Préface

La sécurité informatique est une discipline qui évolue rapidement, et chaque année voit apparaître son lot de nouveautés : nouvelles applications et nouvelles technologies apportent de nouvelles menaces et nous incitent à élaborer de nouveaux mécanismes de protection. La cryptographie a pris de l'importance lorsque le commerce a commencé à se développer sur les réseaux. Les virus se sont propagés lorsque de plus en plus de personnes ont été équipées d'ordinateur, et s'échangeaient des documents. Actuellement, la "mode" est plutôt aux vers, favorisée par l'essor des réseaux.

Par ailleurs, la sécurité informatique est un thème porteur. De nombreux cursus universitaires en incluent dans les cours. De plus en plus d'entreprises se bâtissent autour de services liés à ce secteur. L'Etat vient d'allouer des moyens importants pour encourager la recherche dans cette discipline. Même la Communauté Européenne se décider de se doter d'une agence spécialisée sur ces problématiques.

Cette rapide et incessante évolution d'un côté, et ce besoin accru et perceptible de l'autre ne doivent pas nous faire oublier l'enjeu principal, l'objet même de notre attention dès qu'on s'intéresse à la sécurité : l'information. Celle-ci possède de nombreuses formes, et elle ne se limitent pas aux données sagement stockées sur des disques durs. Par exemple, la connaissance d'un échange entre deux entités, même sans connaître la teneur de l'échange, constitue une information en soit. Bref, on le sait maintenant, la sécurité ne dépend pas uniquement de moyens techniques ou scientifiques, mais sans ces moyens, point de salut.

Ce domaine repose sur des défis, qui en font probablement une discipline à part, passionnante et difficile à appréhender. D'une part, il ne s'agit pas d'une discipline à part entière, mais d'un faisceau complexe mêlant à la fois mathématiques, électronique, réseau, programmation ou encore système. Mais cette richesse et cette diversité ne se limitent pas uniquement, comme nous l'avons vu, à une composante technique ou scientifique : le ¡¡ facteur humain ¿¿ entre en ligne de compte. D'autre part, il s'agit d'un domaine qui demande d'être à la fois réactif et actif. L'apparition d'une nouvelle faille requiert, selon l'importance de l'information menacée, une réponse plus ou moins rapide. Rappelons qu'une nouvelle faille ne se limite pas à un "buffer overflow" dans telle ou telle application. Il pourrait tout aussi bien s'agir d'un problème plus fondamental qui nécessiterait une refonte complète du système d'information au sens large. Si les cryptographes l'ont bien appréhendé, ce doute permanent, ce constante soupçon, n'est pas courant en informatique. Un ordinateur, c'est 0 ou 1, et rien d'autre, mais pas quand il est question de sécurité. Le comportement parfois aléatoire et souvent surprenant voire déroutant de certains systèmes d'exploitation lorsqu'une même opération est répétée conduit les utilisateurs à considérer l'outil informatique comme suspect immédiat au moindre problème. L'appaisement de cette angoisse passe par un apprentissage et une connaissance

sans cesse accrue qui permet alors d'anticiper les menaces : de réactif on devient actif.

C'est en partant de ces constats que nous nous sommes décidés à organiser le Symposium sur la Sécurité des Technologies de l'Information et de la Communication. Nous espérons que vous prendrez autant de plaisir à y assister que nous en avons eu à l'organiser, ce qui me permets une habile et subtile transition vers les remerciements.

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, dont l'engouement pour ce projet aura été un constant encouragement, et qui ont eu la lourde tâche de statuer sur les 43 propositions envoyées pour n'en retenir que 14. Nous sommes reconnaissants envers les auteurs de toutes ces propositions et les conférenciers invités, Yves Correc, le GdI Desvignes, Nicolas Dubée, Philippe Wolf, Éric Wegrzynowski et Anne Cantéro de nous avoir permis de concevoir ce programme attractif. Nous remercions également nos différents sponsors, la revue MISC, l'ESAT, le CEA, Cartel Sécurité, Supélec, SpeKa Networks et la SEE qui nous ont donné les moyens matériels d'arriver à nos fins. Nous remercions également Pascal Lointier et Robert Longeon pour les conseils qu'ils ont bien voulu nous prodiguer. Merci également à Arnaud Metzler pour le bon sens qui nous fait parfois défaut, et à Katia Pacquet pour le sens artistique qui nous fait systématiquement défaut.

Cette partie est toujours la plus délicate car il ne s'agirait pas d'oublier quelqu'un. Donc, pour être certain que cela n'arrive pas, nous terminerons en remerciant les personnes que nous aurions pu oublier dans la liste ci-dessus. Si je faisais de la sécurité, je dirais que je conclus donc en instaurant la politique par défaut.

Juin 2003 — Frédéric Raynal pour le comité d'organisation

Organisation

SSTIC'03 est organisé par l'Ecole Supérieure d'Electricité en coopération avec l'Ecole Supérieure et d'Application des Transmissions, le Commissariat à l'Energie Atomique, la société Cartel Sécurité et le journal de la sécurité informatique MISC.

Comité d'organisation

Christophe Bidan	Supélec
Philippe Biondi	Cartel Sécurité
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
Frédéric Raynal (Président)	MISC Magazine

Comité de programme

Gildas Avoine	EPFL
Christophe Bidan	Supélec
Renaud Bidou	
Philippe Biondi	Cartel Sécurité
Cédric Blanchet	Cartel Sécurité
Patrick Chambet	Edelweb
Yves Correc	CELAR
Eric Detoisien	
Thiébaut Devergranne	
Eric Filiol	ESAT
Nicolas Fischbach	Colt Telecom / Securite.org
Gombault Sylvain	ENST Bretagne
Pascal Lointier	CLUSIF
Robert Longeon	CNRS
Thierry Martineau	ESAT
Marc Mouffron	EADS Telecom
Laurent Oudot	CEA/DAM
Frédéric Raynal	MISC magazine
Ludovic Rousseau	Gemplus
Marc Rybowicz	Université de Limoges
Franck Veysset	France Télécom R&D

Sponsors

- Commissariat à l'Energie Atomique
- Cartel Sécurité
- Ecole Supérieure et d'Application des Transmissions
- MISC - Le journal de la sécurité informatique
- Société Speka Networks
- Société de l'Électricité, de l'Électronique, et des Technologies de l'Information et de la Communication

Table des matières

Conférences invitées

UML as a *honeypot*

Michaël Hervieux and Thomas Meurisse

ENSEIRB - 33402 Talence,
{hervieux,meurisse}@enseirb.fr

Résumé

Notre papier présente une nouvelle technique d'implantation d'un honeypot ou pot à miel. L'objectif de ce type d'outils est de recueillir de nouvelles informations sur les techniques pouvant être employées par un éventuel attaquant.

La solution la plus simple consiste évidemment à sacrifier tout simplement une machine. La supercherie est alors certaine d'être complète puisque le système sur lequel le pirate va se loguer est un système réel. Toutefois, l'intérêt d'un pot à miel est essentiellement de recueillir des logs pour pouvoir ensuite analyser une attaque. Enregistrer les logs sur la machine dont le pirate prendra le contrôle n'apparaît donc pas comme une bonne solution.

La solution proposée dans ce document consiste à se servir d'une machine virtuelle (comme VMware, Boch, UML, ...) UML a été choisi pour sa facilité d'utilisation et son code source était disponible et donc modifiable. L'inconvénient vient alors que de nombreux détails vont permettre à l'attaquant d'observer qu'il ne se trouve pas sur une véritable machine : des noms de devices étranges (ubd0, ...), des commandes aux résultats insolites (dmesg, ...), un */proc* plus que douteux. Laisser une machine virtuelle en l'état présente un autre inconvénient : le manque d'activité sur celle-ci. Pour palier à cela, différents scripts peuvent être mis en place (simulation de connexion ssh, http ou ftp). Simuler la connexion de machines diverses (créer une fausse topologie de réseau) apparaît donc comme un atout majeur pour tromper d'avantage encore le pirate. En outre le système UML propose un moyen puissant et simple de recueillir les logs de tout ce qui aura pu être tapé. L'UML s'exécutant sur la machine hôte, l'intégralité du trafic réseau est controlée depuis cette dernière. Les logs ne sont donc pas enregistrés sur l'UML mais directement sur la machine hôte : une seule et unique machine est nécessaire pour atteindre le but fixé.

1 Introduction

UML (User Mode Linux) permet d'avoir à sa disposition un système Linux virtuel tournant sur un système Linux. L'intérêt est de pouvoir tester, sans risque, de nouveaux programmes ou de nouveaux noyaux. En effet, tout ce qui sera effectué sur ce linux UML sera, d'un point de vue du vrai système, considéré comme lancé en tant qu'utilisateur sans privilèges particuliers et, par conséquent, ne pourra endommager le système principal, hébergeur de l'UML.

Nous avons repris cet outil pour créer un Honeypot. L'intérêt d'un "pot à miel" est en général d'attirer les pirates sur un leurre, que ce soit en créant un faux service ou un faux système. L'UML, en tant que pot à miel, a pour but de prétendre être une machine réelle. Cet article va insister sur différents aspects qui permettront de leurrer au mieux un assaillant, que ce soit au niveau système ou de l'activité réseau. Si l'utilité de faire d'un UML un honeypot ressemble à un challenge (modélisation d'un linux le plus vraisemblable possible), l'observation d'une personne malveillante agissant sur l'UML (sachant qu'elle ne pourra mettre en péril notre système hébergeur) permettra éventuellement de découvrir de nouvelles attaques, qui pourraient être mises en oeuvre sur des ordinateurs où l'issue serait plus critique. Ainsi, cet article présentera les différents moyens pour tracer l'activité malveillante.

2 Protection du système de l'UML

2.1 Introduction

Cet article va ci-après présenter les différents aspects systèmes qui trahissent UML afin de résoudre le problème suivant : "Comment éviter qu'une personne malveillante ayant pris la main sur l'UML ne puisse s'apercevoir que ce n'est qu'un honeypot ?". En effet, le système UML est quelque peu différent d'un système linux classique. Très peu de temps sera nécessaire à un utilisateur averti pour s'en rendre compte. Ci-après sont donc présentés différents problèmes et leurs résolutions. Cela représentera bien souvent du masquage d'information relative au système UML.

2.2 Montages

Le montage du /

Lors d'un premier lancement du système UML sur un file system (rootfs) proposé directement sur le site d'UML, le système ne démarre pas correctement (ce problème se pose sur le root_fs_debian2.2_small proposé par le site). L'option "devfs = mount" dans la ligne de commande de lancement d'UML permet au système virtuel d'avoir un ensemble de périphériques qui font que le système UML puisse démarrer correctement. Toutefois, après listage du répertoire */dev*, qui est en fait un répertoire monté automatiquement au démarrage, l'architecture de celui-ci est bien différente de celles classiques. La solution est donc simple : il faut soit même créer ses devices adéquates et ne pas utiliser l'option devfs.

Après analyse du problème, on peut se rendre compte que les fichiers de configuration présents dans le */etc* font référence à des périphériques qui existent dans le */dev* monté grâce à l'option devfs mais qui n'existent pas dans le */dev* du file system proposé par défaut. Il faut savoir aussi que l'UML fonctionne de la façon suivante lors du démarrage : il ne monte que les partitions présentes dans le fichier */etc/mtab*, comme l'illustre ce test :

```
# more /etc/mtab
/dev/n/importe/quoi / ext2 rw 0 0
```

Après redémarrage du système UML, on liste les partitions et on a :

```
# mount
/dev/n/importe/quoi on / type ext2 (rw)
```

On peut alors lister le répertoire de devices de l'UML sans voir apparaître le fichier */dev/n/importe/quoi*. On reviendra plus loin sur le fonctionnement de la commande "mount". Pour booter, on sait donc qu'il faut désormais modifier le fichier */etc/mtab*. On indique donc dedans le nom d'un device habituel (*/dev/hda1* par exemple) que l'on va donc créer à l'aide de la commande mknod et on pense à modifier le fichier */etc/fstab* pour qu'il soit cohérent.

Le Re Montage du /

Désormais, le système boote et place le / sur ce que l'on croit être */dev/hda1*. Toutefois si l'on veut remonter la racine ailleurs, via "mount /dev/hda1 /ailleurs", le système indique que le block device n'est pas valide. Le problème est que la partition / provenant du rootfs et indiquée en ligne de commande (ubd0=...) est accessible sous UML via l'inode de majeur 98 (UBD_MAJOR) et de mineur 0. Si l'on indique donc que */dev/hda1* est ainsi, on pourra remonter le /, mais on verra que ce n'est pas normal. La solution est donc du côté du noyau UML, que l'on va recompiler en indiquant dans le fichier *include/linux/major.h* que la macro constante UBD_MAJOR vaut 3 (majeurs des disques IDE). On intervertit donc les valeurs 3 et 98 dans ce fichier. Puis on recompile, pour se rendre compte que cela fonctionne comme on le veut. Si l'on veut enfin utiliser le mineur 1, il suffit d'indiquer en ligne de commande de démarrage de l'UML que ubd1 vaut le file system que l'on souhaite et qui sera associé au mineur 1. Il est à noter qu'il est alors nécessaire d'indiquer que l'UML bootera sur le device ubd1 et donc ajouter en ligne de commande "root=/dev/ubd1", ceci si l'on ne veut pas de mineur 0. Désormais, le mount et le df nous donne chacun des informations vraisemblables.

