityfocus.com/bid/6991`
5. Open Relay Data Base, `http://www.ordb.org`
6. P. Graham, A plan for spam, `http://paulgraham.com/spam.html`
7. P. Graham, Better Bayesian filtering, `http://www.paulgraham.com/better.html`
8. SpamAssassin, `http://spamassassin.org`
9. Mozilla, `http://www.mozilla.org`
10. j-chkmail, `http://j-chkmail.ensmp.fr`

Conférences invitées

L'intelligence économique : une nouvelle politique publique pour répondre à la guerre économique

Monsieur le député Bernard CARAYON**

Député du Tarn
Rapporteur spécial au nom de la Commission des Finances
sur le Budget du Premier Ministre (SGDN et Renseignement)

L'intelligence économique a longtemps été perçue en France comme un concept ambigu, interprété tantôt comme une méthode d'espionnage économique – c'est le côté " barbouzerie " d'officine – tantôt comme une méthode classique d'entreprise au service des seules entreprises : veille commerciale, veille juridique, veille concurrentielle.

Mon rapport rompt avec cette double perspective, tantôt sulfureuse, tantôt simpliste, et tire de l'observation de nos grands concurrents et du succès des méthodes qu'ils ont retenues, des leçons pour l'organisation des pouvoirs publics. L'intelligence économique, loin de se résumer à un débat de " spécialistes " sur le concept, doit faire l'objet d'une approche pragmatique, fondée sur l'observation des relations économiques internationales, des relations entre les Ètats, entre les entreprises ; elle se définit par son contenu : **sécurité économique, compétitivité, influence et formation**.

Elle découle aussi d'une certitude : **les critères de conquête des marchés internationaux ne sont pas toujours ceux que définit l'économie libérale : le prix et la qualité des produits et des services**. On peut concevoir que ces critères soient déterminants pour des marchés " classiques " : l'industrie de la chaussure, l'exportation des petits pois... Mais ce n'est pas le cas des métiers ou des activités " stratégiques ", à l'instar des télécommunications, de l'armement, de l'aéronautique, de la pharmacie, de certains pans de l'industrie agroalimentaire et de quelques autres secteurs qui ne sont pas seulement créateurs de richesses mais aussi sources de puissance et d'influence. L'intelligence économique fait ainsi appel à des **compétences croisées entre le secteur public et le secteur privé**, et ces compétences sont mises au service de la stratégie des entreprises, voire des Ètats.

Aujourd'hui encore, l'intelligence économique n'occupe pas en France la place qu'elle mérite : celle qu'elle occupe précisément dans les grands pays occidentaux... Les Français pêchent depuis longtemps par naïveté. Autant ils sont interventionnistes dans la définition des règles qui régissent l'économie interne, autant ils sont libéraux et naïfs lorsqu'ils analysent les ressorts de l'économie internationale. Chez les Anglo-saxons, c'est exactement l'inverse : interventionnistes à l'extérieur, libéraux à l'intérieur. Une vraie politique d'intelligence économique

** Bernard Caryaon est l'auteur du rapport au Premier Ministre " L'Intelligence Économique, compétitivité et cohésion sociale ", paru à la Documentation française - 2003 (`http://www.ladocumentationfrancaise.fr/brp/notices/034000484.shtml`)

doit être une politique de convergence d'intérêts entre le public et le privé, autour d'objectifs stratégiques.

Le retard de notre pays s'explique ainsi essentiellement par les cloisonnements entre la sphère publique et la sphère privée, mais également par les antagonismes traditionnels que l'on observe entre les administrations publiques françaises. Face à ces relations traditionnelles de défiance, l'intelligence économique va contribuer à réformer les relations entre le monde public et le monde privé, à établir des passerelles, à définir des modes de convergence d'intérêts autour, encore une fois, de grands enjeux clairement identifiés.

Pour mettre un terme à ces cloisonnements, pour rapprocher acteurs publics et privés, je crois qu'il faut d'abord une forte et significative impulsion politique qui ne peut venir que de l'Exécutif. C'est la raison pour laquelle j'ai été très satisfait d'observer que le Premier Ministre avait pris à bras le corps ce dossier et qu'il avait confié à Alain Juillet le soin de coordonner les efforts en la matière. Il est indispensable que se crée ainsi, dès que possible, une plate- forme d'aide et de soutien aux entreprises, pour les grands contrats, de telle manière que nos entreprises qui sont confrontées à une concurrence internationale parfois féroce, puissent bénéficier d'une information mutualisée des administrations publiques mais également des meilleurs experts du secteur privé.

Toutefois, je tiens à souligner que cette nouvelle politique publique n'est pas seulement destinée aux grands groupes. Pour les PMI et les grandes entreprises qui travaillent sur des marchés stratégiques, les besoins et les moyens sont évidemment identiques. Par exemple, entre AIRBUS et ses sous-traitants, il y a des besoins évidemment communs, et les concours de l'Etat sont naturellement les mêmes. Il est clair que le Président de la République n'ira pas défendre de la même manière les besoins d'AIRBUS et ceux de ses sous-traitants. Mais derrière AIRBUS, il y a aussi de nombreuses entreprises qui animent nos territoires – je pense bien sûr à Midi-Pyrénées – et qui sont entraînées, en quelque sorte, par la locomotive d'AIRBUS.

Ces secteurs d'activité ou ces entreprises reflètent, dans le fond, les secteurs qui sont les plus exposés à la concurrence internationale et à l'influence, au sein des organisations internationales où s'élaborent les règles juridiques, les normes professionnelles et parfois les modes, d'acteurs nouveaux et puissants : comme les ONG et les fondations. Ces secteurs d'activité sont source, non seulement de création de richesses, de création d'emplois mais aussi d'influence et de puissance. Derrière tout cela, se jouent la sauvegarde de nos emplois, notre modèle social, notre destin de Français et d'Européens, et en définitive l'avenir de nos enfants.

L'opération "Carbone 14" : les modalités pratiques et les implications juridiques d'une attaque informatique

David Bénichou[1] and Serge Lefranc[2]

[1] Magistrat
Cour d'appel de Paris
Paris CEDEX F-75000, France
David.Benichou@justice.fr
[2] Ingénieur de l'Armement
Centre d'Électronique de l'Armement
Bruz CEDEX F-35174, France
serge.lefranc@dga.defense.gouv.fr

Résumé Cet article s'inspire d'un cas réel et a pour but de présenter les modalités pratiques et les implications juridiques d'une attaque informatique, que ce soit du point de vue de l'attaquant, comme de celui de la victime.

1 Introduction

Cet article a pour but de présenter les modalités pratiques et les implications juridiques d'une attaque informatique du point de vue de l'attaquant et de celui de la victime. Le scénario retenu est volontairement proche d'un cas réel : un éditeur de jeux vidéo se fait dérober en ligne le code source d'un jeu avant sa sortie sur le marché. Celui-ci sera compilé et mis en ligne sur internet.

Les auteurs se réfèrent à un événement relaté par un responsable d'une entreprise d'édition de jeux vidéo [1]. Les développements imaginés et réalisés n'ont aucun lien avec cette affaire au delà de son cadre général, transposable à tout type d'établissement dont des données constituant son patrimoine seraient susceptibles d'être pillées

Le but est de montrer que, dès les stades précoces de la réalisation et la préparation de l'attaque, des contre-mesures d'ordre techniques ou juridiques pourront être mises en oeuvre, soit pour empêcher la réalisation de l'attaque soit pour obtenir des preuves qui permettront la réparation du dommage subi ou l'identification et la condamnation de l'auteur de l'attaque.

Le scénario est le suivant : un éditeur informatique emploie un "pirate" pour nuire à son concurrent. La mission du pirate est de réussir à s'introduire dans le réseau du concurrent pour dérober les codes source de son prochain jeu et d'en assurer la diffusion, avant commercialisation officielle, sur internet. Les techniques présentées dans cet article, et censées reproduire les actions du pirate, ne sont

pas exhaustives. Son but n'est pas de trouver toutes les failles de sa cible comme pourrait le faire un consultant en sécurité, mais juste celle qui lui permettra de réaliser son objectif. Le lecteur intéressé par plus de détails concernant les tests d'intrusions peut se réferrer au numéro 11 du magazine MISC [2].

Pour rendre le texte plus vivant nous appellerons ce jeu "demi-vie" et l'opération "carbone 14".

2 Le "pacte des loups"

2.1 Aspects techniques

Un éditeur de jeu vidéo souhaite affaiblir la réputation de son principal concurrent qui est en train de developper la suite d'un jeu à succés. Il decide donc d'engager un pirate informatique dont la mission sera de derober les sources de ce jeu afin de les mettre sur le marché, même si le jeu n'est pas encore finalisé.

Pour ce faire, le pirate doit être d'une compétence technique reconnue, être connu de l'éditeur malhonnête et être vénal. Ces conditions garantissent le succés de l'opération. Dès lors, il existe deux possibilités :

- soit le pirate est recruté en externe, ce qui suppose certains risques importants (notamment au niveau de la fiabilité...),
- soit il est recruté en interne. Il appartient au personnel de l' éditeur malhonnête et ses intérêts sont communs avec ceux de son employeur.

2.2 Aspects juridiques : le délit d'association de malfaiteurs informatiques

Définition Le "pacte des loups" formé par nos agents conspirateurs est susceptible, dès ce stade et sous la réserve d'être matérialisé par au moins un acte préparatoire, de revêtir une qualification pénale.

En effet, l'article 323-4 du code pénal dispose que :
La participation à un groupement formé ou à une entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'une ou de plusieurs des infractions prévues par les articles 323-1 à 323-3 est punie des peines prévues pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

Rappelons que les articles 323-1 à 323-3 définissent le noyau dur des infractions portant atteinte aux systèmes de traitement automatisé de données (ci-après STAD). S'il s'agit pour nos agents de s'approprier le code source d'un programme dans un système distant, l'article 323-1, définissant l'accès et le maintient frauduleux dans un STAD, aura vocation à s'appliquer.

Pour reprendre un formalisme familier aux développeurs, il convient de remarquer que les éléments constitutifs du délit pourraient s'articuler en une fonction formée de la manière suivante :

```
Association de malfaiteurs informatiques ($agent) =

Participation ($agent) à
{
 ( [un groupement formé] ou [une entente établie] )
 et
 (
   [la préparation d'au moins une infraction entre 323-1 et 323-3]
   et
   [caractérisée par un ou plusieurs faits matériels ]
  )
}
```

L'important, au-delà de l'analogie entre les deux domaines, est que cette approche formaliste de la loi montre qu'on peut aborder le droit pénal avec la même rigueur qu'une fonction booléenne : le plaideur avisé s'efforcera de démonter un élément essentiel qui fera, par le jeu des opérateurs cumulatifs ("et"), basculer la prévention vers le non lieu. Le juge, s'il se fonde sur son intime conviction, est soumis à un "contrôle de cohérence" par la cour de cassation, qui ne manquerait pas d'annuler une décision en cas de contradiction de motifs.

Il faut souligner l'intérêt d'une telle incrimination : la poursuite pourra être engagée avant même le *commencement d'exécution de l'infraction* (domaine de la tentative, art. 121-5 du cpp), c'est-à-dire dès le stade du premier acte préparatoire. Lorsque l'association de malfaiteurs informatiques est formée et a accompli un premier acte préparatoire (par exemple en mettant en commun des moyens) elle entre dans le champ de la répression.

La frontière entre la préparation, caractérisée par un ou plusieurs faits matériels (323-4 cp) et avec le commencement d'exécution de la tentative punissable est ténue, mais d'importance. A partir de la simple tentative (premier essai infructueux par exemple) ou de la commission d'un délit, les auteurs seront dans un cumul idéal d'infractions : pour l'association de malfaiteurs (323-4) et pour les infractions tentées (323-7 du cp) ou consommées (323-1 à 323-3).

Fondements Un petit retour en 1987, (ou *rétro-conception* de la loi) nous permet de sonder la volonté du législateur de l'époque, d'abord sur l'intérêt de créer une telle incrimination, où l'on constate qu'il peut s'agir réprimer les groupes de pirates informatiques :

Or, ces dernières années ont vu fleurir les clubs de passionnés de l'informatique qui échangent les renseignements dont ils disposent et les techniques qu'ils ont mises au point. En conséquence, il a paru opportun à votre commission de

punir les ententes établies en vue de la préparation du délit de piratage informatique [3].

Ensuite, c'est le garde des sceaux lui-même qui nous éclaire sur la relative mansuétude de la répression, puisque le délit d'association de malfaiteurs informatiques n'est pas plus sévèrement puni que les infractions vers lesquelles ces associations tendent :

M. le président : Quel est l'avis du Gouvernement ? M. le garde des sceaux : Le Gouvernement [..] retient l'initiative du Sénat de créer une incrimination d'association de malfaiteurs. En revanche, il est opposé à l'idée de frapper cette incrimination plus sévèrement que l'infraction elle-même. Avec ce sous-amendement, les principes sont respectés, et la répression n'est pas excessive.[4]

Le délit d'association de malfaiteurs en matière informatique est donc distinct de l'association de malfaiteurs prévue par l'article 450-1 du code pénal [5] . La formule générique de l'association de malfaiteurs concerne les crimes et les délit punissables d'au moins 5 ans d'emprisonnement, ce qui n'est pas le cas des atteintes aux STAD pour lesquels le maximum encouru est 3 ans d'emprisonnement.

Similitude avec la "bande organisée" La rédaction de l'article 323-4 rappelle directement la définition de la circonstance aggravante de bande organisée prévue par l'article 132-71 du Code pénal et reprise par la loi du 9 mars 2004 portant adaptation de la justice aux évolutions de la criminalité [6] (ci-après " loi Perben II ") :

Constitue une bande organisée au sens de la loi tout groupement formé ou toute entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'une ou de plusieurs infractions.

La définition du délit prévue par l'article 323-4 du code pénal est identique. On pourrait donc parler, s'agissant de 323-4 d'un délit de "bande organisée en matière informatique". C'est que bande organisée et association de malfaiteurs

[3] P. 60, Rapport n°3 de la commission de lois du Sénat, fait par M. le sénateur Jacques THYRAUD, annexé du procès verbal de la séance du 2 octobre 1987.

[4] Compte rendu analytique de l'assemblée nationale, 3e séance du 21 décembre 1987, Journal Officiel, p. 8026.

[5] Art. 450-1 du CP : Constitue une association de malfaiteurs tout groupement formé ou entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'un ou plusieurs crimes ou d'un ou plusieurs délits punis d'au moins cinq ans d'emprisonnement. Lorsque les infractions préparées sont des crimes ou des délits punis de dix ans d'emprisonnement, la participation à une association de malfaiteurs est punie de dix ans d'emprisonnement et de 150000 euros d'amende. Lorsque les infractions préparées sont des délits punis d'au moins cinq ans d'emprisonnement, la participation à une association de malfaiteurs est punie de cinq ans d'emprisonnement et de 75000 euros d'amende.

[6] Loi n° 2004-204 du 9 mars 2004 art. 12 I Journal Officiel du 10 mars 2004.

recouvrent une même réalité. Cette identité de situation se traduit par une définition identique, mais selon que l'on est dans le champ de la bande organisée (132-71) ou de l'association de malfaiteurs (450-1 et 323-4), le but répressif diffère : la première est prévue comme circonstance aggravante de certaines infractions, la seconde est un crime ou un délit autonome en dehors de tout commencement d'exécution de l'infraction principale.

Inapplicabilité des nouveaux moyens d'enquête La loi Perben II permet d'augmenter les pouvoirs d'enquête pour certains crimes ou délits les plus graves, commis en bande organisée et visés limitativement à l'article 706-73 du code de procédure pénale (durée de la garde à vue à 4 jours, détention provisoire plus longue, opérations d'infiltration, de sonorisation ou encore d'interceptions de télécommunications possibles sans passer par le juge d'instruction).

Les infractions commises en bande organisée, visées à l'article 706-74 du Code de procédure pénale (les "moins graves") ne pourront pas bénéficier de ces nouvelles possibilités techniques et juridiques, mais seront susceptibles d'être traitées par les nouveaux pôles interrégionaux de lutte contre la délinquance économique et financière et de lutte contre la criminalité organisée.

Or, le délit *d'association de malfaiteurs informatiques* est, comme l'indiquait la réclame d'une marque de boisson pétillante, le Canada Dry de la bande organisée : il en a la substance (similitude des termes) mais pas l'étiquette (la définition juridique identique par le biais d'une circonstance aggravante distincte).

Cela n'est pas sans implications juridiques puisque au sens de la loi Perben II, le délit d'association de malfaiteurs informatique est insusceptible de bénéficier des moyens nouveaux ni même de ressortir de la compétence des pôles interrégionaux par le biais l'article 706-74, qu'on pourrait qualifier de "procédure balais" des bandes organisées.

On pourrait voir là un oubli ou un paradoxe de la loi : censée adapter la justice aux évolutions de la criminalité, elle ne s'applique pas aux infractions de haute technologie commises en bande organisée (art. 323-4 du cp). Or s'il est un domaine où la criminalité évolue rapidement, est capable de s'organiser et offre des possibilités de gains et d'impunité importantes, c'est bien celui de la cybercriminalité.

Alors qu'on évoque dans les risques liés à un "cyberterrorisme" pour l'instant abstrait, seul ce point d'entrée (l'article 421-1 du code pénal) permettrait de bénéficier des nouveaux moyens légaux offerts par la loi Perben II.

En conclusion, sur le fond, le "pacte des loups" entrera dans l'illégalité dès lors que les agents auront concrétisé leur projet commun, par exemple en mettant à la disposition du pirate une machine, un bureau, un local ou encore des éléments matériels en vue de la commission de l'intrusion.

Sur la procédure, les enquêteurs et magistrats devront se contenter des dispositifs habituels, ceux qu'on réserve à la délinquance de bas niveau, ou inorganisée, ce qui malheureusement n'est pas le cas en l'espèce.

3 La phase préparatoire

3.1 Récupération de l'information publique sur la cible

L'attaquant va réaliser son attaque de façon distante, il se place donc dans le cas ou il n'a pas accès au réseau interne de l'éditeur. Il va donc utiliser Internet et récupérer le maximum d'informations publiques sur sa cible et son contexte. A l'inverse de la prise d'empreinte, le pirate n'aura aucune interaction avec sa cible.

Pour réaliser cet objectif il a plusieurs moyens à sa disposition, nous nous attacherons à présenter ceux concerant les bases Whois, les bases DNS et les moteurs de recherche.

Les bases Whois Les bases Whois permettent de connaître le propriétaire d'une ou plusieurs adresses IP. On les utilise fréquemment pour connaître la ou les adresses IP d'un site Web. Ces bases sont librement consultables via Internet [3]. Elles contiennent également un certain nombre d'informations sur le propriétaire de l'adresse IP (adresse, personne à contacter en cas de problème...).

Il est à noter que la consultation de ces bases se fait de facon complétement furtive du point de vue de la cible, il n'y a acune interaction avec elle.

Dans notre cas, le pirate interroge la base RIPE via le site `network-tools` afin d'obtenir les adresses IP appartenant à sa cible ainsi que toutes les informations légales la concernant et renseignées dans la base.

Domain registry query for sstic.org :

```
Domain ID:D92668917-LROR
Domain Name:BUNKER.COM
Created On:29-Nov-2002 11:02:42 UTC
Last Updated On:24-Jan-2004 12:30:12 UTC
Expiration Date:29-Nov-2004 11:02:42 UTC
Sponsoring Registrar:R42-LROR
Status:OK

Registrant ID:0-674592-Gendi
Registrant Name:John Smith
Registrant Organization:John Smith
Registrant Street1:3rd street
Registrant City:New York
Registrant Postal Code:3492
Registrant Country:US
Registrant Email:john.smith@bunker.com

Admin ID:BP589-GENDI
Admin Name:John Smith
```

```
Admin Street1:4th street
Admin City:New York
Admin Postal Code:3942
Admin Country:US
Admin Phone:564.345623
Admin Email:john.smith@bunker.com

Tech ID:AR41-GENDI
Tech Name:CONTACT NOT AUTHORITATIVE see http://www.gendi.net/whois
Tech Organization:GENDI Corp
Tech Street1:see also whois.gendi.net
Tech City:New York
Tech Postal Code:3942
Tech Country:US
Tech Phone:1
Tech Email:support@gendi.net

Name Server:DNS.BUNKER.COM
```

DNS Records for bunker.com :

```
query from dns.consumer.net to get an authoritative nameserver
NameServer used for query: dns.bunker.com

Answer records
bunker.com 1 NS dns.bunker.com 28705s

Authority records

Additional records
dns.bunker.com 1 A 207.173.176.130 168337s
```

Network IP address lookup :

```
whois whois.ripe.net 207.173.176.142:

inetnum:      207.173.176.0 - 207.173.176.255
netname:      BUNKER
descr:        BUNKER
descr:        10th street
descr:        3942 NY
country:      US
admin-c:      JL644-RIPE
tech-c:       PT316-RIPE
tech-c:       OH251-RIPE
status:       ASSIGNED PA
mnt-by:       OLEANE-NOC
```

```
changed:      hostmaster@oleane.net 19970617
source:       RIPE

route:        62.160.0.0/16
descr:        OLEANE-970501
origin:       AS3215
holes:        62.160.248.0/24
mnt-by:       OLEANE-NOC
changed:      hostmaster@oleane.net 19970502
changed:      hostmaster@oleane.net 20020202
changed:      hostmaster@oleane.net 20021017
source:       RIPE

role:         OLEANE Hostmaster
address:      France Telecom Transpac
address:      20 rue Thomas Edison
address:      92230 Gennevilliers
phone:        +33 1 41 21 78 00
fax-no:       +33 1 41 21 78 99
e-mail:       hostmaster@oleane.net
admin-c:      CP460-RIPE
tech-c:       CW27
nic-hdl:      OH251-RIPE
notify:       hm-dbm-msgs@ripe.net
mnt-by:       OLEANE-NOC
changed:      hostmaster@oleane.net 20000814
changed:      hostmaster@oleane.net 20011105
source:       RIPE

person:       Jerome Leplatre
address:      10 Av De Norvege
address:      Bp 742
address:      91962 Les Ulis
address:      France
phone:        +33 1 69 18 32 32
fax-no:       +33 1 69 28 54 89
nic-hdl:      JL644-RIPE
mnt-by:       OLEANE-NOC
changed:      hostmaster@oleane.net 19970617
source:       RIPE

person:       Patricia Tejeda
address:      10 Av De Norvege
address:      Bp 742
address:      91962 Les Ulis
```

```
address:      France
phone:        +33 1 69 18 32 32
fax-no:       +33 1 69 28 54 89
nic-hdl:      PT316-RIPE
mnt-by:       OLEANE-NOC
changed:      hostmaster@oleane.net 19970617
source:       RIPE
```

Le pirate obtient donc l'adresse IP de sa victime ainsi que le sous réseau qui lui est attribué. Il récupère également les coordonnées de deux contacts, l'un de nature technique, et l'autre de nature administrative. Il obtient également l'adresse IP du serveur DNS de l'entreprise. Cette donnée est très utile car elle va permettre, nous verrons comment par la suite, de connaître l'ensemble des serveurs présents dans l'entreprise et accessible via Internet (à condition que leur soit associé un nom de domaine).