Montage de l'hôte

UML permet de remonter la partition racine de l'hôte. On peut alors accéder facilement aux données de celui-ci sous l'UML. Toutefois, cet intérêt se révèle être un lourd inconvénient lors d'une utilisation en tant qu'honeypot. En effet, on ne veut pas, dans ce cas, pouvoir accéder à l'hôte, qui serait certes accessible qu'avec les droits de l'utilisateur sans privilège, qui aurait lancé UML, car même si l'utilisateur était root sous l'UML, il ne pourrait pas accéder aux fichiers root de l'hôte.

Pour pouvoir ainsi monter à l'intérieur de l'UML la partition / de l'hôte, il suffit d'exécuter la commande "mount /dev/n/importe/quoi /hote -t hostfs". Le type de file system "hostfs" fait parti des ajouts du patch UML que l'on applique sur un noyau normal. Il suffit donc de configurer le noyau UML pour qu'il ne le prenne pas en charge et l'on ne pourra plus monter le / de l'hôte sous l'UML (il suffit de répondre négativement dans le fichier de configuration du noyau pour l'UML : "CONFIG_HOSTFS = n").

La commande mount

Il est temps de préciser le fonctionnement un peu inhabituel de la commande "mount" sous l'UML avec les options hostfs et hppfs. En effet, lorsque l'on utilise certains types de file system (comme hostfs ou hppfs (voir plus loin)), on se rend compte qu'il n'y a pas de vérification de l'argument correspondant au nom du device à monter. C'est pourquoi celui-ci peut ne pas exister (cf. */dev/n/importe/quoi* vu précédemment). Il faut donc être prudent et bien penser à créer le */dev/hda1* correspondant, même s'il ne sert pas réellement pour le montage du slash, car tout est histoire de leurre. Toutefois, ses options ne seront pas disponibles pour l'UML, puisqu'elles seront enlevé du noyau.

2.3 La partition /proc

Une partition très intéressante est la partition montée sur */proc*, qui contient habituellement toutes les informations sur le noyau, les processus, ... Le noyau UML étant quelque peu différent d'un noyau linux habituel, il s'avère que cela se remarque dans le */proc*. En effet, si l'on tape sous l'UML :

```
#more /proc/cmdline
ubd0=rootfs eth0=tuntap,,fr:fd:0:0:0:1,192.168.0.5 root=/dev/ubd0
```

Alors que sous un linux habituel, on aurait eu un résultat du style :

```
auto BOOT_IMAGE=Debian2_4_18 ro root=305
```

On a alors une information capitale : le noyau a été lancé de façon très étrange. Et si l'on regarde les différentes informations dans les divers fichiers du */proc*, on aura plus aucun doute sur la virtualité du système.

L'HoneyPot Proc File System

Une solution existe toutefois sous UML dans les utilitaires, qui sont livrés, avec HPPFS, un file system au fonctionnement un peu particulier. En effet, il faut exécuter un script Perl sur l'hôte (honeypot.pl), dans le répertoire de lancement de l'UML. Ce script crée un répertoire proc et une sous-arborescence avec des entités au nom proche de ceux rencontrés dans un */proc* habituel. Chacun de ces fichiers peut être édité, on place alors les informations que l'on veut dedans.

Sous l'UML, il suffit de monter ce file system de la façon suivante "mount /dev/n/importe/quoi /proc -t hppfs" (il ne faut donc pas avoir monté préalablement le véritable */proc*). Désormais, lorsqu'on liste le */proc* de l'UML, on aperçoit les contenus des fichiers correspondants du *./proc* de l'hôte. On peut donc "corriger" les informations étranges de l'UML, en les indiquant sous l'hôte. De plus, l'outil hppfs permet d'éventuellemet accéder au /proc de l'hôte. Par exemple, pour accéder en lecture au */proc/cmdline* de l'hôte, il faudrait avoir le fichier *./proc/cmdline/r* dans le répertoire de l'hôte. HPPFS fonctionne par requête via socket Unix.

Enfin, il est nécessaire de modifier les sources du noyau UML pour que le file system "hppfs" soit renommé en "procfs", afin qu'une fois encore, l'attaquant

n'y voit que du feu (on doit donc par conséquent supprimer aussi le véritable file system "procfs" du noyau). Il est à noter que puisque "mount" à un comportement étrange avec l'option hppfs (et donc procfs désormais), on ne pourra pas remonter le */proc* ailleurs.

Le travail consiste désormais à renseigner les fichiers générés par hppfs convenablement afin d'avoir un */proc* sous l'UML cohérent (cmdline, version, cpuinfo, etc...). Il est ainsi intéressant de modifier le cpuinfo en fonction de ce que l'on veut prétendre avoir comme machine. Il faut avoir à l'esprit qu'UML n'est qu'un programme et que ces performances dépendent de ce qui lui a été alloué par l'hôte (mémoire, espace disque, ...). Il est donc pertinent de ne pas prétendre avoir un système trop puissant auquel on aurait alloué simplement 16 Mo de RAM.

Dmesg

Un petit programme qui peut trahir pleinement l'UML est "dmesg". En effet, son but est de regarder dans le ring buffer du noyau (kmsg) les messages enregistrés via syslog. Ainsi, les messages d'erreurs et certains messages d'informations comme le "kernel command line" y sont présents. Ce ring buffer, pourtant normalement accessible via le */proc/kmsg*, n'est pas modifiable grâce à hppfs. C'est pourquoi, une nouvelle fois, la solution va se trouver du côté de la recompilation des sources du noyau UML.

Puisque dmesg puise dans le segment de messages de kernel, on va, plutôt que d'en empêcher l'accès en lecture, modifier l'écriture qui est faite dedans. Ainsi, la modification de la fonction printk (qui écrit dans le kernel), présente dans les sources du noyau dans *kernel/printk.c*, qui consistera à ce que tout message, que l'on veut insérer, passe par un filtre, permettra d'accepter (ou non) l'écriture du message dans le kernel. Il est aussi possible de modifier certains messages ou d'en ajouter d'autres. Il suffit donc d'écrire ses propres règles en langage C dans le code de la fonction printk et de recompiler le noyau UML.

L'avantage de cette technique est que, vu de l'UML, on ne pourra pas voir la supercherie. Il ne reste donc plus qu'à filtrer correctement les messages, pour que seuls des messages cohérents figurent dans le */proc/kmsg* (et donc dans le dmesg) de l'UML.

2.4 Divers indices

Il faut être conscient que les indices les plus frappants peuvent être aussi les plus faciles à cacher. Il faut par exemple songer à donner une taille de la partition de l'UML suffisamment importante pour ne pas paraître ridicule. Par exemple, étendre la partition à 200 Mo, via la commande :

```
dd if=/dev/zero of=root_fs_debian2.2 bs=1k count=1 seek=$$[200*1024]
resize2fs root_fs_debian2.2 204800
```

Il en est de même avec la mémoire en passant "mem=64M' " en argument de l'UML lors de son lancement. Puisque le système UML sera un sous-système

de l'hôte, il est plus que pertinent d'associer la taille de la mémoire allouée à la taille de la partition et à la fréquence du processeur. On peut par exemple simuler une machine de quelques années, faible en espace disque et en mémoire. Toutefois, il faudrait songer à ne pas aller trop dans l'excès, car le système virtuel est finalement relativement rapide. On peut donc songer à simplement ralentir le processus UML.

D'autre part, il peut être intéressant de penser à effacer toutes les commandes, que l'on aura faite au lancement de l'UML pour installer le réseau virtuel, ou lancer d'éventuel faux service., grâce à un "history -c". Il faudra faire de même pour les scripts de démarrage s'ils ne sont pas habituels. Il faut privilégier le fait que l'attaquant ne voit rien à un démarrage tout automatique par exemple. Il faut bien entendu penser à configurer l'UML en changeant son nom d'hôte qui est par défaut "user-mode". On peut créer d'autres utilisateurs et peut-être changer les mots de passe triviaux du guest et du root, installer des applications, des services (par exemple un wu-ftp, pour que la prise de main sur l'UML soit plus évidente).

Les options "ide" et "fakehd" sont disponibles au lancement en ligne de commande, toutefois il semble qu'elles ne permettent simplement l'ajout de blocks devices (ce que l'on peut très bien faire soi-même à la main, comme on l'a fait pour le */dev/hda1*). Leur intérêt ne se présente sans doute qu'avec l'utilisation de l'option devfs, que l'on s'est interdit d'utiliser.

Pour les problèmes posés par hdparm (certaines informations retournées par celui-ci produisent un message d'erreur), il suffit dans un premier temps de considérer qu'il n y a pas de disque IDE, mais simplement des disques SCSI et de changer les devices hda1 en sda1).

Enfin, il faut garder à l'esprit que l'UML peut être vulnérable. On ne doit pas simplement tenter d'empêcher que l'attaquant se rende compte qu'il se trouve sur un honeypot, mais on doit aussi protéger l'hôte dans le cas où l'assaillant prendrait la main de l'UML.

3 Simulation - Topologie réseau

3.1 Introduction

Pour créer un bon honeypot, il est important de faire croire au pirate qui aura pris la main sur celui-ci qu'il se trouve sur une machine ayant une certaine importance et donc sur laquelle l'activité n'est pas nulle. Pour donner l'illusion que plusieurs machines sont effectivement connectées sur le système UML, les possibilités offertes par iptables, le firewall de Linux pour les noyaux 2.4, vont être mises en oeuvre.

3.2 Topologie du réseau

Le système UML constitue une machine qu'il va falloir installer sur le réseau afin que diverses attaques puissent y être menées. Deux possibilités sont envisageables quant à l'installation du système UML :

– le système hôte sert de passerelle
– le système hôte joue le rôle de bridge

Ces deux solutions sont envisageables et apportent chacune un intérêt. Le bridge, étant difficilement détectable , permettra d'être plus furtif. Quant à la passerelle elle permettra de faciliter la simulation de multiples connexions puisque le fait , que les différentes adresses MAC provenant d'IPs variées soient toutes identiques, deviendra tout a fait logique.

La solution de la passerelle a été retenue pour cette raison. En outre, dans le milieu des entreprises, une telle disposition est classique et ne renseignera donc pas l'éventuel pirate.

L'hôte va donc jouer le rôle de passerelle pour le système UML ce qui permettra de contrôler tout le flux réseau émis de ce dernier.

3.3 Simulation

Une machine sur laquelle l'activité est nulle est rarement très intéressante et risque donc de rebuter un attaquant. Pour palier à ce problème, une certaine activité va être simulée sur le système UML. Cette activité peut être simulée de plusieurs façons :

– scripts divers simulant des connexions ssh, http ou ftp
– mise en place de services sur le système UML : ftp, http, serveur IRC, ...

3.4 Simulation de plusieurs connexions

Les différents scripts qui auront pu être réalisés vont être lancés depuis la machine hôte. Or cela pourra paraître suspect à un attaquant que toutes les connections effectuées sur le système UML le soit depuis la même machine. Pour éviter cela , des translations d'adresses vont être mises en place.

Un critère pour différencier les différents scripts et donc permettre la silation d'une multitude de machines est nécessaire. Plusieurs solutions pour différencier les scripts sont envisageables :

– le pid du script
– le port source que le script va utiliser
– uid de l'utilisateur exécutant le script

Les deux premiers critères, quoi que tout aussi pertinent, sont plus difficiles à mettre en place. Le dernier critère ne nécessite que la création d'utilisateur fictif (un par machine que l'on souhaite simuler).

Pour effectuer la translation a proprement dîtes, le module owner d'iptables va être utilisé. Pour chaque utilisateur, une règle suivant le modèle suivant va être mise en place :

```
iptables -t nat -A POSTROUTING -m owner --uid-owner id-faux-user
                               -j SNAT --to ip-fausse-machine
```

Plusieurs sessions ssh peuvent ensuite être lancées par les différents utilisateurs fictifs et on obtient bien le résultat attendu. Toutefois, une observation plus

fine met en avant le fait que les ports sources de ces différentes connexions se suivent ou sont très proches. Ceci vient du fait que l'algorithme par défaut de Linux pour le choix des ports sources consiste tout simplement à prendre à chaque fois le port suivant disponible. Des connexions faîtes dans un temps rapproché vont donc logiquement avoir des ports très voisins. Ce problème se contourne facilement en utilisant un patch pour le noyau de l'hôte comme *Grsecurity* par exemple. Ce dernier permet en effet de choisir les ports sources utilisés de façon aléatoire.

3.5 Réponse au ping

Plusieurs machines semblent connectées sur l'UML et le pirate va rapidement les remarquer. Nul doute qu'il risque alors de vérifier si ces machines existent bien. La solution la plus évidente et la plus simple pour ce dernier consiste tout simplement à pinguer les différentes fausses machines. Dans l'état actuel, aucune réponse ne va être reçue.

L'hôte constitue cependant la passerelle du système UML et va donc pouvoir interagir sur les paquets émis par ce dernier. Ainsi les paquets correspondants à un ping vont pouvoir être redirigés directement sur l'hôte grâce à la commande suivante :

```
iptables -t nat -A PREROUTING -p icmp -d ip-fausse-machine -j DNAT
                              --to ip-machine-hote
```

3.6 Modification du TTL

Le Time To Live (TTL) des différents paquets est cependant le même, ce qui peut surprendre si le nombre de machines simulées ou la complexité du réseau est importante. Un patch disponible pour iptables permet de modifier cette information.

Le problème qui est alors posé est, qu'outre cette modification du TTL, l'adresse source va dans le même temps être modifiée. Le paquet va donc à la fois devoir subir une modification dans la table nat (pour la translation d'adresse) et dans la table mangle (pour la modification du TTL). Toutefois le TTL va devoir être fixé suivant la valeur de cette adresse source qui caractérise la fausse machine. Il faudrait donc que la table nat agisse avant la table mangle afin qu'on obtienne le résultat escompté. Malheureusement, ce n'est pas le comportement par défaut de iptables. Une légère modification dans le noyau de l'hôte va être nécessaire. Les propriétés des tables d'iptables sont définies dans le fichier */usr/src/linux/include/linux/netfilter/ipv4.h*. En modifiant la valeur de la variable NF_IP_PRI_MANGLE (en la mettant à 200 par exemple), l'ordre souhaité est obtenu.