Les bases DNS Le pirate souhaite recupérer les enregistrements DNS correspondants au domaine de la cible. Cette étape se fait également sans interaction avec la victime.

Pour chaque adresse IP du domaine cible, le pirate va demander au DNS si il existe une entrée dans sa base. Dans la plupart des configurations, seules les adresses des serveurs sont renseignées de cette façon.

Le pirate va ainsi connaître les adresses IP des différents serveurs visibles sur Internet. Le nom associé à ces serveurs peut parfois nous renseigner sur leurs fonctions. Un script pour `bash` automatisant cette tâche pour `bash` se trouve en annexe 1.

```
Nom :    dns.bunker.com
Address:  207.173.176.130
--
Nom :    web.bunker.com
Address:  207.173.176.142
--
Nom :    mail.bunker.com
Address:  207.173.176.202
--
Nom :    ftp.bunker.com
Address:  207.173.176.250
--
Nom :    ssh.bunker.com
Address:  207.173.176.251
```

Il ressort de cette analyse qu'il y a seulement 5 entrées disponibles depuis l'extérieur : les 4 entrées renseignées par le DNS et le serveur DNS lui-même.

Grâce à ces informations, le pirate sait qu'il a potentiellement accés à 5 serveurs appartenant à l'entreprise cible.

Les moteurs de recherche L'utilisation d'un ou plusieurs moteurs de recherche va permetttre d'obtenir beaucoup d'informations sur la cible (informations sur les employés, la structure du réseau, les logiciels utilisés en interne...). Ces données sont rarement exploitables telles quelles, mais elles permettent de se faire une idée sur l'entreprise en général.

Par exemple, il arrive qu'un utilisateur pose ou réponde à des questions dans des forums de discussion. Ce type de comportements permet de renseigner le pirate sur l'environnement qu'il doit attaquer (dans le cas ou cet utilisateur est un administrateur du système cible), d'avoir une meilleure connaissance des compétences des employés, de leurs centres d'intérêt (très pratique au cas où il a besoin de faire de l'ingénérie sociale).

Le moteur de recherche `Google` [6] est un outil très précieux pour obtenir ce type d'informations car il permet, entre autres, de faire une recherche dans les groupes de discussion.

Le pirate reussit à identifier, via des annonces faites lors de salons informatiques, le responsable de la conception du jeu vidéo. Par ailleurs, il reussi, via des forums et l'option `Original Format` de Google, à recupérer un courier électronique et son entête qui lui permet de savoir quel est le serveur mail utilisé par l'entreprise cible.

```
Reply-To: "Bill John" <bill@bunker.com>
From: "Bill John" <bill@bunker.comd>
Newsgroups: comp.dev.microsoft.directx
Subject: Re: directx optimisation problems
Date: Sun, 5 Oct 2003 21:36:28 +0200
Organization: Bunker
MIME-Version: 1.0
Content-Type: text/plain;
charset="iso-8859-1"
Content-Transfer-Encoding: 8bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Newsreader: Microsoft Outlook Express 6.00.2800.1158
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1165
Lines: 124
Message-ID: <3f8074dc$0$27598$626a54ce@news.bunker.fr>
NNTP-Posting-Date: 05 Oct 2003 21:45:32 MEST
NNTP-Posting-Host: 82.65.102.16
X-Trace: 1065383132 news2.bunker.com 27598 207.173.176.202:2504

You could resolve your problems by searching in the MSDN Library.
```

```
Enter your query in the index and follow the answer!

good luck and keep us in touch!

Bill

--
Chief Develloper/Demi-vie
Bunker Software
```

3.2 Cartographie et prise d'empreinte

Les méthodes précédentes n'engendraient aucune interaction avec le système cible, ce n'est plus le cas avec les méthodes de cartographie et de prise d'empreinte que le pirate va maintenant utiliser.

Ces méthodes offrent une granularité de résultats plus importante, mais en contrepartie elles sont beaucoup plus bruyantes en terme d'interactions avec le système cible. Si ce dernier possède un détecteur d'intrusion ou si les fichiers de journalisation sont correctement configurés et analysés, il sera capable de détecter ces interactions.

Le routage La connaissance du chemin emprunté par les paquets réseau permet souvent d'identifier la présence d'un pare-feu. Pour cela, le pirate va utiliser un outil très pratique et présent sur toutes les plate-forme : `traceroute`. Son fonctionnement est relativement simple, il va nous donnner des informations le chemin parcouru par nos paquets avant d'arriver sur la cible.

```
TraceRoute to www.bunker.com (207.173.176.142), 40 byte packets
 1  thing-i.sdsc.edu (198.202.76.40)  1.019 ms
 2  medusa.sdsc.edu (198.202.75.10)  0.759 ms
 3  piranha.sdsc.edu (132.249.30.8)  7.170 ms
 4  sdg-dc1--sdsc-sdsc2-ge.cenic.net (137.164.24.157)  6.171 ms
 5  lax-dc1-sdg-dc1-pos.cenic.net (137.164.22.46)  5.183 ms
 6  dc-sac-dc1--lax-dc1-pos.cenic.net (137.164.22.127)  12.831 ms
 7  dc-oak-dc2--csac-dc1-ge.cenic.net (137.164.22.110)  15.942 ms
 8  dc-oak-dc1--oak-dc2-ge.cenic.net (137.164.22.36)  15.870 ms
 9  dc-paix-px1--oak-dc1-ge.cenic.net (137.164.40.11)  16.017 ms
10  198.32.175.27 (198.32.175.27)  18.212 ms
11  p7-0.cr01.sntd.eli.net (207.173.114.137)  25.384 ms
12  p9-0.cr02.rcrd.eli.net (207.173.114.58)  23.388 ms
13  srp3-0.cr01.rcrd.eli.net (208.186.20.241)  25.533 ms
14  p9-0.cr02.ptld.eli.net (207.173.115.41)  34.162 ms
15  srp3-0.cr01.eli.net (208.186.21.3)  43.713 ms
16  srp0-0-0.gw01.eli.net (208.186.20.38)  43.211 ms
```

```
17  gw0-bunker-COM.eli.net (209.63.173.46)  47.575 ms
18  * * *
19  * * *
```

Cette méthode est sans interactions avec le serveur cible lorsque nous utilisons un `traceroute` disponible sur Internet. Cependant, cela ne permet pas toujours d'avoir la granularité que nous pourrions obtenir en interagissant directement avec les machines cibles.

En effet, le pirate constate qu'il n'arrive pas à atteindre les serveurs cibles (l'opération a été réalisé pour les 4 serveurs figurant dans les entrées DNS, seule celle concernant le serveur Web est présentée ci-dessus).

Ses paquets semblent en effect bloqués par un pare-feu. Etant donné qu'il souhaite connaître précisément l'architecture réseau de sa cible, notamment la présence d'éléments entre le pare-feu et les serveurs, il va être obligé de réaliser certaines actions en interrogant directement les serveurs. Cette méthode laisse des traces, mais c'est la seule qui permettent de faire les tests désirés.

Pour cela, il va utiliser l'utilitaire `Hping2` [5] qui va nous permettre de réaliser un `traceroute` en utilisant le protocole TCP. Le pirate va changer la valeur du champ IP Time To Live (`TTL`) et ainsi joindre exactement la machine souhaitée.

```
[root:/home/lefranc/hping2]# ./hping2 -S -p80 207.173.176.142 -t 19
HPING 207.173.176.142 (eth0 207.173.176.142): S set, 40 headers + 0
data bytes
len=46 ip=207.173.176.142 ttl=53 id=0 sport=80 flags=SA seq=0
win=5840 rtt=36.9ms
```

Le pirate arrive à atteindre le serveur sans problème puiqsu'il répond au paquet que nous lui avons envoyé. Ce serveur se trouve donc bien au hop 19.

```
[root:/home/lefranc/hping2]# ./hping2 -S -p80 207.173.176.142 -t 17
HPING 207.173.176.142 (eth0 207.173.176.142): S set, 40 headers + 0
data bytes
TTL 0 during transit from ip=209.63.173.46 /
name=gw0-bunker-COM.eli.net
```

La valeur donnée par le pirate au TTL ne permet d'atteindre que le routeur Internet. C'est lui qui répond à la requète en indiquant que le TTL a expiré.

```
[root:/home/lefranc/hping2]# ./hping2 -S -p80 207.173.176.142 -t 18
HPING 207.173.176.142 (eth0 207.173.176.142): S set, 40 headers + 0
data bytes
TTL 0 during transit from ip=207.173.176.10 name=UNKNOWN
```

La valeur donnée au TTL par le pirate permet d'atteindre une machine située entre le routeur Internet et le serveur. Selon toute vraissemblance, c'est un pare-feu que le pirate vient d'identifier à l'adresse 207.173.176.10.

Identification système : le balayage de ports Il est possible de déterminer les systèmes qui sont actifs et accessibles à partir d'Internet en utilisant un certain nombre de grandeurs fournies dans les divers champs des protocoles de communications.

En effet, bien que les RFC fournissent un cadre de mise en oeuvre et permettent ainsi une standardisation des protocoles d'échanges. La plupart des implémentations faites par les fabricants de systèmes d'exploitation sont conformes à ces RFC, mais en ont une interprétation différente, et permettent ainsi aux pirates de les identifier.

Dans notre cas, le pirate va utiliser l'outil `Nmap` [4] pour réaliser le balayage de port des différentes machines précédemment identifiées. Cela va lui permettre de vérifier que les services sont bien operationnels et d'être renseigné sur la nature du système d'exploitation utilisé.

Cette technique de prise d'empreinte est relativement fiable, mais elle génère une interaction avec la cible qui peut faire l'objet d'une détection de sa part. C'est pour cette raison que nous allons utiliser le mode furtif (`Stealth Scan`) de `Nmap` afin de ne pas initier complétement la connexion, et ainsi, d'être le plus discret possible.

```
[root:/home/lefranc]# ./nmap -sT -sV -vv -O -P0  207.173.176.142

Starting nmap 3.51-TEST2 ( http://www.insecure.org/nmap/ )
at 2004-04-13 21:37 CEST
Host web.bunker.com (207.173.176.142) appears to be
up ... good.
Initiating Connect() Scan against web.bunker.com
(207.173.176.142) at 21:37
Adding open port 80/tcp
The Connect() Scan took 23 seconds to scan 1660 ports.
Initiating service scan against 1 service on 1 host at 21:38
The service scan took 6 seconds to scan 1 services on 1 host.
Interesting ports on web.bunker.com (207.173.176.142):
(The 1649 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE       VERSION
21/tcp   filtered ftp
22/tcp   filtered ssh
23/tcp   filtered telnet
80/tcp   open     http          Microsoft IIS webserver 5.0
135/tcp  filtered msrpc
136/tcp  filtered profile
137/tcp  filtered netbios-ns
138/tcp  filtered netbios-dgm
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
```

```
3389/tcp filtered microsoft-rdp Microsoft Terminal Service
(Windows 2000 Server)
Device type: general purpose
Running: IBM AIX 4.X
OS details: IBM AIX 4.3.2.0-4.3.3.0 on an IBM RS/*
OS Fingerprint:
TSeq(Class=TR%TS=0)
T1(Resp=Y%DF=N%W=FFFF%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=N)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=N)
PU(Resp=N)

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: ED8BE3AF 82230B4C D8C1E49E 368DB819
      B5C869F8 14227B78
IPID Sequence Generation: Busy server or unknown class

Nmap run completed -- 1 IP address (1 host up) scanned in
39.508 seconds
```

`Nmap` n'a pas été capable d'identifier le système d'exploitation de façon correctte, le manque de données en est probablement la cause. En effet, plus la machine cible "répond" aux requétes du pirate, plus `Nmap` va être capable de correctement l'identifier.