On ajoute alors la règle suivante :

```
iptables -t mangle -A POSTROUTING -s ip-fausse-machine -j TTL
                                  --ttl-set valeur-ttl
```

3.7 Traceroute

Les TTLs donnent des indications sur le nombre de sauts qui séparent les différentes machines. La commande traceroute, qui permet de connaître les différentes machines par lesquelles nos paquets vont transiter, va permettre de vérifier que le nombre de sauts est bien en accord avec le TTL obtenu. Pour chaque fausse machine simulée, une réponse différente à ce type de requête va devoir être renvoyées. Pour leurrer l'attaquant sur cet aspect plusieurs solutions sont possibles :

- countertrace : script Perl utilisant la chaîne QUEUE d'iptables qui permet a des programmes de traiter les différents paquets reçus
- mise en place d'honeyd simulant un faux réseau

La première solution est relativement facile à mettre en oeuvre même si certaines modifications sont nécessaires au niveau du script puisque celui-ci est prévu à la base pour simuler un faux traceroute sur une seule machine. En outre, il prend en compte le s principaux protocoles utilisés pour effectuer ce genre d'opération : icmp, udp et tcp. Ainsi, aussi bien un tracert sous un Windows, qu'un traceroute sous un Linux amèneront au même résultat. Il est également possible d'ajouter une latence aux paquets pour faire effectivement croire que plusieurs machines ont bien été traversées.

3.8 Serveur DNS

Les fausse IPs ne sont à priori renseignées dans aucun serveur DNS. L'hôte va donc offrir un service en tant que faux serveur DNS pour ces IPs, permettant ainsi de résoudre les noms associés.

3.9 Fingerprint

Des outils tels ettercap sous Linux permettent de deviner le système d'exploitation d'une machine à l'aide des différents paquets que celle -ci a émis. En outre, ces outils disposent d'une base de données permettant d'identifier le constructeur d'une carte réseau en fonction de l'adresse MAC de celle -ci. Il est possible de spécifier l'adresse MAC qui sera utilisé par le système UML lors du lancement de ce dernier en ajoutant sur la ligne de commande :

```
eth0=tuntap,,adresse-mac-systeme-uml,ip-systeme-uml
```

Le fingerprint ne laisse alors transparaître aucune information sur la virtualité du système.

3.10 Protection de l'hôte

Il ne faut pas perdre de vue que lorsque le pirate prend la main sur le système UML, il se trouve en réalité très proche de la machine hôte. Une bonne protection de celle -ci est alors indispensable. En l'occurrence des règles de firewalling très

poussée doivent être mises en place afin de pouvoir contrer les différentes attaques qui pourraient être menées depuis l'UML. En outre, il faut porter une attention particulière aux différents services que l'on souhaite lancer sur l'hôte et s'assurer qu'aucun bug connu n'existe sur ceux-ci.

4 Log et Traces

4.1 Introduction

Le but premier de faire du système UML un honeypot est de pouvoir découvrir de nouvelles attaques sans mettre en danger une véritable machine contenant des informations importantes.

Pour pouvoir découvrir les techniques qu'un attaquant va pouvoir mettre en oeuvre pour pénétrer le système, une gestion de logs va être mise en place. Ainsi, après l'attaque, le dépouillement de ceux-ci donneront une idée très précise quant à l'attaque menée.

4.2 Logs réseau

Les différents échanges entre l'attaquant et le système UML vont se faire au travers du réseau. En outre, l'attaquant va sans aucun doute tenter de se connecter à des machines extérieures afin de récupérer divers outils qui lui seront nécessaires pour continuer son attaque.

Il paraît donc intéressant d'enregistrer les différents paquets qui vont circuler. La façon la plus simple de procéder est de lancer un tcpdump sur la machine hôte. Il faut cependant faire en sorte de loguer l'intégral du trafic réseau. Pour cela, on tape la commande suivante :

```
tcpdump -i interface-hote-uml -s 1600 -w fichier-enregistrement
```

L'enregistrement du traffic réseau est également possible en utilisant iptables et plus précisémment la cible ULOG qui, outre le fait qu'elle prend moins de place que la cîble LOG, contient également plus d'informations.

4.3 Logs TTY

UML offre également une option intéressante et très utile dans la construction d'un pot à miel : le log tty. Il est en effet possible d'enregistrer tout ce qui est tapé dans une console dans un fichier afin de pouvoir le rejouer ultérieurement.

Cette opportunité n'est pas présente dans les binaires directement téléchargeables et il faut donc recompiler le noyau UML pour profiter de cette option.

Pour activer la prise en compte du log, il faut répondre yes à :

```
Enable tty logging (CONFIG_TTY_LOG) [N/y/?]
```

Les logs ainsi enregistrés sont stockés sur le disque dur de l'hôte et ne seront donc pas accessibles depuis l'UML. Si aucune option n'est précisée sur la ligne de commande lançant l'UML, chaque nouvelle session ouverte sera stockée directement dans un fichier. Toutefois celui-ci sera difficilement lisible car il contiendra à la fois ce qui aura été tapé et ce qui aura été affiché.

Il est toutefois possible d'enregistrer l'ensemble des logs sous un format réutilisable par la suite en ajoutant sur la ligne de lancement de l'UML :

```
tty_log_fd=numero_file_descriptor numero_file_descriptor>tty_log_file
(soit par exemple tty_log_fd=3 3>log-tty.txt)
```

L'ensemble des logs est alors enregistré dans un seul et unique fichier qui pourra ensuite être relu grâce à un script : playlog.pl.

Il est alors possible de rejouer les enregistrements en temps réels, permettant ainsi d'observer si les frappes ont été effectuées par un humain ou par un script.

4.4 Prelude

Les différents logs mis en place permettent de récupérer l'ensemble des informations souhaitées. Toutefois, il peut être intéressant d'ajouter un détecteur d'intrusion afin de faciliter le tri et d'attirer l'attention sur les principales attaques tentées. L'outil retenu pour effectuer cette opération est Prelude.

Prelude est un outil modulaire comprenant :

- prelude-manager : chargé de récupérer les logs des différents espions (sensors). Il joue un rôle capital et est donc placé sur l'host. De cette façon, un attaquant ne pourra pas y avoir accès
- prelude-nids : ce module est chargé de s'occuper des attaques provenant du réseau : scannage de ports, arp attack, ... Le trafic du système UML passant sur la carte réseau de l'hôte, cet espion est à installer sur l'hôte directement
- prelude-lml : ce module se charge de contrôler les actions des utilisateurs directement (login par ssh, login du root, ...). Cet espion est à installer sur le système UML puisque c'est lui qui doit être surveillé

4.5 Analyse d'une attaque

Que ce soit le trafic réseau, le log tty du système UML ou même les alertes fournies par Prelude, ces informations se trouvent toutes sur un système distant (l'hôte à priori) et ne seront donc pas modifiables par un attaquant qui aurait pris le contrôle sur le système UML.

En outre, après avoir analyser les alertes identifiées par Prelude, il est possible de rejouer d'une part le trafic réseau (avec la commande tcpdump -r fichier-enregistrement), ainsi que l'ensemble des logs tty (grâce au script playlog.pl).

Les nouvelles éventuelles attaques qui auront pu être mises en oeuvre seront donc enregistrées et pourront être étudiées.

5 Conclusion

Il est évident que faire d'un UML un honeypot n'est pas si simple, son comportement étant quelque peu différent d'un linux habitue l sur bien des points, la tâche n'est pas toujours simple pour masquer la virtualité de celui-ci. L'étude d'UML est intéressante puisque qu'elle permet de mieux connaître le système Linux, même si ça n'en est pas le but.

On peut donc au vu de cet article prétendre pouvoir surveiller plus facilement et avec moins de danger une attaque sur notre propre machine. Nous n'avons toutefois pas eu l'occasion de "mettre à disposition" notre système UML sur Internet, afin qu'il se fasse attaquer et qu'on puisse alors tester sa furtivité. Il faut cependant rester vigilent, car il reste de nombreux points qui pourraient trahir le système UML et il faut continuer de résoudre ces problèmes.

Une piste, que nous n'avons pas développé réellement, se situe autour de la création d'activité via des scripts Perl, un peu à la manière de ceux qui travaillent sur honeyd, mais du côté client cette fois-ci.

REFERENCE

http://user-mode-linux.sourceforge.net/
http://www.netfilter.org/
http://www.grsecurity.net/
http://www.prelude-ids.org/
http://michael.toren.net/code/countertrace/
http://www.honeynet.org/papers/uml/

Droit pénal et cybercriminalité : la répression des infractions liées aux TIC

Anne Cantéro

Cabinet Caprioli Avocats
9, avenue Henri Matisse,
F-06200 Nice
cabinet.caprioli-avocats@laposte.net

Résumé Le présent article reprend pour partie de façon actualisée l'article rédigé par Eric A. Caprioli, *Les moyens juridiques de lutte contre la cybercriminalité* et publié dans la Revue Risques, Les cahiers de l'assurance, juillet-septembre 2002, n° 51.

Les réseaux numériques abolissent les distances, les frontières, les territoires géographiques. Réduction de l'espace auquel s'ajoute l'accélération du temps. Les échanges à distance deviennent quasi-instantanés. Dans les entreprises ou les administrations, les données qui sont traitées par des moyens informatiques imposent des investissements de plus en plus importants. Ces données constituent le patrimoine informationnel de l'organisation, elles possèdent un caractère stratégique. L'information est source de pouvoir, sa maîtrise devient un enjeu majeur. On ne parle plus d'ordinateur ou de système informatique mais de système d'information.

Néanmoins, l'utilisation des réseaux tels l'internet présentent des risques[1] et des vulnérabilités inhérentes à leur nature ouverte et internationale. Le développement de l'internet dans le grand public s'est caractérisé par la médiatisation d'affaires frauduleuses liées directement ou non aux TIC (Technologies de l'Information et de la Communication). Il en a été ainsi par exemple de l'affaire Yahoo ! (Ordonnance de Référé du 20 novembre 2000, TGI Paris) où le juge des référés français a condamné une entreprise américaine, sur le fondement du droit français (article R-645-1 du code pénal), pour des faits (ventes d'objets nazis) commis virtuellement sur le territoire français, à partir du territoire américain.

Ce faisant, une brèche a été ouverte dans le principe de territorialité du droit pénal. De même, le 22 mars 2001, les experts informatiques de la police de New York interpellaient Abraham Abdallah, auteur présumé d'escroqueries de plusieurs dizaines de millions de dollars à des américains par le biais des réseaux en usurpant leur identité et en dupant ainsi les plus prestigieuses banques internationales. Une telle arrestation a été possible dans la mesure où ce délinquant informatique opérait depuis le territoire américain.

[1] La Commission européenne a établi une typologie des risques le 6 juin 2001 élaborée par la Communication de la Commission et intitulée *Sécurité des réseaux et de l'information - Proposition pour une approche politique européenne* (COM(2001) 298 ; voir sur le site `http://europa.eu.int`).

Si tel n'avait pas été le cas, comme par exemple l'auteur présumé du virus ILOVEYOU diffusé en mai 2000, il est fort probable qu'il surferait et sévirait encore. Plus récemment, des activistes se sont livrés à des activités de terrorisme sur les réseaux. En effet, à la suite des attentats du 11 septembre 2001, il a été annoncé que les terroristes d'Al-Quaida avaient eu recours aux systèmes de messageries électroniques associés à l'usage de moyens de cryptologie et de stéganographie pour assurer la confidentialité de leurs échanges tendant à la préparation et à l'organisation de leurs attentats.

L'énumération d'exemples d'activités criminelles liées aux technologies de l'information pourrait se multiplier. Ce, notamment car de telles activités peuvent prendre des formes très variées. Il peut ainsi s'agir d'atteintes aux systèmes d'information et/ou aux données informatisées, d'attaques de serveur par saturation (spamming), de violation du secret des correspondances privées, de violation des règles de protection des données personnelles, d'espionnage industriel ou militaire, de contrefaçon de droits de propriété intellectuelle (brevets, marques, dessins, droits d'auteur, ...), de délits de presse (ex : diffamation), de fraude fiscale, de fraude à la carte bancaire, de blanchiment d'argent, de réseaux de pédophiles, d'usurpation d'identité, d'organisation de la prostitution, ... En outre, la diffusion de certains virus (ex : Melissa, Iloveyou, Code Red) a causé d'importants dommages aux entreprises et autres organismes publics. Leurs modes de propagation sont divers : e-mail, Web, partage de réseau, etc.

Dans les systèmes d'information, et plus précisément l'internet, les intervenants sont multiples : opérateurs télécoms, fournisseurs de microprocesseurs et de matériels informatiques, SSII, fournisseurs d'accès, fournisseurs de contenu, hébergeurs, prestataires de services de certification (les tiers de confiance ou "Trusted third party", qui sont des fournisseurs de services à valeur ajoutée), ... Cet ensemble complexe est censé fournir les conditions techniques de la qualité des transmissions, des traitements et de la conservation des données.