Dans notre cas, la machine cible n'a pas été suffisamment loquace! (c'est peut-être le signe d'un pare-feu correctement configuré...). A l'inverse, il a bien reconnu la version du serveur web installé sur la machine `web.bunker.com`.

Etant donné la nature de ses applications, le pirate en conclu que le serveur Web fonctionne sous le système d'exploitation Windows. Cela lui est confirmé par le site Web `Netcraft` [7] qui répertorie, entre autre, les évolutions de configuration des serveurs Web qui sont dans sa base de données. Ce site identifie le serveur Web du système cible comme étant Windows 2000. C'est cohérent par rapport à l'information que le pirate a déjà recueilli.

Identification logiciel : les entêtes Chaque service possède une entête (ou bannière) qui permet de l'identifier. La connaissance de ces entêtes associée aux résultats fournis par `Nmap` permettent de connaître la machine cible de façon très fine.

L'attaquant va donc récupérer les entêtes des différents services précédemment identifiés.

```
[root:/home/lefranc]# telnet 207.173.176.142 80
Trying 207.173.176.142...
Connected to web.bunker.com (207.173.176.142).
Escape character is '^]'.
GET HTTPS
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Tue, 13 Apr 2004 17:40:30 GMT
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title></head><body>The parameter
is incorrect.
</body></html>Connection closed by foreign host.
```

le pirate recupère la bannière du serveur Web de l'entreprise. Cela confirme ce qu'il avait déjà conclu à l'étape précédente, le serveur Web est basé sur une architecture à base de produits Microsoft.

Toutes les informations que le pirate vient de récupérer, à la fois de façon passive, mais aussi en agissant sur le système, vont lui fournir la connaissance nécessaire à la réalisation de son attaque.

3.3 L'arbre d'attaque

Lors des phases précédentes, le pirate a récolté autant d'informations que possible sur sa cible, à la fois de façon passive, mais aussi en interagissant avec elle. Il a une bonne idée de son architecture matérielle et logicielle (figure 1).

Le but du pirate est de s'introduire sur le réseau de sa cible afin de dérober les sources du futur jeu. Le fait d'identifier un ou plusieurs serveurs vulnérables ne lui est pas d'une grande utilité, même si par "rebond" il lui serait possible d'atteindre son objectif. Le plus simple, compte tenu de ses impératifs de réussite, serait de prendre la main sur la machine d'un développeur du jeu et ensuite d'accéder au serveur ou sont stockées les sources du jeu.

Les informations qu'il a recupérés via `Google` lui indiquent deux choses :

- les employés ont accés à Internet depuis leur poste travail car il existe des traces de leurs discussions dans les forums,
- le serveur de mail est le logiciel `Exchange` de Microsoft, il est donc fort probable que les postes utilisateurs fonctionnent avec le système d'exploitation Windows et utilise le navigateur Internet Explorer.

Ce dernier point est renforcé par le fait que Windows est également la plateforme sur laquelle s'exécutera le futur jeu, il y a donc de fortes chances que ce soit également le type d'OS utilisé par les développeurs.

La mise en corrélation de toutes les informations que le pirate a obtenues lui permet de construire l'arbre d'attaque nécessaire à la réalisation de son objectif.

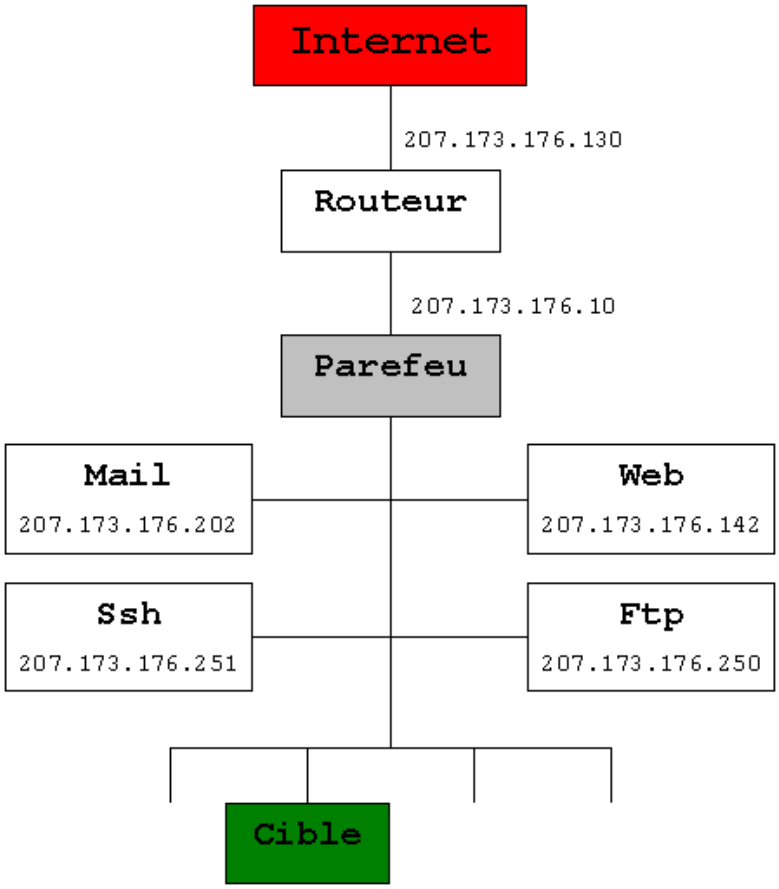


Fig. 1. Architecture de l'entreprise cible

Cette attaque va consister à faire télécharger et exécuter un cheval de Troie par le chef de l'équipe de dévelopement.

Pour cela, il utilise une faille dans Internet Explorer [8] qui permet de faire télecharger un programme et de le faire exécuter lorsque qu'un utilisateur visite une page web, et ce, sans que cette personne en ait conscience. Le pirate va donc forger un mail à l'attention du chef d'équipe, ce mail contient un lien vers la page piégée. L'astuce, pour le pirate, va être de faire en sorte que sa victime aille sur la page incriminée.

Une fois installé, le cheval de Troie va essayer de contacter une machine cliente appartenant au pirate. Ce cheval de Troie n'est rien d'autre que le programme `netcat`. Etant donné que l'entreprise est dotée d'au moins un pare-feu fonctionnant via la méthode `statefull inspection`, le serveur `netcat`ne sera pas accessible de l'extérieur du réseau interne. Pour remédier à ce problème, ce sera le serveur qui initiera la connexion (fonctionnement en mode `reverse connection` vers le client.

Le cheval de Troie permet au pirate d'avoir accés à la machine victime et, notamment, de télécharger et d'installer tout ce qui lui est nécessaire pour atteindre son objectif. Ces composants supplémentaires sont au nombre de trois :

- L'outil Hacker Defender [9] permet de dissimuler une ou plusieurs applications au sein du système d'exploitation. Une fois ce logiciel installé l'utilisateur peut cacher des fichiers, des processus, des services, des drivers système, des clefs de registre, des ports ouverts et enfin modifier l'espace disque disponible.
- un keylogger [10] pour recupérer toutes les touches frappées par l'utilisateur, et notamment les mots de passe de connexions aux autres machines.
- un sniffeur [11] pour analyser les flux auxquels la machine victime a accès, afin de consolider les informations obtenues par les précédents outils..

4 L'opération "Carbone 14"

4.1 La tentative consommée

Le pirate met en pratique l'arbre d'attaque qu'il a élaboré. Il crée une page web sur chez un hébergeur en ligne, en ayant au préalable tout réalisé depuis un "café Internet" afin de minimiser ses traces. Il élabore le contenu du faux site en fonction des centres d'intérêts de la personne qui va aller le visiter.

Il forge un mail dans lequel il se fait passer pour une connaissance de la victime et l'incite à se rendre sur la page en question. Dès que la victime visionne la page, le cheval de Troie s'installe et essaie de se connecter à son client.

4.2 Le vol de sources

Le pirate rapatrie via le réseau les données contenant le code source du jeu "demi-vie". Il est aidé dans cette tâche par le cheval de Troie. En effet, le chef

de projet a, comme l'ensemble des développeurs, un accés direct sur le serveur contenant les sources.

Le cheval de Troie a permis l'installation d'un keylogger qui va pouvoir enregistrer toutes les touches frappées par la victime. Il va notamment permettre de recupérer le mot de passe permettant l'accés au serveur des sources. Lorsque cette opération est réalisée, le pirate prend le contrôle à distance de la machine du chef de projet et se connecte sur un serveur `FTP` afin d'y déposer l'ensemble des fichiers présents sur le serveur.

4.3 La réalisation du préjudice : la dissémination

Le pirate compile les sources et les diffuse ensuite via un service de pair à pair. Rapidement le jeu se dissémine sur le réseau. La presse spécialisée s'empare de cette information et rapidement le tout le microcosme du monde du jeu vidéo est au courant. Les éditeurs cocurrents s'empressent de télécharger le jeu afin d'analyser son potentiel et, eventuellement, de le comparer avec leurs propres productions.

Il y a également une phase 3 bis : c'est celle qui permet au pirate de toucher sa rétribution. Celle ci n'est pas à négliger sur le terrain de la preuve, car tout flux d'argent laisse des traces : au moins un débit au départ et un crédit à l'arrivée.

5 L'enquête

5.1 L'enquête technique post-intrusion réalisée par la victime

L'utilisateur légitime de la machine piratée va remarquer une instabilité de son système. Il décide de faire appel aux administrateurs système afin d'élucider les causes de ces disfonctionnements.

6 Conclusion

La morale de l'histoire : si vous ne voulez pas que le cadavre de votre entreprise doit daté au "Carbone 14", n'attendez pas une "demi-vie" pour protéger simplement votre patrimoine : un tandem juridique et technique sera plus rentable que des logiciels de sécurité seuls ou des contrats d'assurance seuls.

7 Remerciements

Nous souhaitons remercier toutes celles et ceux qui ont pu contribuer à la réalisation de cet article.

Références

1. Valve Software, http://www.halflife2.net/forums/showthread.php?threadid=10692
2. MISC Magazine, http://www.miscmag.com
3. Network Tools, http://www.network-tools.com
4. Nmap, http://www.insecure.org
5. Hping2, http://www.hping.org
6. Google, http://www.google.com
7. Netcraft, http://www.netcraft.com
8. Internet Explorer Unspecified CHM File Arbitrary Code Execution , http://www.k-otik.net/bugtraq/02.18.InternetExplorer.php
9. Hacker Defender, http://rootkit.host.sk/
10. Klogger, http://www.ntsecurity.nu/toolbox/klogger/
11. TCPdump, http://www.tcpdump.org

8 Annexe 1

```
#!/bin/bash

if [ -z "$1" ]; then
    echo
    echo "USAGE : ./lookup addr_ip_range bit_inf bit_sup string_match"
    echo "   ex : ./lookup 10.11.12 1 255 coco"
    echo
    exit
fi

base_ip="$1"
inf="$2"
sup="$3"
string="$4"

for i in ‘seq $inf $sup‘;
do
nslookup $base_ip.$i >> nslookup_res.txt
done

grep -A 1 -i $string nslookup_res.txt > addr_ip_res.txt

exit 0
```

(In)sécurité de la Voix sur IP (VoIP)

Nicolas FISCHBACH

COLT Telecom/Sécurité.Org
nico@{colt.net,securite.org}

1 Introduction

Jusqu'à récemment, la voix sur IP était plutôt la technologie d'une minorité de "férus du net" car elle permet de téléphoner à moindre coût via l'Internet, la voix restant une forme de communication bien plus conviviale que le courrier électronique ou les formes de messageries instantanées. Avec l'évolution de l'Internet, le déploiement de réseaux privés virtuels, l'introduction de la qualité de service dans les réseaux, l'arrivée des PBX IP, la disponibilité de postes téléphoniques intégrant des fonctionnalités de plus en plus avancées ainsi que l'opportunité de réduction des coûts, l'intérêt que suscite la voix sur IP pour l'entreprise et pour les utilisateurs est grandissant jusqu'à devenir un projet stratégique. Tout d'abord les différents protocoles ainsi que leurs évolutions récentes seront présentés, puis les éléments qui composent une solution VoIP ainsi que les différentes formes d'attaques et les moyens de s'en prémunir ou de réduire les risques, et enfin pour terminer, une comparaison de la sécurité de la voix sur IP par rapport au réseau téléphonique classique et au réseau GSM.