L'ouverture sur le monde des réseaux qui ne connaissent pas les frontières politiques (étatiques) ou géographiques contribue à faire croître les risques de façon exponentielle ; les objectifs de sécurité des systèmes d'information deviennent quasiment illimités. Aujourd'hui, il ne suffit plus de garantir la fiabilité des traitements sur le ou les postes de travail de l'organisation en cause (par exemples par des anti-virus ou fire wall), ou dans le cadre du réseau interne, mais il convient d'être sûr que l'information à vocation interne ou qui circule sur des réseaux numériques ouverts, ne se perde pas, aille au bon destinataire, ne soit pas modifiée, reste confidentielle, etc. Ce d'autant plus que parfois, les informations qui circulent sont soumises au secret ; il en est ainsi pour le secret des affaires pour les entreprises et du secret professionnel auquel sont tenus certains professionnels tels les avocats, médecins et autres – cette obligation étant pénalement sanctionnée. En outre, le souci de sécurité concerne tant les réseaux ouverts que fermés.

Compte tenu des risques et des vulnérabilités induits par l'utilisation des réseaux, la question de la sécurité informatique est devenue fondamentale. Des réponses techniques y sont apportées. Mais le droit joue également un rôle impor-

tant pour répondre aux besoins exprimés. Ainsi, le droit, notamment en consacrant des principes tels les libertés individuelles, les droits de la personnalité ou en fournissant des outils contractuels qui sanctionnent les obligations non respectées, ou en encadrant l'utilisation de certains procédés techniques (dont la cryptologie), contribue à la sécurité sur les réseaux.

Plus précisément, en matière de criminalité informatique ou de criminalité informatisée (que l'on nomme ensemble usuellement "*cybercriminalité*"), dès lors que le recours à l'appareil répressif est décidé par la victime ou par le Ministère Public, une réalité classique s'impose : **le droit pénal est une des expressions de la souveraineté des Etats, en fonction de laquelle il possède une dimension territoriale**.

Compte tenu de la nature des TIC, le fait d'appréhender des comportements délictueux sur les réseaux se heurte à trois types de contraintes qui sont l'anonymat qui peut être organisé sur les réseaux, la volatilité des informations numériques (possibilité de modifier et de supprimer des éléments de preuve quasi instantanément), les comportements délictuels qui revêtent souvent un caractère transnational.

Face à ces réalités, au niveau national, le droit pénal a pris en compte ce type de délinquance en distinguant selon que "l'informatique" est victime ou moyen de commission d'infractions (section 1). De plus, une harmonisation internationale du droit et des procédures ainsi qu'une étroite coopération judiciaire se mettent en place pour être en mesure de lutter efficacement contre des cybercriminels qui tendent à s'organiser et à agir au niveau planétaire (section 2).

1 La répression pénale en droit français (principaux textes)

Deux catégories d'infractions sont à considérer distinctement : celles où l'informatique est l'objet même de l'infraction (section 1.1) et celles où l'informatique est le moyen de commission de l'infraction[2] (section 1.2).

1.1 La répression pénale des atteintes aux systèmes d'information

Les textes adoptés afin de protéger les systèmes d'information concernent différents domaines et peuvent prendre la forme d'obligations légales. Il en est ainsi de la protection des données à caractère personnel. Sur ce point, la Convention du Conseil de l'Europe du 28 janvier 1981, article 5, énonce le principe de la sécurisation des données qui pèse sur le responsable du traitement.

Ce principe se retrouve en tant qu'obligation à la charge du responsable du traitement (et de son sous-traitant) aux articles 16 et 17 de la directive

[2] Voir sur le sujet, E. A. Caprioli, *Les moyens juridiques de lutte contre la cybercriminalité*, Revue Risques, juillet-septembre 2002, n° 51 ; Ministère de la Justice - Direction des affaires criminelles et des grâces, *Le traitement judiciaire de la cybercriminalité - Guide méthodologique*, édité en mai 2002 (téléchargeable à partir du site `www.internet.gouv.fr`).

européenne du 24 octobre 1995 relative à la protection des données à caractère personnel[3] , ainsi qu'à l'article 29 de la loi informatique, fichiers et libertés du 6 janvier 1978 en France.

Le projet de transposition de la directive qui devrait être adopté courant 2003 en France reprend ces catégories d'obligations[4]. En cas de manquement à l'obligation de protection ainsi que dans l'hypothèse d'intrusion frauduleuse, des sanctions pénales sont prévues conformément aux articles 226-16 à 226-19 du Code pénal[5]. De même, les articles 226-13 du Code pénal relatif au secret professionnel ainsi que l'article 226-15 du Code pénal concernant le secret des correspondances et des communications peuvent trouver à s'appliquer selon les cas.

En conséquence, lorsqu'une entreprise détient des informations confidentielles communiquées par un partenaire commercial, elle a l'obligation de protéger la confidentialité des données transmises de la même façon que si elles lui appartenaient.

Par ailleurs, dès 1988, le législateur a prévu des dispositions afin de protéger les systèmes informatisés. Ainsi, la loi Godfrain du 5 janvier 1988 (modifiée depuis) a introduit les articles 323-1 à 323-7 dans le Code pénal. Ces articles traitent des infractions relatives aux atteintes aux systèmes de traitement automatisé de données des entreprises. Se pose néanmoins le problème d'identification des pirates et de la preuve de l'infraction notamment quant à l'établissement de son caractère frauduleux. L'affaire Kitetoa.com mérite à cet égard attention. Dans ce cas d'espèce, infirmant en cela le jugement du tribunal de grande instance de Paris du 13 février 2002, la Cour d'Appel de Paris a relaxé le 30 octobre 2002 le webmaster dudit site dans la mesure où celui-ci n'avait pas utilisé une méthode de piratage mais une manipulation accessible à tout internaute averti (non ingénieur) sans intention malveillante pour accéder à une base de données puisqu'il en avait informé le responsable afin d'attirer son attention sur l'absence de protection des données à caractère personnel traitées. Au demeurant, le projet de loi pour la confiance dans l'économie numérique prévoit d'alourdir

[3] La directive européenne du 24 octobre 1995 a été complétée par la directive du 12 juillet 2002 concernant le traitement des données à caractère personnel et la protection de la vie privée dans le secteur des communications électroniques (dite directive vie privée et communication électronique, J.O.C.E., 31 juillet 2002, L 201/37).

[4] Projet téléchargeable à partir du site de la C.N.I.L. (`www.cnil.fr`)

[5] Voir sur le sujet, A. Lucas, J. Devèze et J. Frayssinet, *Droit de l'informatique et de l'internet*, PUF, éd. Thémis, novembre 2001 ; J. Frayssinet, *Le projet de loi relatif à la protection des personnes physiques à l'égard des traitements de données à caractère personnel : constantes et nouveautés*, JCP éd. Communication - Commerce électronique, janvier 2002, chr. n° 1 ; J. Frayssinet, *Le transfert de la protection des données personnelles en provenance de l'Union européenne vers les Etats-Unis : l'accord dit "sphère de sécurité" (ou* safe harbour*)*, JCP éd. Communication V Commerce électronique, mars 2001, chr. n° 7 ; J. Huet, *Les contrats encadrant les transferts de données personnelles*, JCP éd. Communication - Commerce électronique, mai 2001, chr. n° 11.

les peines prévues aux articles 323-1 à 323-3 du Code pénal[6]. En outre, il est créé une nouvelle infraction avec l'introduction d'un article 323-3-1 dans le Code pénal. Cette disposition vise le fait de fournir le moyen (équipement, instrument, programme informatique ou toute donnée) conçu ou spécialement adapté pour commettre les faits réprimés aux articles 323-1 à 323-3 du Code pénal.

Il convient également de signaler les articles 441-1 et suivants du Code pénal relatifs aux infractions portant sur des faux, falsifications et usages de faux qui peuvent être appliquer à la cybercriminalité en cas par exemple d'usurpation d'identité. Enfin, les articles 410-1 et suivants du Code pénal traitent des atteintes aux intérêts fondamentaux de la nation (espionnage et autres sabotages). En conséquence, cette catégorie d'infractions bénéficie d'un régime de répression renforcé lorsque les systèmes d'information de l'État sont concernés par la cybercriminalité et remplissent les conditions posées.

1.2 La répression pénale lorsque l'informatique a servi de moyen de commission de l'infraction

Lorsque l'infraction "*classique*" a été commise au moyen de l'informatique, le régime de droit pénal commun s'applique mais les sanctions peuvent être renforcées lorsqu'un texte prévoit que le recours à un moyen informatique constitue une circonstance aggravante. A titre d'illustrations, il en est ainsi pour les atteintes sexuelles par un majeur sur un mineur de moins de 15 ans, "*lorsque le mineur a été mis en contact avec l'auteur des faits grâce à l'utilisation pour la diffusion de messages à destination d'un public non déterminé, d'un réseau de télécommunication*" (art. 227-26 du code pénal - peine d'emprisonnement maximale de dix ans et amende de 150 000 euros), pour les infractions à la loi sur la presse (loi du 29 juillet 1881) telles la provocation à la discrimination, à la haine ou à la violence raciale qui sont sanctionnées par une peine d'emprisonnement maximale de un an et une amende de 45 000 euros.

Par ailleurs, la loi n° 2001-1062 du 15 novembre 2001 sur la sécurité quotidienne (dite L.S.Q.)[7] ainsi que le projet de loi pour la confiance dans l'économie numérique[8] constituent les apports les plus significatifs quant au régime juridique répressif applicable lorsque l'informatique a servi à commettre l'infraction établie.

Les articles 29 et suivants de la loi du 15 novembre 2001 prévoient **pour la conservation des données**, diverses obligations qui conduisent à lutter contre

[6] Ce projet succède à celui de la loi sur la société de l'information et à celui de la loi pour l'économie numérique. Le texte adopté à l'Assemblée nationale le 26 février 2003 est téléchargeable à l'adresse `http://www.assemblee-nationale.fr/12/ta/ta0089.asp`. Cf. sur le point qui nous intéresse l'article 33 du projet.

[7] J.O. du 16 novembre 2001. Voir pour un commentaire, L.Costes, *Le renforcement du contrôle et de la sécurité sur les réseaux par la loi relative à la sécurité quotidienne*, Lamy droit de l'informatique et des réseaux, Bull. G, n° 141, novembre 2001, p. 1 et suiv.

[8] Réf. citées supra.

la cybercriminalité. Ainsi, il est fait obligation aux fournisseurs d'accès Internet de conserver les traces de connexions de leurs clients pendant un an "*pour les besoins de la recherche, de la constatation et de la poursuite des infractions pénales, et dans le seul but de permettre, en tant que de besoin, la mise à disposition de l'autorité judiciaire d'informations*". Cette obligation est une exception au principe d'effacement ou d'anonymat immédiat après la fin de la communication qui est posé à l'article L. 32-3-1 du CPT (et qui est sanctionné par une peine d'emprisonnement maximale de un an et une peine d'amende de 75 000 euros). Une autre exception est celle tenant à la facturation. De plus, l'article 20 de la loi n° 2003-239 du 18 mars 2003 pour la sécurité intérieure permet aux opérateurs de conserver certaines données pour assurer la sécurité de leurs réseaux. L'application de cette nouvelle disposition est soumise à l'adoption d'un décret.

Cette obligation de conservation des données de connexion ne porte que sur l'identification du poste émetteur et du poste récepteur, sur la durée de connexion voire ses conditions techniques et en aucun cas sur le contenu du message (en la matière ce sont exclusivement les dispositions de la loi du 10 juillet 1991 relative au secret des correspondances par voie de télécommunication et prévoyant les conditions des interceptions téléphoniques qui s'appliquent - voir les articles 100 à 100-7 du code de procédure pénale qui restent inchangés). Cependant, les modalités de conservation devaient être précisées par décret. En outre, la loi de finances rectificative pour 2001 du 28 décembre 2001 a élargi l'accès aux données de connexion auprès des fournisseurs d'accès et des opérateurs de télécommunications, alors que la loi sur la sécurité quotidienne réservait cette possibilité aux seuls juges, aux agents des douanes, du fisc et aux enquêteurs de la Commission des Opérations de Bourse[9].

Au demeurant, la France était déjà intervenue en ce domaine avec la loi du 1er août 2000 (réformant la loi du 30 septembre 1986 sur la liberté de communication) qui encadre la responsabilité et les obligations des fournisseurs d'accès à l'internet (F.A.I.) et des fournisseurs d'hébergement. Elle leur impose d'identifier leurs clients bénéficiant de services d'accès à l'Internet et leur donne obligation de conserver et de communiquer ces données identifiantes sur réquisition de l'autorité judiciaire. Toutefois, elle ne précisait pas combien de temps les F.A.I. avaient l'obligation de conserver ces données, ni ce qu'il convient d'entendre par autorité judiciaire. Par ailleurs, les personnes physiques ou morales dont l'activité est d'éditer un service de communication en ligne, autre que des correspondances privées, doivent s'identifier précisément et tenir ces informations à la disposition du public.

Le projet de loi pour la confiance dans l'économie numérique tel qu'adopté par l'Assemblée nationale le 26 février 2003 devrait modifier la loi n° 86-1067 du 30 septembre 1986 relative à la liberté de communication par un chapitre VI intitulé "*Dispositions relatives aux services de communication publique en ligne*"

[9] Cela figure à l'article 62 de la LFR n°2001-1276, adoptée le 28 décembre 2001, J.O. du 29 décembre 2001, p.21133 ; article 65 du code des douanes ; article L. 83 du Livre des Procédures Fiscales ; article L. 621-10 Code monétaire et financier.

ainsi que le code des postes et télécommunications (articles L. 32-3-3 et L. 32-3-4). Aux termes des dispositions projetées, les responsabilités civiles et pénales ainsi que les obligations de conservation des données d'identification pesant sur les fournisseurs d'accès internet et les hébergeurs de site sont déterminées. Si le futur article 43-11 de la loi du 30 septembre 1986 exclut une obligation générale de surveillance des informations transmises ou stockées par ces prestataires, en revanche, il leur incombe d'agir avec promptitude pour faire cesser tout trouble illicite dont ils auraient été informés.

Au demeurant, en application du futur article 43-12 de la loi du 30 septembre 1986, il est prévu que l'autorité judiciaire est compétente pour prescrire en référé toutes mesures propres à faire cesser un dommage occasionné par le contenu d'un service de communication publique en ligne. En ce qui concerne plus précisément les données d'identification, le législateur entend imposer aux F.A.I. et aux hébergeurs "*de vérifier, de détenir et de conserver les données de nature à permettre l'identification de quiconque a contribué à la création du contenu ou de l'un des contenus des services dont elles sont prestataires*". Ces prestataires sont soumis aux articles 226-17, 226-21 et 226-22 du code pénal relatifs à la protection et au respect des données à caractère personnel traitées. Un décret doit intervenir pour définir les données concernées et fixer la durée et les modalités de leur conservation ; étant noté que l'autorité judiciaire peut requérir la communication de ces données auprès desdits prestataires. En cas de manquement à ces obligations, une peine d'amende de 3 750 euros est applicable (futur article 79-7 de la loi du 30 septembre 1986 modifiée) ; la responsabilité des personnes morales est prévue.

La conservation de ces données a pour objectif de permettre l'identification des délinquants et de matérialiser des éléments de preuve des infractions. Néanmoins, ces mesures peuvent s'avérer insuffisantes si l'Etat en cause ne se dote pas des moyens juridiques appropriés pour procéder à **des interceptions légales et aux déchiffrements des messages codés suspects**. En France, l'autorité judiciaire a déjà la possibilité de procéder par voie de réquisition auprès des opérateurs téléphoniques pour accéder au contenu des messages, ces prestataires ayant l'obligation de fournir les données de connexion (la prestation est facturée environ 90 euros).

En ce qui concerne le contenu des communications, c'est le régime classique sur les écoutes téléphoniques (procédure judiciaire) et les interceptions de sécurité (procédure administrative) qui a vocation à s'appliquer, sans qu' il y ait besoin de profonde modification (loi du 10 juillet 1991). De plus, la L.S.Q. a introduit un nouvel article 11-1 dans la loi du 10 juillet 1991, aux termes duquel les autorités judiciaires peuvent avoir accès aux conventions de chiffrement des données chiffrées du présumé délinquant par l'entremise des prestataires de services de cryptologie[10]. En outre, d'une part, la L.S.Q. a intégré plusieurs articles dans le code de procédure pénale afin de permettre aux magistrats (du parquet,

[10] En cas de non remise ou de non mise en œuvre des conventions secrètes, les personnes sont passibles de deux ans d'emprisonnement et de 30.000 euros d'amende.

de l'instruction et de la juridiction de jugement) d'ordonner le déchiffrement des données.

D'autre part, elle a établi une nouvelle incrimination avec l'introduction de l'article 434-15-2 dans le code pénal : toute personne qui a connaissance de la convention secrète de déchiffrement d'un moyen de cryptologie susceptible d'avoir été utilisé pour préparer, faciliter ou commettre un crime ou un délit, a l'obligation de remettre ladite convention aux autorités ou de la mettre en œuvre.

En cas de défaut d'exécution, des peines de prisons et d'amende sont prévues[11]. Ces dispositions sont reprises et confirmées de façon quasi identique dans le projet de loi pour la confiance dans l'économie numérique[12] ; étant relevé que le code de procédure pénale fait l'objet de modifications complémentaires afin d'intégrer les données informatiques[13] . Enfin, l'article 18 de la loi du 18 mars 2003 pour la sécurité intérieure a modifié l'article 60-1 du code de procédure pénale.

Désormais, les officiers de police judiciaire après autorisation exprès des magistrats compétents selon la procédure concernée, peuvent demander aux organismes publics ou personnes morales de droit privé, sauf exception légale, la mise à disposition des informations utiles à la manifestation de la vérité, hormis celles légalement protégées par le secret, lorsque ces informations sont contenues dans le ou les systèmes informatiques ou traitement de données nominatives qu'ils administrent. De plus, ils peuvent obliger les opérateurs de télécommunication à prendre toutes mesures propres à assurer la préservation du contenu des informations consultées par les personnes utilisatrices de leurs services.

Ils ne peuvent néanmoins intervenir en la matière que sur réquisition du procureur de la République préalablement autorisé par ordonnance du juge des libertés et de la détention et ces mesures de préservation doivent être imposées pour une durée maximale d'un an. En cas de manquement à ces obligations légales sans motif légitime, les personnes concernées encourent une peine de 3 750 euros. La responsabilité pénale des personnes morales est prévue. Un décret doit être adopté afin de déterminer les organismes concernés ainsi que les modalités de transmission des données et leur nature.

Le dispositif répressif adopté en France se trouve renforcé par la mobilisation internationale et communautaire qui s'organise afin de lutter efficacement contre la cybercriminalité.

---

[11] Article 434-15-2, alinéa 1 : 3 ans d'emprisonnement et 45.000 euros d'amende et Article 434-15-2, alinéa 2 : 5 ans d'emprisonnement et 75.000 euros d'amende.

[12] Si les articles 30 et 31 de la L.S.Q. sont en effet abrogés par le projet, il n'en demeure pas moins que le contenu desdites dispositions est repris aux articles 26 et 27 de ce projet.

[13] Cf. articles 30, 31 et 32 du projet de loi.

2 L'organisation de la répression pénale aux niveau international et communautaire

Deux textes majeurs tendent à harmoniser la répression au niveau international et communautaire. Il s'agit de la convention du Conseil de l'Europe sur la cybercriminalité du 23 novembre 2001 (section 2.1) et de la proposition de décision-cadre de la Commission européenne relative aux attaques visant les systèmes d'information du 19 avril 2002 (section 2.2).

2.1 La convention du Conseil de l'Europe sur la cybercriminalité (23 novembre 2001)

L'harmonisation en matière de cybercriminalité est devenue une priorité majeure des Etats qui sont entrés dans la société de l'information, spécialement les Etats membres du Conseil de l'Europe, conformément aux bases qui ont été définies par le G8. C'est dans cette perspective que la Convention du Conseil de l'Europe sur la cybercriminalité a été signée le 23 novembre 2001 à Budapest par 33 Etats, dont les membres de l'Union européenne, les Etats-Unis d'Amérique, le Canada, le Japon et l'Afrique du Sud[14] . Trois axes ont été dégagés.

D'une part, en ce qui concerne l'accès aux contenus des messages, c'est à dire les interceptions de sécurité et déchiffrement des données codées, les Etats ont l'obligation d'adopter les mesures nécessaires afin de pouvoir enjoindre à une personne ou à une entreprise de conserver certaines données informatiques stockées ou des données de connexion relatives à une infraction pénale, sous le sceau du secret procédural, notamment lorsque ces données risquent de disparaître ou d'être modifiées et de pouvoir les divulguer à l'autorité compétente de l'Etat partie. De la même manière, les Etats doivent habiliter leurs autorités compétentes pour qu'elles puissent légalement enjoindre aux personnes présentes sur leur territoire et aux fournisseurs de services internet à communiquer les informations en leur possession.

D'autre part, compte tenu de l'objectif d'harmonisation des infractions, les Etats parties à la Convention s'engagent à adopter en conformité avec leur droit interne des législations qui définissent un certain nombre d'infractions ainsi que leur tentative de commission, tout en laissant aux Etats une marge d'adaptation plus ou moins large selon les faits visés. En droit français, ces incriminations reprennent globalement la substance de la loi Godfrain du 5 janvier 1988 (articles 323-1 à 323-7 du code pénal). Les Etats doivent prendre les mesures nécessaires pour que les infractions visées soient sanctionnées par des "*sanctions effectives, proportionnées et dissuasives y compris la privation de liberté*". Ils doivent adopter des législations incriminant les infractions concernant la confidentialité, l'intégrité et la disponibilité des données et des systèmes informatiques, les falsifications et fraudes informatiques, la pornographie enfantine, les infractions liées aux atteintes à la propriété intellectuelle et aux droits connexes. La responsabilité pénale des personnes morales est prévue.

[14] Le texte est consultable sur le site http ://conventions.coe.int.

Enfin, les Etats parties à la Convention ont l'obligation d'habiliter leurs administrations respectives à perquisitionner les systèmes informatiques, à saisir des données et à imposer aux personnes concernées de fournir les données en leur possession, de conserver les données "*vulnérables*" ou de les faire conserver par les personnes concernées. De plus, les autorités compétentes d'un Etat doivent pouvoir perquisitionner et saisir les informations relatives à une infraction liée à l'utilisation de l'informatique. La question des perquisitions à distance et transfrontalières dans des systèmes informatiques implantés dans un Etat partie, à partir d'un autre Etat a été abordée, mais aucune position commune n'a pu être adoptée, ni aucune mesure technique envisagée concrètement. En définitive, pour mettre en œuvre cet arsenal pénal international, la Convention pose un principe général de coopération. Mais, si cette Convention présente l'avantage d'énoncer les mesures que les Etats doivent adopter, elle n'en précise pas le contenu. Ceci explique pourquoi, concernant l'harmonisation des sanctions relatives aux attaques visant les systèmes d'information, la Commission européenne a élaboré une proposition de décision cadre le 19 avril 2002.

2.2 La proposition de décision-cadre du 19 avril 2002 de la Commission européenne relative aux attaques visant les systèmes d'information

Cette proposition de décision-cadre a été établie suite à de nombreuses réunions et initiatives européennes qui avaient pour préoccupation générale d'assurer une plus grande sécurité sur les réseaux[15] . Elle vise à rapprocher les dispositions matérielles des droits pénaux des Etats membres par l'institution d'incriminations communes en matière d'attaques contre les systèmes d'information et tend à optimiser la coopération policière et judiciaire pour les infractions pénales liées aux systèmes d'information. La proposition de décision-cadre prévoit ainsi des : "*règles minimales relatives aux éléments constitutifs des infractions pénales*", et ce, notamment en ce qui concerne la criminalité organisée et le terrorisme. Dans cette logique, la proposition (articles 3 et 4) détermine les critères qui doivent être remplis pour qu'un acte soit criminalisé (c'est à dire juridiquement qualifié de crime et non plus de délit). Ce faisant, il est toujours nécessaire que les infractions criminalisées (couvertes par la proposition) aient été commises intentionnellement. La proposition de décision-cadre tend également à "*assurer la compatibilité des règles applicables dans les Etats membres*" en vue d'assurer et d'accélérer la coopération entre les autorités judiciaires. Le but est qu'il n'y ait pas de législation plus clémente qu'une autre, voire une opposition à la répression compte tenu d'un droit applicable par un Etat membre plus souple au regard des infractions commises et criminalisées aux termes de la proposition de décision-cadre.

[15] Pour la dernière version du texte, voir COM(2002)173, J.O.C.E., 27 août 2002, C203E/109 ; pour la doctrine, L. Costes, *Une proposition de décision-cadre de la Commission visant à renforcer la sécurité des réseaux et des systèmes d'information*, Lamy droit de l'informatique et des réseaux, bulletin d'actualité B, n° 147, mai 2002.

Cette proposition s'applique aux Etats membres cibles des infractions mais également lorsque les actes constitutifs d'une infraction ont été perpétrés sur le territoire de l'Union européenne à l'encontre de systèmes d'information situés sur le territoire de pays tiers. Cette portée confirme la volonté de prendre en compte et de lutter contre la cybercriminalité non seulement en Europe mais également au niveau mondial. Cette proposition de décision-cadre n'empiète cependant pas sur les protections et règles dégagées dans le cadre de directives européennes (comme par exemple la protection des données à caractère personnel avec la directive du 24 juin 1995).

Deux infractions pénales sont définies afin de lutter contre la cybercriminalité. Ainsi, aux termes de l'article 3, l'infraction d'*accès illicite à des systèmes d'information* est établie si elle a été commise "*contre toute ou partie d'un système d'information faisant l'objet de mesures particulières ; ou avec l'intention de porter préjudice à une personne physique ou morale ; ou avec l'intention d'obtenir un avantage économique*". En définissant de la sorte cette infraction, la commission prend donc en compte la notion de piratage. Cette infraction ne remet pas en cause les obligations de protection qui incombent aux personnes titulaires ou utilisatrices de systèmes d'information.

Est également créée l'infraction d'interférence illicite avec des systèmes d'information qui couvre : "*le fait de perturber gravement le fonctionnement d'un système d'information ou de l'interrompre, en introduisant, transmettant, endommageant, effaçant, détériorant, modifiant ou supprimant des données informatiques*" lorsque ces faits ont été commis intentionnellement et sans droit. Cette effraction couvre également le fait "em d'effacer, de détériorer, d'altérer de supprimer ou de rendre inaccessibles des données informatiques dans un système d'information lorsque l'acte est commis avec l'intention de porter préjudice à une personne physique ou morale." L'incitation, l'aide ou la complicité volontaire à commettre l'une de ces infractions sont également punissables. Il en est de même des tentatives de les commettre.