2 Les différents protocoles

La voix sur IP est une description relativement générique et ne définit pas une liste exclusive de protocoles, ni d'équipements. Avant d'étudier les aspects sécurité nous allons présenter les différents protocoles, dont certains sont en train de devenir obsolètes, et d'autres émergents.

2.1 Le protocole H.323

H.323 est le premier protocole développé pour permettre des communications multi-médias. SIP son " concurrent ". H.323 est relativement complexe et SIP tente de simplifier les échanges en utilisant une sémantique proche de HTTP.

H.235 définit des mécanismes de sécurité.

2.2 SIP

SIP (*Session Initiation Protocol*) est le standard IETF pour la signalisation (établissement, terminaison, redirection, relayage, etc) de communications multimédias interactives. Ce protocole est de nos jours celui qui est déployé couramment. Le format est proche d'une adresse de messagerie : *sip :nico@securite.org*

SIMPLE (*SIP for Instant Messaging and Presence Leveraging Extensions*) est une extension de SIP dont le but est de supporter les messageries instantanées.

Le projet PROTOS s'est intéressé à SIP, et comme pour SNMP, a trouvé un nombre important d'implémentations qui ne passaient pas le " batch " de tests sans se planter ou redémarrer. Cela concerne aussi bien les téléphones, que les relais SIP et les équipements de sécurité.

2.3 RTP

RTP (*Real Time Transport Protocol*) encode, transporte et décode la voix.

2.4 MEGACO/H.248

MEGACO (*MEdia GAteway COntrol Protocol*) est un protocole commun à l'IETF et l'ITU-T qui joue le rôle d'interface entre des passerelles (par exemple entre la passerelle qui gère la signalisation et celle qui gère le flux voix). C'est une évolution de MGCP (*Media Gateway Control Protocol*). A la différence de H.323 et de SIP qui reposent sur une architecture distribuée, MEGAGO et MGCP sont orientés client/serveur (ie. centralisé).

3 Les protocoles secondaires

3.1 DNS

Le service DNS est utilisé pour fournir des services d'annuaire et de localisation.

3.2 TFTP et HTTP

Ces deux protocoles sont utilisés par les téléphones et différents autres éléments pour télécharger leur configuration.

3.3 ENUM

ENUM permet de lier des adresses SIP via DNS aux numéros de téléphone au format E.164

4 Architecture d'une solution VoIP

Après cette liste de protocoles et d'acronymes, qui bien que longue est loin d'être exhaustive, nous présentons les différents éléments qui composent l'architecture d'une solution VoIP.

4.1 Le réseau

Le réseau interconnecte les différents équipements qui participent à une solution de voix sur IP. A la différence du réseau téléphonique, la signalisation et la voix sont transportés sur le même réseau partagé. Comme pour IPsec, des solutions alternatives ont dû être développées pour gérer les contraintes introduites par la traduction d'adresses (NAT).

4.2 Le téléphone

Le téléphone peut se présenter sous deux formes : soit un téléphone " classique ", soit un téléphone " logiciel " qui s'éxécute sur l'ordinateur de l'utilisateur. En terminologie SIP c'est un UA (*User Agent*).

4.3 Les systèmes

La liste des protocoles usités dans les solutions de téléphonie et de voix sur IP est conséquente. Il en va de même de celle des systèmes.

4.4 Serveurs SIP : proxy, redirect et registrar

Le " *SIP proxy* " joue le rôle de relais et fait suivre une requête SIP au prochain serveur. Le " *SIP redirect server* " renvoit une réponse au client contenant le prochain serveur à contacter. Le " *SIP registrar* " enregistre le lien entre l'adresse IP et l'adresse SIP.

4.5 Le " Call Manager " / IP PBX

Le CM fournit des fonctions de base de gestion d'appel, des utilisateurs et des configurations, et également des fonctionnalités avancées comme la conférence, les boîtes vocales, etc. Il peut être vu comme un IP PBX. A ne pas confondre avec des PBX traditionnels (pas de VoIP) que l'on peut administrer à distance via une connexion TCP/IP (qui remplace la connexion locale ou de télé- maintenance via un modem).

4.6 Les passerelles

Une passerelle s'occupe des échanges entre le réseau voix sur IP et le réseau téléphonique classique. Cela comprend la conversion de la voix et de la signalisation.

5 Les équipements de sécurité

5.1 Pare-feu

Bon nombre de pare-feux se limitent à gérer l'ouverture de ports en fonction des communications et n'inspectent pas les flux (au niveau protocolaire). De plus cet élément additionnel risque d'introduire un délai ainsi qu'une gigue, c'est pourquoi ils sont absents dans bien des déploiements.

5.2 IDS

Il n'est pas très courant de trouver des outils de détection d'intrusion pour des solutions de voix sur IP. La quantité de faux positifs dus à l'observation du flux RTP pourrait être plus que conséquente. Bien qu'elle permette de détecter des dénis de service par exemple, la détection d'intrusion se ramène souvent à de la détection de fraude.

6 Les attaques et les parades

6.1 Les types d'attaques

Déni de service Les attaques par déni de service se retrouvent sous plusieurs formes. Les plus classiques sont celles qui visent à utiliser toute la bande passante disponible ou abuser de problèmes intrinsèques à TCP/IP.

Dans le cadre d'une solution VoIP bien des éléments peuvent être attaqués : le téléphone, le réseau, le système d'exploitation, l'application, etc. Autant un déni de service sur l'Internet peut être filtré avec des mécanismes et des techniques plus ou moins avancées, autant celui à l'encontre d'une communication sera difficile à traiter et aura un impact direct sur les possibilités de communications.

Par exemple un nombre trop important de messages SIP INVITE ou de simples messages ICMP peuvent créeer une situation de déni de service.

Interception L'interception d'une communication peut être l'oeuvre d'un " criminel informatique " ou des autorités. En effet, l'interception légale de communications via un réseau de données (LI – *Lafwul Intercept on packet switched networks*, standard ETSI pour l'Europe et CALEA - *Communications Assistance for Law Enforcement Act* par exemple) est un sujet de discussion dans bon nombre de pays. Reste à voir comment sera traitée la voix sur IP par rapport aux communications téléphoniques classiques où l'opérateur est quasiment obligé d'être impliqué.

Les techniques bien connues d'écoute de réseau (" sniffing " ou de l'homme du milieu (" man in the middle " s'appliquent à l'interception. En revanche, contrairement à une attaque MITM contre un protocole comme telnet ou HTTP où il est possible de modifier le contenu à la volée, cela s'avère plus contraignant et plus facilement détectable avec de la voix encodée.

Le protocole RTP transporte la voix encodée, sans aucun chiffrement, ce qui rend l'interception relativement triviale.

" Call-ID " Le service de présentation du numéro de l'appelant est devenu, pour bon nombre d'abonnés le facteur principal de prise ou de rejet d'appel, tout particulièrement depuis l'essor des téléphones portables.

A la différence du téléphone classique où le numéro est lié à la ligne physique et qu'un voisin indélicat ne peut prendre ou écouter vos appels qu'à condition de mettre des pinces crocodiles sur votre ligne, ou écouter la bonne fréquence si

vous utilisez un téléphone sans-fil, cela est différent dans une solution VoIP. En effet, la mobilité qu'apporte la téléphonie sur IP introduit également un problème d'authentification de l'utilisateur : celui-ci doit se souvenir de son nom/numéro d'utilisateur et de son mot de passe, que quelqu'un pourrait lui " voler ", et donc s'enregistrer à sa place ou de manière concurrente.

Il est également relativement simple de manipuler l'identifiant de l'appelant. L'impact n'est pas très important, sauf dans le cas ou le CLID est utilisé pour authentifier l'appelant et autoriser l'accès à une ressource (boîte vocale, appels internationaux ou numéros spéciaux, etc...).

Non-répudiation et fraude La fraude la plus connue est l'accès gratuit ou à un côut réduit à des services à valeur ajoutée ou des appels internationaux.

La compromission de serveurs Les serveurs jouent un rôle important dans une solution de voix sur IP, et même s'il n'est pas forcément possible d'intercepter un appel si un serveur est compromis, il est souvent possible de récupérer des CDRs (*Call Detail Records*) qui contiennent toutes les traces des appels effectués. En revanche la compromission d'une passerelle entre le réseau VoIP et le réseau téléphonique classique permet d'écouter de manière transparente les appels, même s'ils sont chiffrés du côté VoIP (SRTP).

6.2 L'architecture sécurisée

Les systèmes et le réseau En fonction des options de déploiement choisies, il est probable que la majorité des systèmes doivent être accessibles depuis " partout ". Il convient donc de sécuriser ces éléments comme tout serveur, et dans la mesure du possible de mettre en place une solution avec des pare-feux.

La séparation entre le trafic voix et la signalisation peut se faire au niveau du réseau à l'aide de VLANs.

Déni de service Les différents éléments qui composent une architecture de type SIP, comme le relais, sont livrés par certains vendeurs avec des mécanismes de détection de déni de service (comme l'envoi massif de messages INVITE).

Le mécanisme le plus important est la qualité de service (QoS) Sans déploiement de bout-en-bout, et tout particulièrement sur les réseaux locaux où ce n'est pas très commun, il est possible pour n'importe qui de générer un déni de service. La bande passante disponible est également un facteur, mais la QoS et la gestion des files d'attentes est le point clé.

La voix sur IP n'implique pas que le trafic utilise l'Internet comme média, un déni de service sur un réseau privé est plus difficile à mettre en oeuvre.

Des solutions de filtrage de contenu ou d'analyse anti-virus commencent à être disponibles et utilisent SIP pour permettre l'inspection, un peu à l'image de protocoles comme OPSEC.

Interception La seule solution pour limiter l'interception est l'usage de mécanismes cryptographiques pour les flux de signalisation et de données (voix encodée). Ce chiffrement devrait être de bout en bout et il est important d'en évaluer les impacts. Le plus important étant celui sur la qualité de la communication à cause du délai supplémentaire introduit. Comme dans toute solution cryptographique, l'authentification forte des parties impliquées dans les échanges est un élément, sinon les attaques classiques de MITM peuvent s'appliquer.

Comme dans toute solution où les données sont chiffrées, elles doivent être déchiffrées. Cela nous ramène à la problématique du poste client. Autant il était impossible de mettre un cheval de Troie ou une porte dérobée sur le téléphones de nos grands-parents, autant il est aujourd'hui possible de " patcher " quasiment n'importe quel téléphone... sans parler des téléphones logiciels (" soft phones ").

La version " S " de SIP, SIPS, reposant sur TLS (*Transport Layer Security*) sur TCP permet de chiffrer les échanges SIP qui contiennent, par exemple, le nom d'utilisateur, le mot de passe ainsi que le numéro appelé.

Une nouvelle version du protocole RTP, S-RTP intègre des mécanismes de chiffrement.

Non-répudiation et fraude La fraude peut être limitée par la configuration correcte de la passerelle VoIP vers RTC (Réseau Téléphonique Commuté) ainsi qu'une gestion des droits/classes de service par utilisateur. Les passerelles doivent être configurées pour éviter qu'un utilisateur se connecte directement sans passer par le relais SIP.

7 Les réseaux de téléphonie " classiques "

A titre de comparaison, nous allons discuter, sans rentrer dans les détails, de la sécurité des réseaux téléphoniques plus classiques.

7.1 PSTN/POTS

Le réseau téléphonique filiaire (PSTN/POTS – *Public Switched Telephone Network/Plain Old Telephone System*) transporte voix et données. A la différence d'un réseau VoIP où les équipements qui " gèrent " les communications sont souvent dans le même réseau et accessibles, cela n'est pas le cas dans le RTC (SS7 et le " switch " par exemple).

7.2 GSM

Le réseau cellulaire (GSM – *Global System for Mobile Communications*) à la différence d'un réseau sans fil de type 802.11 ne permet pas d'écouter facilement une communication. En revanche les communications ne sont chiffrées qu'entre le téléphone de l'utilisateur et la station à laquelle il est rattaché. Ce n'est pas un chiffrement de bout-en-bout entre l'appelant et l'appelé, mais des solutions

existe qui emploient par exemple le mode données pour transporter de la voix chiffrée.

Un changement majeur est apparu ces dernières années avec le déploiement des réseaux de troisième génération (GPRS et UTMS) : les téléphones ne sont plus des grille-pains mais sont livrés avec un système d'exploitation, Java, une pile TCP/IP, etc. Et de plus sont connectés en permanence au réseau et disposent d'une adresse IP.

8 Conclusion

Du fait que tous les équipements " parlent " IP et que bien souvent l'Internet soit le média de transport, et également par manque d'éducation des utilisateurs il est clair le nombre d'attaques à l'encontre de solutions voix sur IP est plus large que celle ä l'encontre du RTC ou GSM. Peut-être également parce que bon nombre d'attaques sur l'Internet sont relativement anonymes et qu'il n'est pas facile de les tracer.

A propos

Nicolas FISCHBACH est Senior Manager chez COLT Telecom et dirige l'équipe sécurité au sein du département européen d'ingénierie IP. Il est également co-fondateur de Sécurité.Org, un site web francophone dédié à la sécurité informatique, d'eXperts, un groupe informel de spécialistes sécurité ainsi que du chapitre français du Honeynet Project. Nicolas participe à de nombreuses conférences (BlackHat Briefings, CanSecWest, Defcon, JSSI, Eurosec, NANOG, Cisco Systems, RIPE, SwiNOG, Libre Software Meeting, etc), publie des articles (MISC) et donne des cours dans différentes écoles et universités (HEC, Université de Genève, ITIN, etc). Pour plus d'informations : `http://www.securite.org/nico/`

Les compromis temps-mémoire et leur utilisation pour casser les mots de passe Windows

Philippe Oechslin

Laboratoire de Securité et de Cryptographie (LASEC)
École Polytechnique Fédérale de Lausanne
Faculté I&C, 1015 Lausanne, Switzerland
philippe.oechslin@epfl.ch

Résumé Les compromis temps-mémoire sont des méthodes qui permettent de réduire le temps d'exécution d'un algorithme en augmentant la quantité de mémoire utilisée. Dans cet article nous présentons différentes variantes de compromis temps-mémoire qui permettent d'accélérer le cassage de mots de passe. Nous expliquons comment configurer un tel compromis pour obtenir le cassage le plus rapide, comment estimer les performances que l'on peut espérer atteindre et nous montrons comment implémenter cette méthode de manière efficace pour casser les mots de passe des systèmes Windows.

1 Introduction

La base de ce travail est une méthode originale publiée en 1980 par Martin Hellman [1] et améliorée par Whitfield Diffie en 1982. Dernièrement nous avons mis au point une variante de cette méthode qui augmente son efficacité d'un ordre de magnitude. Ces méthodes cherchent toutes à résoudre le même problème : Comment inverser rapidement une fonction irréversible. Il peut s'agir par exemple de retrouver la clef qui a servi à chiffrer un texte prédéfini [1] ou, ce qui est notre cas, de retrouver un mot de passe à partir de son empreinte. Dans les deux cas, on peut utiliser une méthode de force brute, qui consiste à essayer toutes les possibilités jusqu'à ce qu'on trouve le bon mot de passe. Une autre approche consisterait à calculer une fois les empreintes de touts les mots de passe possibles et de les stocker, avec les mots de passe, dans un énorme dictionnaire. La force brute est une méthode qui nécessite énormément de temps mais aucune mémoire, alors qu'un dictionnaire complet permet de trouver le mot de passe immédiatement mais à l'aide d'une mémoire énorme. Le but des compromis temps-mémoire est de créer des solutions intermédiaires qui permettent de trouver un mot de passe plus vite qu'avec la force brute en utilisant moins de mémoire qu'un dictionnaire complet.

Dans la suite nous allons d'abord voir le principe général de fonctionnement des compromis temps-mémoire qui nous intéressent. Ensuite nous allons voir de manière plus formelle comment trouver les paramètres optimaux des compromis

[1] *fixed plaintext attack.*

et quelles sont les performances que l'on peut attendre de ces méthodes, en particulier dans le cas des mots de passe Windows. Nous verrons finalement comment dans ce cas précis l'implémentation de la méthode peut être affinée pour obtenir des résultats encore plus performants.

2 Compromis à base de chaînes

Les compromis temps-mémoire sont réalisés à l'aide de chaînes. Pour pouvoir créer des chaînes on définit une fonction de réduction, qui génère un mot de passe arbitraire à partir d'une empreinte. En alternant la fonction de hashage, qui génère une empreinte à partir d'un mot de passe, et la fonction de réduction qui génère un mot de passe à partir d'une empreinte on peut donc générer des chaînes dans lesquels s'alternent mots de passes et empreintes. Pour obtenir un compromis temps-mémoire on génère un nombre m de chaînes de longueur t et on stocke le premier et le dernier élément de chaque chaîne dans une table.

Pour retrouver un mot de passe il faut d'abord retrouver à quelle chaîne appartient une empreinte. Pour ce faire on génère une chaîne à partir de l'empreinte donnée jusqu'à ce qu'on trouve un mot de passe qui est égal à une fin de chaîne stockée dans la table. A ce moment on sait que l'empreinte appartient à la chaîne dont le début et la fin ont été stockés – ou à une chaîne qui a une fin identique. En regénérant la chaîne à partir du début qui a aussi été stocké dans la table on retrouve le mot de passe que l'on recherche ou on constate qu'il s'agit d'une *fausse alarme* due au fait que deux chaînes ont la même fin. En effet il existe une chance que des chaînes *fusionnent*, ce qui se produit quand la fonction de réduction génère le même mot de passe à partir de deux empreintes différentes. Ceci est inévitable quand l'ensemble des empreintes possibles est plus grand que l'ensemble des mots de passe considérés.

Le procédé est illustré à la figure 1. A partir de l'empreinte "h8" on génère une chaîne jusqu'à ce que tombe sur le mot de passe "1" qui est une fin de chaîne qui nous avons stocké. On regénère la chaîne à l'aide du début de chaîne qui a aussi été stocké pour trouver le mot de passe "8".

2.1 Tables multiples et arcs-en-ciel

Nous avons vu que dans certains cas deux chaînes peuvent fusionner. Ceci est un problème quand on veut générer de grandes tables. En effet, si beaucoup de chaînes on déjà été stockées dans une table, la probabilité est grande qu'une chaîne supplémentaire fusionne avec une chaîne qui est déjà dans la table. Dans ce cas une partie de la chaîne est donc faite de mots de passe qui sont déjà dans la table. Il vaut mieux dans ce cas recommencer une nouvelle table avec une autre fonction de réduction. Le compromis temps-mémoire consiste à générer ℓ tables de m chaînes de longueur t a l'aide de ℓ fonctions de réduction et de ne stocker que les début et les fins de chaînes.

Une variante plus efficace que nous avons développée récemment consiste à générer des chaînes dans lesquelles on utilise une fonction de réduction différente

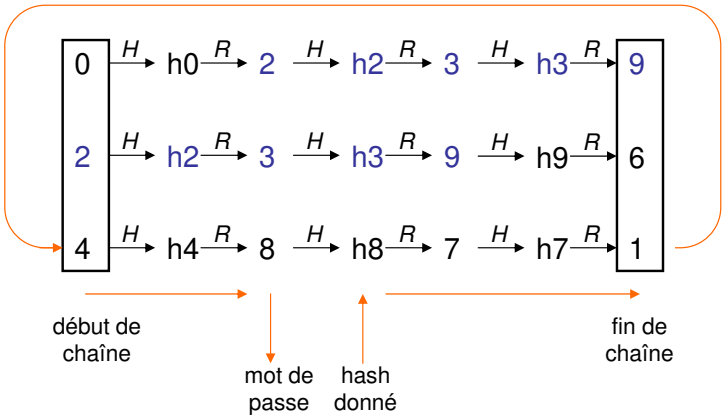


Fig. 1. Recherche du mot de passe "8" à partir de son empreinte "h8" à l'aide de trois chaînes dont on a stocké que le début et la fin.

pour chaque position dans la chaîne. On calcule donc la première réduction avec la fonction R_1, la deuxième avec la fonction R_2 et ainsi de suite. On réduit considérablement la probabilité de fusion, car pour que deux chaînes fusionnent il faut non seulement que le même mot de passe apparaisse dans les deux chaînes, mais qu'en plus il apparaisse à la même position dans la chaîne. Cet artifice permet de générer des tables beaucoup plus grandes ce qui à l'avantage de réduire le nombre d'opérations nécessaires à rechercher un mot de passe. Ce type de table est appelé table *rainbow*, ou arc-en-ciel, parce que chaque chaîne contient tout le spectre des fonctions de réduction. Pour rechercher à partir d'une empreinte un mot de passe dans une table rainbow on suppose d'abord qu'il se trouve dans la dernière colonne. On applique la dernière fonction de réduction et on cherche le résultat dans les fins de chaîne. Si on ne le trouve pas, on suppose que le mot de passe se trouve à l'avant-dernière position des chaînes. On applique l'avant-dernière réduction, un hashage et la dernière réduction pour à nouveau chercher le résultat dans les fins de chaînes stockés. Dans le pire des cas il faudra $\frac{t(t-1)}{2}$ opération de hashage pour trouver un mot de passe dans la table. Comme les table rainbow peuvent être t fois plus grandes que les tables classiques, on gagne au moins un facteur 2 à l'aide des tables rainbow. En réalité ce gain peut être bien plus élevé, les détails de cette méthode sont donnés dans [3].

3 Performances

Pour exprimer les performances d'un compromis temps-mémoire on utilise les variables suivantes :

- T : le temps nécessaires à retrouver un mot de passe, exprimé en nombre d'opérations de hashage ;

- M : la mémoire nécessaire à stocker les tables, mesurée en nombre de chaînes stockées ;
- N : le nombre de mots de passe possibles ;
- P : la probabilité que l'on trouve le mot de passe dans les tables qui ont été préparée.