Selon l'article 6 de cette proposition de décision cadre, ces infractions sont passibles de "*sanctions effectives, proportionnées et dissuasives*" ; ce qui reprend la formulation de la Convention du Conseil de l'Europe sur le cybercriminalité. Mais la proposition de décision-cadre est plus précise puisqu'elle prévoit que les infractions déterminées doivent être punies d'une peine maximale qui ne peut être inférieure à un an. Les Etats membres ont également "*la possibilité d'infliger des amendes en plus des peines privatives de liberté ou comme alternatives à ces dernières*". Ces sanctions peuvent être alourdies lorsque ces infractions sont accompagnées de "*circonstances aggravantes*" en application de l'article 7 de la proposition de décision-cadre) et ce, afin de lutter contre le terrorisme et le grand banditisme. Sont considérées comme circonstances aggravantes, les actes qui ont été commis "*dans le cadre d'une organisation criminelle*", ceux qui ont provoqué "*une perte économique importante, (...) des dommages corporels à une personne physique ou un dommage important à une partie des infrastructures critiques de l'Etat membre*" ou ceux qui ont entraîné "*des profits importants*". Dans ces situations, la durée maximale de la peine d'emprisonnement ne peut

être inférieure à quatre ans. Il convient de relever que la responsabilité pénale des personnes morales est prévue.

Enfin, l'article 11 traite de la coopération judiciaire et de la compétence reconnue aux autorités judiciaires nationales. Pour faciliter la coopération entre les autorités judiciaires des Etats membres et la coordination de leurs actions, la Commission rappelle notamment que les Etats membres peuvent avoir recours à tout organe ou mécanisme établi au sein de l'Union européenne (à savoir Eurojust et le réseau judiciaire européen). Les Etats membres ont jusqu'au 31 décembre 2003 pour adopter en droit national ces dispositions et s'y conformer.

Détection des systèmes d'exploitation avec RINGv2

Olivier Courtay, Olivier Heen, and Franck Veysset

olivier.courtay@enst-bretagne.fr,
École Nationale Supérieure des Télécom.
olivier.heen@thomson.net,
Thomson Recherche et Innovation
franck.veysset@francetelecom.com,
France Télécom Recherche et Développement

Résumé Après un survol du domaine de la reconnaissance à distance des systèmes d'exploitation, nous rappelons le principe de la reconnaissance temporelle mise en œuvre dans l'outil RING. Nous présentons ensuite une évolution, RINGv2, et indiquons divers exemples de fonctionnement.

Mots-clé *OSFP*, Congestion *TCP/IP*, *RTT*, *RTO*.

Introduction

Détecter à distance les systèmes d'exploitation de machines en réseau peut s'avérer utile pour :

- Qualifier un parc, y détecter des systèmes inhabituels.
- Conduire des audits de sécurité, éventuellement automatisés.
- Préparer des attaques spécifiques de certains systèmes d'exploitation.
- Alimenter des bases de données à des fins statistiques.
- Mieux connaître la source d'une attaque en cours.
- Abaisser la priorité d'une alerte IDS lorsque le système d'exploitation n'est, en fait, pas sensible à l'attaque en cours [6].
- Concevoir des leurres simulant précisément le comportement de systèmes d'exploitation spécifiques.

Cet article est essentiellement consacré à la présentation de RINGv2. La première section situe le domaine de la détection à distance des systèmes d'exploitation[1] et en indique les principales difficultés. La deuxième section décrit les principes de l'analyse temporelle tels qu'ils sont mis en œuvre dans RING et RINGv2. La troisième section montre des cas concrets d'utilisation des deux outils. Quelques pistes pour les évolutions de RINGv2 sont données en conclusion.

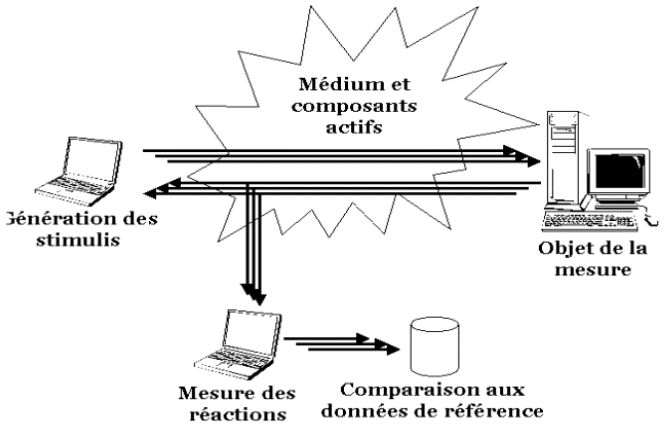


Fig. 1. Principe général de l'*OSFP*. Lorsqu'une même entité génère les stimuli et mesure les réactions on parle d'*OSFP* actif.

1 Détection des systèmes d'exploitation

1.1 Principes

Les systèmes d'exploitation sont reconnus au travers de leurs propriétés caractéristiques. Dans le cas de la détection à distance, les propriétés en question sont liées à des comportements réseau (cf. figure 1).

Les spécifications de protocoles comme ICMP, UDP ou TCP laissent une grande liberté d'implémentation. Chaque éditeur interprète, optimise et paramètre : autant d'indices qui vont ultérieurement *signer* la pile. Les caractéristiques les plus révélatrices d'une pile IP particulière se nichent préférentiellement dans des portions de code :

- Rarement exécutées comme pour la gestion d'erreurs ou la prise en charge de cas non conformes aux RFC.
- Complexes comme pour la gestion de congestion.
- Très optimisées comme pour l'assemblage de paquets fragmentés et la génération aléatoire de numéros de séquence.
- Mal programmées, sans contrôle de la taille des mémoires tampon.

Un grand nombre de techniques ont vu le jour, reprenant toutes les grandes lignes indiquées figure 2. L'état d'avancement des techniques varie de la simple publication pour l'analyse de la répartition des numéros de séquence [11], au logiciel éprouvé et maintenu dans le cas de `NMAP` [4], en passant par le démonstrateur pour `TBIT` [7], `RING` [9], `RINGv2` [10] et le démonstrateur avancé pour `XPROBE2` [1].

[1] Par la suite, la notation *OSFP* pour *Operating System FingerPrinting* est utilisée.

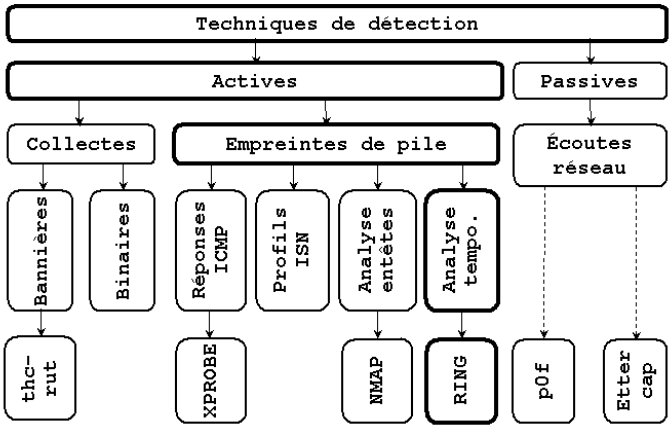


Fig. 2. Arbre des techniques d'*OSFP* étiqueté par quelques outils.

1.2 Difficultés

Dans le cas particulier de la détection active, le schéma de principe laisse présager de nombreuses difficultés :

- Tous les stimuli envoyés sont-ils bien reçus et traités par l'objet de la mesure ?
- L'ordre des stimuli à-t-il une incidence sur l'objet de la mesure ? Sur les composants intermédiaires ?
- Toutes les réactions sont-elles correctement perçues et mesurées ?
- Les données de référence sont-elles exactes et en nombre suffisant ?
- Quel sont les effets du médium et des composants intermédiaires ?

Sensibilité aux conditions de test

La qualité d'une détection peut varier grandement selon qu'elle a lieu en environnement LAN (filaire ou non), ou WAN en présence de routeurs, de pare-feu, voire d'IDS actifs. La charge réseau peut également influencer les mesures. le cas de `RING`.

L'outil `NMAP`, par exemple, peut être très perturbé par la présence d'un pare-feu. En effet il effectue quelques tests sur des ports fermés, ce sont les tests `TCP Xmas (F/P/U)`, `TCP SYN`, `TCP ACK` et `UDP` avec réponse ICMP. Les résultats ne sont pas les mêmes selon que le pare-feu met en œuvre des règles de type `reject` ou bien `drop` (cf. table 1).

Perturbations induites

Certains stimuli peuvent placer la pile dans une situation d'erreur. Ce dernier point est bien connu et utilisé pour des attaques de type déni de service. Pour mémoire, voici quelques exemples historiques :

```
Agressive OS guesses : Win2000, WinXP
OS Fingerprint :
TSeq(Class=RI%gcd=1%SI=29D09%IPID=RD...
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags...
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%O...
T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags...
T4(Resp=Y%DF=N%W=0%ACK=O%Flags...
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%
RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

```
Aggressive OS guesses : FreeBSD2.2.1-4.1
(90%),Windows Me,Win2000 or XP (86%), ...
TCP/IP fingerprint :
TSeq(Class=RI%gcd=1%SI=6F65%IPID=I%TS=U)
T1(Resp=Y%DF=Y%W=FFFF%ACK=S++%Flags...
T2(Resp=N)
T3(Resp=Y%DF=Y%W=FFFF%ACK=S++%Flags...
T4(Resp=Y%DF=N%W=0%ACK=O%Flags...
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

Tab. 1. Effet d'un pare-feu sur la détection d'un Microsoft Windows 2000 SP1 avec NMAP (`# nmap -O --osscan-guess -n -P0 -p 139,1234 x.x.x.x`). À gauche le résultat correct sans pare-feu, à droite le résultat erroné en présence d'un pare-feu où les ports fermés sont protégés par des règles `drop`.

- Le *Ping of Death* et ses nombreuses variantes où la pile ne contrôle pas la taille du tampon pour les paquets ICMP `echo-request`.
- Certaines attaques comme `WinNuke` ou `Teardrop`, où la pile gère mal une conjonction, absurde au sens des RFC, du drapeau `URG` et du pointeur `OOB` dans un même paquet `SYN`.
- Certains pare-feu avec tables d'état, très perturbés par des paquets `NMAP` sortant du réseau interne.

Une méthode primaire et terriblement "bruyante" d'*OSFP* consisterait à envoyer successivement des tentatives de blocage spécifiques de chaque système d'exploitation. Si le détecteur est capable de vérifier l'état de disponibilité de la cible après chaque tentative, il peut obtenir une mesure du système d'exploitation.

A contrario, les méthodes qui respectent les RFC induisent moins de perturbation sur les cibles et les composants intermédiaires. Elles seront donc préférées par des auditeurs, des administrateurs ou même des systèmes de détection automatique. Lorsque la détection est utilisée comme prélude à une attaque, cette discretion peut s'avérer ennuyeuse pour les pare-feu et les IDS chargés de bloquer ou de repérer les signes avant coureurs (cf. 2.3 pour le cas de `RINGv2`).

Apprentissage

Une fois correctement induites et mesurées, les réactions caractéristiques doivent encore être comparées à une base de référence. Plusieurs cas peuvent se produire, parmi lesquels quelques cas limites :

- Le système n'est pas dans la base, mais l'algorithme de correspondance confond avec un système proche. C'est souvent le cas avec des systèmes issus de souches BSD (très utilisées).
- Le système est dans la base mais l'algorithme de correspondance ne parvient pas à établir l'identité. C'est souvent le cas lorsque les caractéristiques sont trop peu discriminantes ou lorsque les résultats de mesure sont incomplèts.

L'obtention de valeurs de référence pose des problèmes de confiance dans les sources d'information, dans les conditions de mesure au moment de la prise de référence (elles ne sont pas toujours connues) et de validité dans un environnement réel : certaines références correctes en LAN non filtré sont inutilisables en WAN.

*En résumé : une bonne méthode d'*OSFP *suppose une bonne base de connaissances, un jeu de tests discriminants et correctement ordonnés, un minimum de robustesse par rapport aux fluctuations du réseau et un certain respect des RFC pour minimiser les perturbations et être efficace.*

2 Principes de `RING` et de `RINGv2`

2.1 Principe de base

Le protocole *TCP/IP* est *full duplex* et connecté, en ce sens qu'il nécessite un accord préalable entre les machines désirant communiquer (cf. figure 3). Il assure également la distribution ordonnée et sans erreur des paquets dès la première étape de la connexion. Pour ce faire, *TCP/IP* met en place des mécanismes sophistiqués de retransmission des paquets perdus, d'acquittement, de gestion de congestion, d'adaptation des tailles de fenêtres, etc.

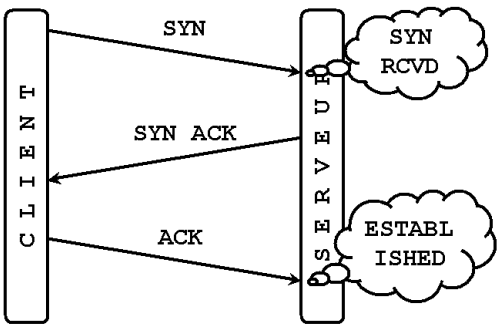


Fig. 3. Établissement d'une connexion *TCP/IP* (*3-way handshake*) et évolution des états internes du serveur.

La gestion de congestion

Lorsqu'un paquet *TCP/IP* est perdu, le protocole réagit aussitôt comme s'il y avait congestion du réseau : il ralentit le débit, tout en provoquant la retransmission des paquets perdus afin de maintenir l'intégrité du flux. Pour ne pas entretenir ou aggraver la congestion, les paquets doivent être retransmis avec une cadence de plus en plus lente tant que la communication n'est pas rétablie.

Le protocole *TCP/IP* doit maîtriser deux valeurs importantes :

- Le temps moyen d'aller/retour d'un paquet appelé RTT(*Roud Trip Time*).
- La temporisation associée à chaque segment non acquitté, appelée RTO (*Retransmission Time Out*). La valeur du RTO est dépendante de celle du RTT.

Pour approximer ces valeurs, les implémentations ont rapidement utilisé une moyenne lissée. Plus précisément, la RFC 793 *Transmission Control Protocol* suggère les méthodes de calcul suivantes :

```
Measure the elapsed time between sending a data octet with a particular
sequence number and receiving an acknowledgment that covers that sequence
number (segments sent do not have to match segments received).
This measured elapsed time is the Round Trip Time (RTT).  Next compute a
Smoothed Round Trip Time (SRTT) as:
                  SRTT = ( ALPHA * SRTT ) + ((1-ALPHA) * RTT)
and based on this, compute the retransmission timeout (RTO) as:
                   RTO = min[UBOUND,max[LBOUND,(BETA*SRTT)]]
where UBOUND is an upper bound on the timeout (e.g., 1 minute), LBOUND is
a lower bound on the timeout (e.g., 1 second), ALPHA is a smoothing factor
(e.g., .8 to .9), and BETA is a delay variance factor (e.g., 1.3 to 2.0).
```

P. Karn [5] a proposé quelques heuristiques maintenant adoptées dans les implémentations courantes de *TCP/IP* :

- En cas de retransmission de paquet, le RTT n'est pas mesuré (pour éviter toute ambiguïté sur la mesure).
- Le RTO est alors doublé $RTO =$ `old_RTO` * 2.
- Si `old_RTO` n'est pas connu (lors du démarrage), alors RTO vaut 6 secondes.

N.B. : croire à la congestion dès qu'un paquet est perdu n'est pas toujours une bonne chose. Dans les réseaux sans fil la perte d'un paquet n'est pas obligatoirement synonyme de congestion. Il se peut également que l'une des deux machines, le serveur par exemple, soit persuadé qu'il y a congestion, alors qu'en fait le client ignore délibérément les messages qu'il reçoit (cf. 2.1).

Fonctionnement de `RING`

Pour tenter une reconnaissance à distance, `RING` nécessite un (seul) port *TCP/IP* ouvert. Il crée délibérément une situation que le serveur interprète comme une congestion. Ce dernier retransmet alors des paquets, conformément à son algorithme de gestion de congestion et à ses valeurs de paramètres. La machine de mesure calcule la suite des délais de retransmission, puis la confronte à une base de référence pour déduire les caractéristiques de la pile *TCP/IP* du serveur. Une fois ces caractéristiques connues, il est souvent possible d'en déduire

le système d'exploitation, sa version, et dans certains cas la sous-version ou le niveau de *patch*.

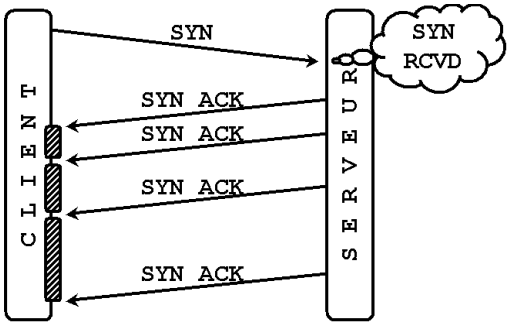


Fig. 4. Schéma de principe de `RING`.

Pour la première version de `RING`, la congestion est simulée dès la première étape du *3-way handshake* (cf. figure 4), en ignorant au niveau du client tous les paquets `SYN ACK` renvoyés par le serveur. Dans ce cas particulier, à la fois la RFC 2988 et la pratique montrent qu'un même serveur réagit toujours de la même manière.

2.2 Extensions conduisant à `RINGv2`

Dès qu'un protocole spécifie des délais entre des évènements déclenchés par l'objet de la mesure, et dès que ces délais varient d'une implémentation à l'autre, le principe utilisé dans `RING` peut être appliqué. Cette généralisation est très vaste et peut même concerner des protocoles comme SSL ou le fonctionnement d'applicatifs, ce qui peut conduire à de la reconnaissance de services. Dans le cas de *TCP/IP*, l'état `SYN RCVD` n'est pas le seul pour lequel le serveur doit renvoyer une suite de paquets avec délais.

Autres états analysables

Lors de la fermeture d'une connexion, plusieurs phénomènes peuvent se produire. La figure 5 montre le dialogue lorsque le client décide de clore une connexion. Il émet tout d'abord un paquet `FIN`. Le serveur passe alors dans l'état `CLOSE WAIT`, accuse réception du paquet en émettant un `ACK`, puis ferme sa demi-connexion en envoyant un `FIN` au client. Si ce `FIN` n'arrive pas, le serveur croit à une congestion réseau et retransmet des `FIN` avec des intervalles d'attente de plus en plus longs pour ne pas aggraver la congestion. Les temps d'attente peuvent être mesurés et, puisqu'ils sont fortement dépendants des

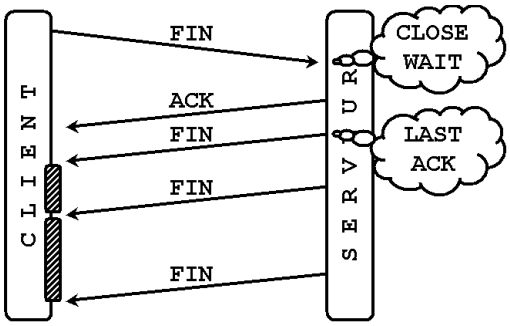


Fig. 5. Terminaison à l'initiative du client, avec congestion simulée.

implémentations, comparés à une base de référence pour tenter une détection du système d'exploitation.

La figure 6 montre un autre cas de terminaison, plus difficile à obtenir, où c'est le serveur qui décide de terminer la connexion. Ce cas se produit par exemple après un silence prolongé du client, la durée variant selon l'application utilisant la connexion *TCP/IP* côté serveur. Avec certaines applications, il est possible de forcer cet état ; c'est le cas pour les serveurs `HTTP` auxquels il suffit d'envoyer une requête `GET␣/␣HTTP/1.0\r\n\r\n`.

Dès que l'application décide de clore la connexion, le serveur passe dans l'état `FIN WAIT 1` et émet un paquet `FIN` pour informer le client. Si celui-ci ne répond pas, le serveur conclut à la congestion, et donc retransmet. Ce comportement autorise le même type de mesures que dans les cas précédents.

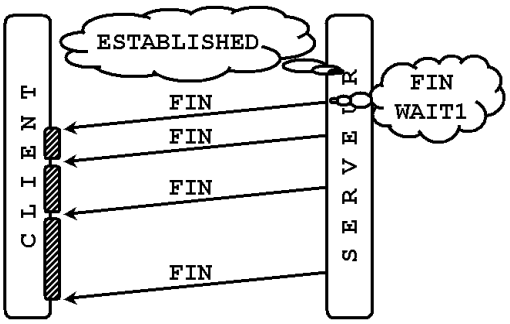


Fig. 6. Terminaison à l'initiative du serveur, avec congestion simulée (par le client).

Fonctionnement de `RINGv2`

L'outil `RINGv2` mesure les délais liés aux trois états `SYN-RCVD` (comme `RING`), `CLOSE WAIT` et `FIN WAIT 1`. Pour les deux derniers états, la mesure est délicate car elle peut dépendre du passé de la connexion et de la manière dont le client et le serveur ont ajusté leurs valeurs de RTT et de RTO. En effet, afin d'optimiser les performances de leur connexion *TCP/IP*, le client et le serveur calculent en permanence les valeurs de RTT et RTO. Sur certains systèmes d'exploitation, ces valeurs sont prises en compte pour déterminer les délais entre paquets retransmis en cas de congestion. C'est par exemple le cas pour des systèmes Linux ou SUN Solaris mais pas pour certains IBM AIX.

À lui seul, ce comportement donne donc un indice sur le système d'exploitation, mais il est également source d'erreurs de détection. En effet, lorsqu'une comparaison simple entre des temps mesurés et des temps de référence ne permet pas de conclure, deux attitudes sont possibles : conclure que le système détecté est inconnu de la base, ou bien tenter une nouvelle comparaison en multipliant les valeurs de références par le RTT approximé. Si le système détecté existe dans la base de référence et s'il prend effectivement en compte les valeurs de RTT et de RTO dans ses calculs de délai, alors cette nouvelle comparaison sera conclusive ; dans le cas contraire il y a un nouvel échec. Cette seconde comparaison conduit à plus de faux-positifs lorsqu'elle est activée, et plus de faux-négatifs lorsqu'elle est désactivée.

En résumé : `RING` *mesure des temps révélateurs de l'algorithme de gestion de congestion pour l'état* `SYN RCVD`. `RINGv2` *mesure de plus les temps liés aux états* `LAST ACK` *et* `FIN WAIT 1` *et obtient en général des mesures plus précises. Dans quelques cas toutefois, l'interprétation des valeurs mesurées est beaucoup plus complexe.*

2.3 Effet des composants intermédiaires

L'*OSFP* n'est pas une technique exacte, notamment à cause de toutes les inconnues concernant l'effet des composants intermédiaires.

Fluctuations du réseau

Le premier composant intermédiaire traversé lors d'une détection est le réseau lui-même. S'il s'agit d'un réseau local, on peut s'attendre à quelques fluctuations liées à des congestions (véritables :-), des ralentissements de machines, etc. Sur Internet des changements de routes, des congestions supplémentaires liées aux routeurs, etc. peuvent également se produire.

En pratique, avec les paquets sans *payload* utilisés par `RINGv2`, les variations constatées sur Internet en contexte opérationnel restent de l'ordre du dixième de seconde. À titre d'exemple, voici une mesure ponctuelle d'écart sur des liens France-France ou France-Japon :

```
France-France 20 packets transmitted, 20 received, 0\% loss, time 19017ms
rtt min/avg/max/mdev = 67.828/70.299/73.020/1.372 ms
```

```
France-Japon 50 packets transmitted, 50 received, 0\% loss, time 49002ms
rtt min/avg/max/mdev = 333.417/336.614/343.022/2.121 ms
```

Les variations constatées sur cet exemple (et sur bien d'autres) n'excèdent donc pas 1% des temps mesurés par `RING` et `RINGv2`, de l'ordre de la dizaine de secondes.

Effet des relais `SYN`

Ces appareils, généralement intégrés à des routeurs ou à des pare-feu, protègent un réseau en endossant pour le compte des serveurs tout ou partie de la phase de connexion *TCP/IP*. En présence d'un relais `SYN`, la plupart des outils d'*OSFP* vont donc mesurer les caractéristiques du relais, et pas celle du serveur (cf. figure 7).

Il est rare qu'un relais `SYN` traite également les transmissions à l'initiative du serveur, et encore moins le traffic lié à la terminaison par le serveur de sa demi-connexion *TCP/IP*. Or c'est exactement le cas que provoque `RINGv2` en plaçant le serveur soit dans l'état `LAST ACK` soit dans l'état `FIN WAIT 1`. Dans ce cas, c'est bien les caractéristiques du serveur qui sont mesurées, malgré la présence du relais.

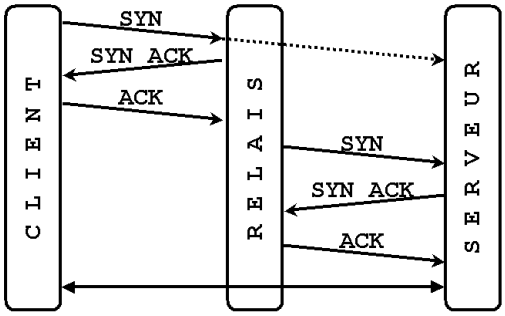


Fig. 7. Schéma de principe d'un relais `SYN`. Les paquets analysés par `RINGv2` sont émis à l'initiative du serveur, ils ne sont pas obligatoirement relayés.

Effet des systèmes de détection d'intrusion

Les IDS fonctionnant par signature repèrent difficilement les paquets échangés au cours d'une détection `RINGv2` puisque ceux-ci sont conformes aux RFC. Une règle simple a pourtant été proposée sur la liste de discussion de `SNORT` :

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any
(msg:"Possible RING Scan";flags:SA; classtype:attempted-recon;
tag: host, 10, packets, dest;)
```

Cette règle produit énormément de fausses alertes, en particulier lorsque le réseau congestionne réellement ou lorsque les machines internes ralentissent. T. Beardsley [2] a remarqué une caractéristique des paquets forgés dans la première version de `RING` : l'`ISN` et le `timestamp` ont toujours les mêmes valeurs. Précisons que cette caractéristique est entièrement liée à l'implémentation actuelle de `RING`, la supprimer ne pénaliserait pas le processus de détection. Les règles `SNORT` suivantes reconnaissent spécifiquement ces paquets :

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(flags:S;seq:221002;msg:"SCAN RING";classtype:attempted-recon;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any
(flags:SA;ack:221003;msg:"SCAN RING response";
                          classtype:successful-recon-limited;)
```

Un mot sur les *scrubbers*

Ce terme apparu assez récemment dans la littérature désigne un élément réseau filtrant permettant de "nettoyer" à la volée une partie du trafic. M. Smart et al. [8] proposent ainsi d'effectuer des actions de nettoyage et d'anonymisation au niveau *TCP/IP*. Ils abordent la notion d'*OSFP* basé sur des analyses temporelles mais indiquent qu'une protection contre ce type de prise d'information, bien que possible à mettre en œuvre, s'avère assez complexe.

En résumé : Les composants intermédiaires ont finalement peu d'impact sur le trafic généré par `RING` *ou* `RINGv2`*. Les IDS par signature ne sont pas suffisament discriminants pour déclencher un blocage actif. Les relais* `SYN` *fournissent une bonne protection face à* `RING` *(et beaucoup d'autres outils), mais restent peu efficaces face à* `RINGv2`*.*

3 Utilisation de `RING` et de `RINGv2`

Cette section est essentiellement destinée au lecteur souhaitant utiliser `RING` et `RINGv2`, voire modifier les codes source pour des usages particuliers.