Tous les compromis basés sur des chaînes suivent la relation suivante :

$$T \approx \frac{N^2}{M^2}$$

En d'autres mots, le produit du temps de cassage et du carré de la mémoire utilisée est constant. Une chiffre que l'on voit souvent en littérature est que les compromis temps-mémoire permettent de réduire le temps de cassage brut de $T = N$ avec une mémoire nulle à un temps égal à $T = N^{\frac{2}{3}}$ en utilisant une mémoire de $M = N^{\frac{2}{3}}$, ce qui correspond à réduire d'un tiers la longueur du mot de passe ou de la clef à trouver. Comme le temps de cassage augmente avec le carré de la mémoire disponible il est important d'utiliser la mémoire de la manière le plus efficace possible. Un première optimisation consiste à utiliser des tables *parfaites*, c'est à dire, des tables dans lesquelles on élimine les chaînes qui fusionnent. Ensuite il faut aussi choisir un format de stockage des chaînes qui utilise qui permette de stocker un maximum de chaînes dans un nombre d'octets donnés.

3.1 Tables parfaites

Les tables parfaites sont des tables dans lesquelles on retire les chaînes qui fusionnent. Dans le cas des chaînes rainbow cette opération est simple car les chaînes qui fusionnent ont des fins de chaîne identiques. Comme on doit de toute façon trier les chaînes d'après leur fin pour faciliter la recherche dans les tables, l'élimination des fusions se fait sans effort. A chaque fois que l'on retire un chaîne qui fusionne avec un autre on élimine pas seulement des mots de passes qui étaient à double mais aussi le début de la chaîne qui contient des mots de passe uniques. Pour compenser cette perte, il faut donc générer plus de chaînes, obtenir le taux de réussite désiré après élimination des fusions. Le nombre de chaînes sans fusions que l'on obtient après génération d'un nombre de chaînes données est illustré à la figure 2. Le nombre maximum de chaînes que l'on peut générer est égal N, car un peut idéalement utiliser chaque mot de passe comme un point de départ d'une chaîne. Dans notre cas, en générant 80 milliards de chaînes on obtiendra pas plus de 35 millions de chaînes sans fusion et on obtient déjà 30 millions de chaînes sans fusion après avoir généré 210 millions de chaînes.

3.2 Configuration optimale

Pour une mémoire M et un taux de succès P donnés nous allons maintenant essayer de trouver la configuration t, m, ℓ optimale qui va produire le temps de cassage le plus petit. Il se trouve que pour des tables parfaites, cette configuration

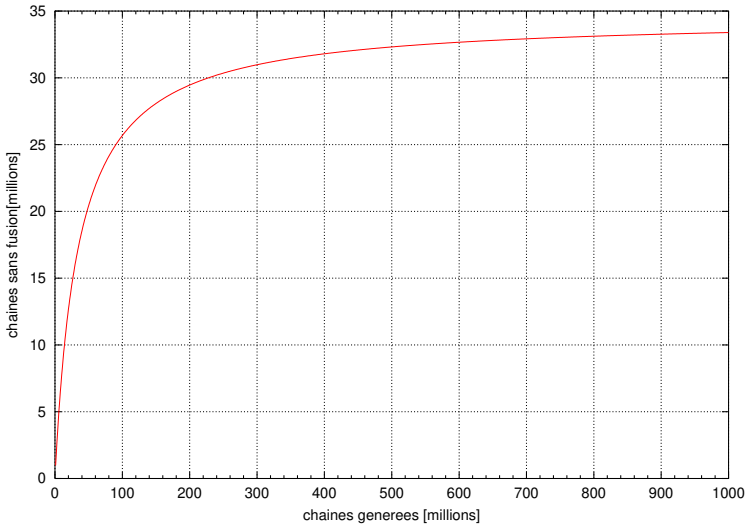


Fig. 2. Effort nécessaire pour obtenir des chaînes sans fusion.

est très simple à calculer. En premier nous allons essayer des tables parfaites aussi grandes que possible, pour limiter le nombre de tables. Le nombre maximal de chaînes de longueur t sans fusions que l'on peut trouver peut être approximé par la formule suivante :

$$m_{max}(t) \approx \frac{2N}{t}$$

Un fait intéressant est que la probabilité de succès d'une telle tables est toujours de 86% indépendamment de N ou t :

$$P_{max} = 1 - \left(1 - \frac{m_{max}}{N}\right)^t \approx 1 - e^{-t\frac{m_{max}}{N}} \approx 1 - e^{-2} = 86\% \qquad (1)$$

La sélection du nombre de tables ℓ est donc simple. Si la probabilité de réussite voulue est plus petite que 86%, il suffit d'une seule table. Sinon, il faut calculer le nombre de tables ℓ qui permet d'arriver au taux de réussite désiré. Comme le nombre de table doit être entier on arrondi vers le haut ce qui a pour effet de baisser le taux de réussite de chaque table un peu en dessous de 86%. Du nombre de tables ℓ découle le nombre de chaînes par table m puisque chaque table aura à disposition une part de mémoire égale à $\frac{M}{\ell}$. Connaissant le nombre de chaînes et le taux de réussite pour chaque table il est facile de calculer la longueur t des chaînes.

$$\ell = \lceil \frac{-\ln(1-P)}{2} \rceil \tag{2}$$

$$m = \frac{M}{\ell} \tag{3}$$

$$t = \frac{\ln(1-P)}{\ln(1-\frac{M}{\ell N})\ell} \approx \frac{N}{M}\ln(1-P) \tag{4}$$

Les formules ci-dessus nous permettent de trouver la configuration optimale du compromis temps-mémoire en fonction de la mémoire disponible et du taux de réussite escompté. Pour pouvoir analyser la performance du compromis il nous reste à calculer combien d'opérations seront nécessaires en moyenne pour retrouver un mot de passe grâce aux tables précalculées.

3.3 Travail à fournir

La structure régulière des tables rainbows permet de calculer exactement le nombre moyens d'opérations nécessaires pour retrouver un mot de passe, en incluant même les opérations dues aux fausses alarmes. Le cassage d'un mot de passe se fait en cherchant dans la dernière colonne de toutes les tables, puis dans l'avant-dernière colonne et ainsi de suite jusqu'à ce que le mot de passe soit trouvé ou que la première colonne est atteinte. Il y a donc au maximum ℓt recherches. Nous calculons le nombre moyen d'opérations de hashage en faisant la somme du nombre d'opérations nécessaires pour trouver un mot de passe dans une recherche donnée pondéré par la probabilité que le mot de passe soit trouvé dans cette recherche.

La probabilité de trouver un mot de passe lors de la k-ème recherche est :

$$p_k = \frac{m}{N}(1-\frac{m}{N})^{k-1}$$

Lors de la k-ème recherche on se trouve dans la colonne

$$c = t - \lfloor \frac{k}{\ell} \rfloor$$

Le nombre d'opérations effectuées pour la k-ème recherche (incluant les recherche précédentes) est égal à

$$T_k = \frac{(t-c)(t-c-1)}{2}$$

La probabilité q_c de tomber sur une fausse alarme lorsque l'on fait une recherche à partir d'une colonne c est égale à la probabilité de tomber sur une fin de chaîne existante moins la probabilité qu'il s'agisse de la bonne chaîne.

$$q_c = 1 - \prod_{i=c}^{i=t}(1-\frac{m_i}{N}) - \frac{m}{N}$$

avec

$$m_t = M \quad \text{et} \quad m_{i-1} = N \ln(1 - \frac{m_i}{N})$$

En combinant ces résultats on trouve que le nombre moyen d'opérations de hashage nécessaires pour retrouver un mot de passe est égal à :

$$T = \sum_{k=0}^{\substack{k=\ell t \\ c=t-\lfloor \frac{i}{t} \rfloor}} p_k \left(\frac{(t-c)(t-c-1)}{2} + \sum_{i=t}^{i=c} q_i i \right) + (1 - \frac{m}{N})^{\ell t} \left(\frac{t(t-1)}{2} + \sum_{i=t}^{i=1} q_i i \right) \tag{5}$$

Le deuxième terme de la somme correspond au travail fourni lorsque le mot de passe n'est trouvé dans d'aucune recherche. Ces formules ont été vérifiées par les mesures suivantes :

$N = 8.06e7, t = 4666,$ $m = 23.8e6, l = 5$	théorie	moyenne de 1000 mesures
calculs d'empreintes (moyenne)	4.30e6	4.11e6
calculs d'empreintes (max)	7.49e7	7.50e7
nombre de fausses alarmes (moyenne)	596	569
nombre de fausses alarmes (max)	12315	12309

Tab. 1. Performances calculées et mesurées des tables rainbow

3.4 Les graphes temps-mémoire

Maintenant que nous savons calculer les performances du compromis dans sa configuration optimale nous pouvons générer les courbes exactes de la relation entre le temps et la mémoire. Dans la figure 3 nous avons représenté ces courbes pour différents taux de réussite.

L'information que l'on retire de la figure 3 est que les tables rainbow génèrent des compromis qui suivent bien la fonction $T \approx N^2/M^2$. Cette relation est valable pour tous les taux de réussite, l'effet du taux est simplement de décaler les courbes. En décrivant l'effet du taux de réussite par une fonction f que nous appelons la *caractéristique* du compromis, nous obtenons un calcul exact du compromis :

$$T = \frac{N^2}{M^2} f(P) \tag{6}$$

La caractéristique du compromis peut être calculée avec les mêmes formules qui ont permis d'établir les courbes temps-mémoire. Le résultat est donné à la figure 4. On y voit comment f évolue en fonction du taux de réussite. Des paliers apparaissent dans la courbe à chaque fois qu'une table supplémentaire

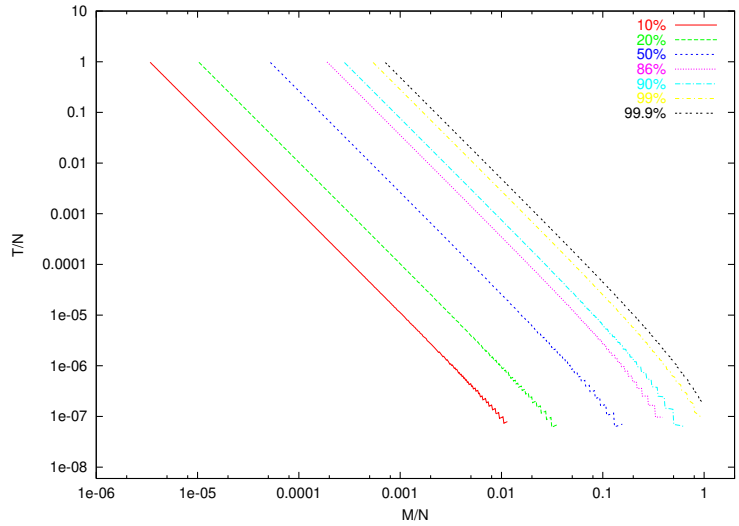


Fig. 3. Courbes temps-mémoire pour différents taux de réussite ($N = 1.6e07$).

est nécessaire pour obtenir le taux de réussite voulu, le premier étant à 86%. Au bas de chaque palier, le taux de réussite par table est proche de 86% par table, ce qui implique un effort énorme pour générer les tables. Il vaut donc souvent mieux "prendre le palier", c'est à dire utiliser une table supplémentaire lorsque le taux de réussite s'approche trop des 86%.

Avec les équations du chapitre 3.2 et le graphe de la caractéristique du compromis, nous avons toutes les informations nécessaires pour trouver la configuration optimale des tables et calculer le temps qu'il faudra pour trouver un mot de passe en fonction de la mémoire disponible. Nous pouvons donc maintenant nous atteler à l'implémentation efficace de notre casseur de mots de passe.

4 Implémentation de l'attaque

Avant de discuter des détails de l'implémentation d'un casseur de mots de passe, nous rappelons brièvement les deux types d'empreintes utilisés par les systèmes Windows.

Le LanManager hash (LM hash) est le type d'empreintes le plus ancien. Il fonctionne avec des mots de passe d'une longueur maximale de 14 caractères. Le mot de passe est d'abord coupé en deux blocs de 7 caractères. Puis toutes les minuscules sont remplacées par des majuscules. Finalement chacun des blocs est utilisé comme clef de 56 bits pour chiffrer un texte prédéfini à l'aide de l'algorithme DES. Les deux résultats de 64 bits chacun sont concaténés pour former le LM hash. Cette manière de faire est très peu efficace. En effet, alors qu'il

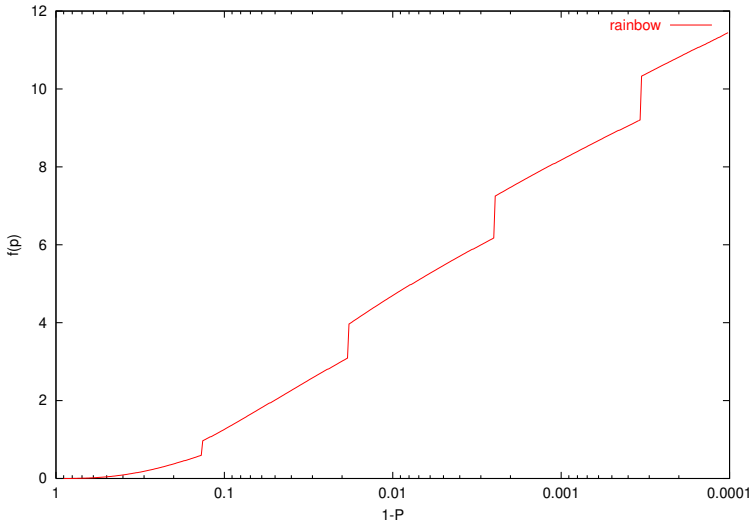


Fig. 4. La caractéristique temps-mémoire des tables rainbow. Les paliers apparaissent à chaque fois qu'une table supplémentaire est nécessaire pour atteindre le taux de réussite désiré.

est possible d'exprimer 2^{84} différents mots de passe alphanumériques de 14 caractères de long, ces mots de passe ne génèreront que 2^{36} moitiés d'empreintes différentes.

Le NT hash est un type d'empreinte plus récent. Il fonctionne avec des mots de passe de longueur arbitraire et ne transforme pas les minuscules en majuscules. L'empreinte est obtenue en appliquant l'algorithme de hashage MD4 sur le mot de passe. Le résultat est une empreinte de 128 bits. Cette méthode génère réellement 2^84 empreintes différentes pour les 2^84 mots de passes alphanumériques possibles.

Pour des raisons de compatibilité il se trouve que toutes les versions de Windows génères les deux types d'empreintes à chaque fois qu'un mot de passe d'un utilisateur est ajouté ou modifié. A partir de Windows 2000 la génération peut être désactivée en modifiant une clef dans le registre Windows. Dans Windows Server 2003 la génération des LM hash est désactivée par défaut.

La méthode la plus efficace pour casser des empreintes Windows est de s'attaquer d'abord aux deux moitiés du LM Hash pour trouver le mot de passe majuscule qui correspond à cette empreinte. Ensuite il ne reste plus qu'a vérifier quelle combinaison de majuscules et minuscules (au plus 16384 possibilités) donne le bon LM hash.

4.1 Optimisations

Les performances réelles d'un casseur à base d'un compromis temps-mémoire dépendent finalement de la vitesse des opération de hashage et du nombre de chaînes que l'on arrive stocker dans la mémoire disponible.

Efficacité du DES : Pour les opérations de hashage il suffit de trouver un bonne implémentation de DES, par exemple celle qui se trouve dans la librairie `libssl`. Notre utilisation de DES a la particularité que la clef de chiffrement change avec chaque nouvelle opération de chiffrement. L'utilisation d'une implémentation de DES en *bitslice* permettrait éventuellement d'augmenter les performances. Ce genre d'implémentation exécute par exemple 32 DES en parallèle en utilisant les mots de 32 bits du processeur pour stocker un bit de 32 différents DES. Les opérations de DES qui se font sur des bits peuvent ainsi être appliquée à 32 calculs en une seule opération de 32 bits. Cette méthode peut même être étendue à 64 bits avec des processeurs de 64 bits ou des opération de 64 bits de processeurs 32 bits (p.ex. MMX sur processeur Pentium). Une implémentation en bitslice est par exemple utilisée par le casseur de mots de passe en force brute `John the Ripper[2]`.

Efficacité du stockage : L'optimisation du stockage des tables est plus important que l'optimisation de DES puisque le temps de cassage diminue avec le carré du nombre de chaînes stockées. Une méthode naïve consisterait à stocker pour chaque chaîne les sept octets qui correspondent au mot de passe du début de la chaîne et les sept octets qui correspondent à la fin. C'est ce qui est par exemple fait dans le logiciel `rainbowcrack`. Au fait, pour faciliter l'alignement des chaînes, ce logiciel utilise 16 octets par chaînes.

Binarisation : La première constatation que l'on doit faire est qu'il n'y a que $2^{36.23}$ mots de passes alphanumériques d'un à sept caractères. On ne devrait donc pas utiliser plus de 37 bits pour stocker un mot de passe. Il suffit de définir une représentation que nous appelons représentation *binaire*, qui assigne à chaque mot de passe un chiffre entre 0 et $2^{36.23}$. Une méthode simple consiste à considérer les mots de passe alphanumériques comme des nombres en base 37 faits des 36 caractères alphanumériques et d'un caractère vide pour les mots de passe courts. En représentation binaire une chaîne peut être stockées à l'aide de 74 bits, ce qui est 1.5 fois plus petit que les 14 octets proposé ci-dessus. Le temps de cassage en est réduit d'un facteur de 2.3 (1.5^2). Au fait, les débuts de chaînes que nous devons stocker sont des mots de passe que nous avons choisis arbitrairement lors de la création des tables. Si nous avons besoins de m_0 débuts de chaînes pour générer les tables, nous choisirons donc les mot de passe qui ont les représentation binaires allant de 0 à $m_0 - 1$. Il nous faudra donc que $\log_2(m_0)$ bits pour stocker les débuts de chaînes. Cette manière de faire permet facilement de stocker un début de chaîne alphanumérique dans un entier de 32 bits.

Indexation des fins de chaînes : Les fins de chaînes peuvent prendre des valeur quelconques parmi les $2^{36.23}$ possibilités. Par contre ces valeurs seront triées par ordre croissant. Les valeurs qui se suivent ont donc des préfixes communs. On peut dès lors ne pas stocker ces préfixes et créer une table d'indexation qui indique à partir de quelle position on trouve un préfixe donné. Un exemple simple consiste à créer 36 fichiers qui correspondent à la première lettre des fins de chaînes et de stocker les chaînes dans les fichiers correspondants. Il n'est donc plus nécessaire de stocker le premier caractère des fins de chaînes puisqu'il est identique à toutes les chaînes stockées dans un fichier. Il se trouve que des mots de passe de 6 caractères alphanumériques peuvent être représentés de manière binaire à l'aide de 32 bits, ce qui permet donc de stocker les fin de chaînes dans un entier de 32 bits. Avec ce qui a été dit précédemment sur les début de chaînes nous pouvons donc stocker une chaîne dans 8 octets, ce qui réduit le temps de cassage d'un facteur 3 par rapport des chaînes de 14 octets ou même un facteur 4 par rapport à des chaînes sur 16 octets comme dans `rainbowcrack`. C'est la solution que nous avons implémentée dans la démonstration que nous avions mis en ligne au cours de l'été 2003[4]. En moins d'un Go nous avons ainsi pu stocker 115 millions de chaînes (5 tables à 23.8 millions de chaînes).

L'utilisation de préfixes plus longs permet d'augmenter encore le gain du à l'indexation. Pour chaque bit supplémentaire que l'on considère dans le préfixe, il y a un bit de moins à stocker pour chaque chaîne mais deux fois plus d'indexes à générer. Dans le cas qui nous intéresse, la solution optimale se trouve autour de 20 bits de préfixe et 16 bits stockés par chaîne. Le stockage de l'index lui-même utilise moins de 10% de la mémoire nécessaire à stocker une table. On arrive donc finalement à 6 octets par chaîne, ce qui permet avec la même quantité de mémoire, de casser les mots de passe 5.4 fois plus vite qu'avec un représentation sur 14 octets.

5 Conclusions

Les compromis temps-mémoire permettent d'obtenir des résultats exceptionnels pour le cassage de mots de passe Windows. Ceci est du à plusieurs raisons. D'abord, les empreintes des systèmes Windows (LM hash et NT hash) peuvent être calculées à l'avance. A notre connaissance, Windows est le seul système d'exploitation actuel ou cela est possible. Dans toutes les variantes d'Unix, par exemple, une valeur aléatoire (le sel) est ajoutée lors de la calculation d'une empreinte. Cette valeur n'étant pas connue à l'avance on ne peut pas préparer des tables à l'avance. La deuxième raison qui rend l'utilisation de compromis temps-mémoire idéale pour casser des mots de passe est qu'il s'agit d'un problème avec une complexité réduite. En effet l'ensemble des mots de passe qui sont potentiellement choisis par des utilisateurs est bien plus petit que les 2^{56} clés possibles pour DES. Si on devait casser un système qui utilise DES avec des clés arbitraires, on arriverait peut-être à casser une clef de 56 bits en quelques heures, mais il faudrait des années de calcul pour préparer les tables !

Outre les mots de passe Windows nous n'avons pas trouvé systèmes qui n'utilisent pas de composante aléatoire (sel, vecteur d'initialisation) et qui sont d'une complexité raisonnablement faible. Nous avons appliqué avec succès notre méthode pour casser le LM hash de mots de passe alphanumériques ($N = 2^{36}$) et des mots de passe faits de chiffres, de lettres et de 16 caractères spéciaux ($N = 2^{40}$). La seule parade proposée par Microsoft contre ce genre d'attaques et de désactiver la génération du LM Hash. Ceci ne peut être qu'une solution momentanée. En optimisant encore un peu l'implémentation et en profitant des performances de machines plus récentes il devient aussi possible de casser directement des NT Hash puisqu'ils ne contiennent pas non plus de sel. Il y a par exemple moins que 2^{48} mots de passes alphanumériques de 8 caractères à casse mixte. Comme les compromis temps-mémoire bénéficient doublement de la loi de Moore (par l'augmentation de la mémoire disponible et de la vitesse des processeurs) il faut espérer que Microsoft ne tarde pas à proposer un nouveau type d'empreintes pour préserver la sécurité des mots de passe de ses systèmes d'exploitation.

Remerciement

Le travail présenté dans cet article a été supporté, en partie, par le Pôle de Recherche National en Systèmes Mobiles d'Information et de Communication (NCCR-MICS), un centre supporté par le Fonds National Suisse pour la Recherche Scientifique (subside 5005-67322).

Références

1. M. E. Hellman, A cryptanalytic time-memory trade off, *IEEE Transactions on Information Theory*, IT-26, pp. 401–406, 1980.
2. John the Ripper, `http://www.openwall.com/john`. Casseur de mots de passe par force brute.
3. Philippe Oechslin, Making a faster cryptanalytic time-memory trade off, *Proceeding of IACR Crypto 2003*, 2003.
4. Philippe Oechslin, Claude Hochreutiner et Luca Wullschleger, Les compromis temps-mémoire ou comment cracker un mot de passe Windows en 5 secondes, *Multisystem and Internet Security Cookbook (MISC)*, numéro 10, 2003.

Index des auteurs