3.1 Implémentation

L'implémentation actuelle de `RINGv2` fait appel à trois bibliothèques :
- `Libpcap` pour l'écoute du réseau et la récupération des paquets.
- `Libnet` pour la construction précise de paquets et leur transmission.
- `Libdnet` pour ses fonctions de filtrage et de contrôle des pare-feu de nombreux systèmes d'exploitation.

La fonction de capture des paquets fournie par `libpcap` est exécutée avant la fonction de filtrage fournie par `libdnet`, ce qui permet de recevoir des paquets normalement filtrés par le pare-feu. La `libdnet` n'est utilisée que pour les tests liés à l'état `SYN RCVD`. Pour les nouveaux tests de `RINGv2` les fonctions de `libdnet` ne permettaient pas un contrôle suffisament précis. Une partie du

code est donc dédiée, et moins facilement protable (actuellement, cette partie est programmée pour Linux 2.4 uniquement).

Le pseudo-code ci-après donne l'algorithme général de RINGv2 :

```
ringv2() {
  libdnet.filtrer_paquet (machine_cible, port_ouvert) ;
  libnet.envoi_paquet (SYN, machine_cible, port_ouvert) ;
  while(1) {
    temps = libpcap.reception_paquet (machine_cible, ports, flags) ;
    mesures[n++] = temps ;
  }
  ringv2.confronter (mesures[ ], signatures[ ]) ;
}
```

3.2 Intégration avec NMAP

Chaque technique d'*OSFP* fait des erreurs de détection, mais toutes les techniques ne font pas forcément les mêmes erreurs. D'où l'idée d'utiliser plusieurs techniques pour une même détection, en gérant correctement la complémentarité. Cette idée est poussée à l'extrême avec les travaux d'O. Arkin et F. Yarochkin sur XPROBE2 [1] qui permet d'intégrer plusieurs techniques et de pondérer leurs résultats.

Pour des raisons de simplicité et de connaissance des outils, nous avons choisi d'intégrer RINGv2 à NMAP, deux outils qui offrent une grande complémentarité (en particulier, NMAP différencie mal FreeBSD 4.4 et Windows 2000 alors que RINGv2 les confond moins souvent).

Au-delà de la complémentarité, l'intégration avec NMAP permet de bénéficier d'un format de signature, de conventions de nommage des système d'exploitation et d'une ligne de commande déjà bien connus des utilisateurs. L'outil résultat s'appelle nmap-ringv2 et sa ligne de commande à l'allure suivante :

```
# ./nmap-ringv2  -p 80,1234  --ring --ring_method slf
                 --ring_timeout 60  -O shal.no-ip.org
      -p 80,1234         Ports 80 et 1234 testés
      --ring             Méthodes ring et / ou ringv2 utilisées
      --ring_method slf  Etats s(yn), l(ast-ack) et f(in-wait1) analysés
      --ring_timeout 60  Ecoute de 60 secondes pour chaque état analysé
      -O                 Prise en compte des tests habituels de NMAP
      shal.no-ip.org     Nom de la cible
```

Les signatures sont une simple extension des signatures NMAP habituelles, en ajoutant 1, 2 ou 3 lignes supplémentaires correspondant respectivement aux tests s(yn), (l)ast-ack et (f)in-wait1. Voici un exemple de signature unifiée nmap-ringv2 :

```
Fingerprint: Windows NT
TSeq(Class=RI%gcd=1%SI=3CC4%IPID=I...
T1(Resp=Y%DF=Y%W=FFFF%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=FFFF%ACK=S++%Flags=AS%Ops=MNWNNT)
```

```
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
Ring_Syn(nbPkt=2%Time=30%p=2806316%p=6010910)
Ring_LastAck(nbPkt=3%Time=30%Connect=134571%p=2918695%p=5909092%p=12014080)
Ring_FinWait(nbPkt=0%Time=30%Connect=136213)
```

3.3 Exemples simples

Sur ce premier exemple, un système Windows 2000 est protégé par un pare-feu très fermé. Les tests T2 à T7 et PU de `NMAP` n'apportent donc pas d'information. Ici, c'est `RINGv2` permet de conclure correctement.

```
# nmap -n -P0 -v -d -O --osscan_guess -e eth0
                                  -p 22,80,6523 win-2K.qqpart.com
Aggressive OS guesses: AIX 4.3.2.0-4.3.3.0 on an IBM RS/* (88%),
AIX v4.2 (87%), IBM AIX v3.2.5-4 (87%), AIX 4.2-4.3.3 (87%)
SInfo(V=3.00%P=i686-pc-linux-gnu%D=4/2%Time=3E8B431D%O=80%C=-1)
TSeq(Class=TR%IPID=RD%TS=0)
T1(Resp=Y%DF=N%W=4000%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N) T3(Resp=N) T4(Resp=N) T5(Resp=N) T6(Resp=N) T7(Resp=N)
PU(Resp=N)

# nmap-ringv2 -n -P0 -v -d --ring --ring_method sf -e eth0
                                 -p 22,80,6523i win-2K.qqpart.com
Remote operating system guess: Windows 2K RINGv2
Ring_Syn(nbPkt=2%Time=30%p=3272209%p=6562494)
Ring_FinWait(nbPkt=4%Time=30%Connect=230166%p=849410%
                                   p=1749083%p=3280573%p=6563935)
```

Le deuxième exemple met en évidence la synergie entre `NMAP` et `RINGv2` lorsque les deux outils arrivent à une même conclusion. L'indice de croyance est renforcé : $44/46 < 46/48$.

```
# nmap-ringv2 -n -P0 --osscan_guess -O   -e eth0
                                        -p 80 lin24.qqpart.com
Remote OS guesses: Linux Kernel 2.4 Ringv2(44success/46tests),
Linux Kernel 2.4.0 - 2.5.20(44success/46tests)

# nmap-ringv2 -n -P0 --osscan_guess -O --ring
                 --ring_method sf -e eth0 -p 80 lin24.qqpart.com
Remote OS guesses: Linux Kernel 2.4 Ringv2(48success/48tests),
Linux Kernel 2.4.0 - 2.5.20(46success/48tests)
```

3.4 Exemples avancés

Voici un exemple où `NMAP` et `RINGv2` ont tous deux l'air de fonctionner. Pourtant, chacun donne des résultats très différents. `NMAP` utilisé seul répond "`Nokia M1122 DSL`

Router". Pour sa part, RINGv2 utilisé seul, c'est-à-dire avec seulement les tests liés aux états LAST ACK et FIN WAIT 1, répond "Win2k".

```
#nmap -n  -P0  -p 80 -O  x.x.x.x
Port       State       Service
80/tcp     open        http
Remote operating system guess: Nokia M1122 DSL Router
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=5937%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(Resp=N) T5(Resp=N) T6(Resp=N) T7(Resp=N) PU(Resp=N)

#nmap-ringv2  -n  -P0  -p 80 --ring --ring_method fl x.x.x.x
Port       State       Service
80/tcp     open        http
Remote OS guesses: Win2k Pro Base/SP1/SP2/SP3, Win2k Srv
Base/SP1/SP2/SP3, Win2k AdvSrv Base/SP1/SP2/SP3 (2success/2tests),
Windows 95 B RINGv2(2success/2tests)...
OS Fingerprint:
Ring_FinWait(nbPkt=3%Time=30%Connect=63863%p=1454762%p=4796315%p=9609807)
Ring_LastAck(nbPkt=3%Time=30%Connect=26047%p=2971331%p=6002374%p=12020256)
```

Finalement, c'est après avoir lancé les deux outils qu'on obtient une analyse acceptable de la situation : la cible est probablement un Windows 2000 caché derrière un relais SYN Nokia. En fait, NMAP a fait une empreinte du relais SYN et RINGv2 a fait une empreinte de la cible réelle.

Conclusion

Avec des outils comme NMAP, XPROBE2, TBIT, RINGv2, etc. le domaine de l'*OSFP* a fortement évolué depuis les premières publications sur le sujet ; gageons qu'il reservera encore des surprises dans un avenir proche.

Concernant une évolution de RINGv2, de prochains développements pourraient offrir plus de fonctionnalités et une meilleure précision, par exemple en mesurant les réactions des systèmes à la réception de paquets non prévus dans le diagramme d'état *TCP/IP*. NMAP procède de cette manière depuis toujours, mais uniquement lorsque le serveur se trouve dans l'un des états LISTEN ou CLOSED. D'autres états comme SYN RCVD ou FIN WAIT 1 sont pourtant très prometteurs...

A propos du projet

RINGv2 est un projet *Open Source* hébergé sur le site ringv2.tuxfamily.org. Les commentaires ou questions sont encouragés à l'adresse ringv2@tuxfamily.org. Les auteurs désaprouvent l'usage de RING et de RINGv2 à des fins malhonnêtes ou illégales.

Remerciements

Les auteurs tiennent à remercier pour leur aide messieurs Nicolas Prigent et David Fort, ainsi que les rapporteurs de cet article, messieurs Laurent Oudot et Sylvain Gombault.

Références

1. O. Arkin, F. Yarochkin, 2002. *XProbe2 - A 'Fuzzy' Approach to Remote Active Operating System Fingerprinting.* `www.xprobe2.org/archive/papers/Xprobe2.pdf`.
2. T. Beardsley, Plan B Security, 2002. *RING Out The Old, RING In The New : OS Fingerprinting through RTOs.* `www.planb-security.net/wp/ring.html`.
3. D. Comer, J. Lin, USENIX Summer Conf. 1994. *Probing TCP Implementations.* `www.bell-labs.com/user/johnlin/probing-TCP.pdf`.
4. Fyodor, Phrack 1998. *Remote OS detection via TCP/IP Stack FingerPrinting.* `www.insecure.org/nmap/nmap-fingerprinting-article.txt`.
5. P. Karn, C. Partridge, SIGCOMM 87. *Improving Round-Trip Time Estimates in Reliable Transport Protocols.*
6. B. Morin, L. Mé, H. Debar, M. Ducassé. *M2D2 : A Formal Data Model for IDS Alert Correlation. RAID 2002 : 115-127*
7. J. Padhye, S. Floyd, SigComm 2001. *Identifying the TCP Behavior of Web Servers.* `www.icir.org/tbit/nanog-tbit.pdf`.
8. M. Smart, G. R. Malan, F. Jahanian, 9th USENIX Security Symp. *Defeating TCP/IP Stack Fingerprinting.*
9. F. Veysset, O. Courtay, O. Heen *RING : New Tool and Technique For Remote OSFP.* `www.intranode.com/site/techno/techno-articles.htm`
10. F. Veysset, O. Courtay, O. Heen. *RINGv2 Information Page.* `ringv2.tuxfamily.org`
11. M. Zalewski, 2001. *Strange Attractors and TCP/IP Sequence Number Analysis.* `lcamtuf.coredump.cx/newtcp`.
12. Quelques outils :
    Ettercap (`ettercap.sourceforge.net`), p0f (`www.stearns.org/p0f/README`), IP Personality (`ippersonality.sourceforge.net`), RUT (`www.thc.org/thc-rut`)

Sécurité des réseaux domestiques : optimaux les grands remèdes ?

Nicolas Prigent[1], Christophe Bidan[2], Olivier Heen[1], and Alain Durand[1]

[1] *Thomson R&I France, Rennes
{nicolas.prigent|olivier.heen|alain.durand}@thomson.net
[2] **Supélec, Rennes
christophe.bidan@supelec.fr

Résumé Au carrefour de l'informatique traditionnelle et du *consumer electronics*, les réseaux domestiques promettent de nouveaux usages et de nouveaux horizons pour le grand public. Ils ne pourront tenir ces promesses si leur sécurité n'est pas assurée. Cet article montre en quoi les solutions initialement élaborées pour un environnement professionnel offrent peu de sécurité en environnement domestique ou contraignent trop le réseau et son utilisateur. Puis il aborde quelques pistes récentes vers une sécurisation spécifique.

1 Réseaux domestiques

De plus en plus accessibles, de moins en moins coûteux, les équipements et réseaux informatiques se sont largement démocratisés : selon Médiamétrie [9], 38% des foyers français sont équipés d'un ordinateur et 25% disposent d'une connexion à l'Internet.

Les réseaux domestiques, au carrefour de l'informatique et de l'électronique grand public, sont la prochaine évolution attendue.

Un réseau domestique consiste en l'interconnexion d'équipements domestiques hétérogènes (ordinateurs, téléviseurs, assistants numériques personnels, matériel Hi-Fi, appareils électroménagers, etc.) dans le but de proposer des services augmentés aux utilisateurs. Cette interconnexion permet par exemple d'accéder aux contenus ou services multimédias présents sur le réseau depuis tout équipement en faisant partie et doté de capacités suffisantes. L'accès peut se faire soit depuis l'intérieur du domicile (sur des média filaires ou non), soit depuis l'extérieur via l'Internet. La synchronisation automatique entre les équipements domestiques fixes et les réseaux personnels (*PAN, Personnal Area Network*) de chacun des habitants est un autre exemple de service augmenté offert par les réseaux domestiques.

1.1 Origines

Les réseaux domestiques trouvent leur origine dans le projet *Ubiquitous Computing* (informatique omniprésente) mené par Marc Weizer au Xerox Parc à la fin des années 80.

Ce projet s'appuie sur le constat suivant : chaque fois qu'une technologie devient primordiale, son utilisation devient transparente pour l'utilisateur[3]. Weizer [11] af-

[3] L'écriture, par exemple, est devenue "transparente" pour une grande partie de la population : nous lisons chaque jour énormément de messages (publicités, marques, panneaux) sans nous en rendre compte.

firme que, pour devenir omniprésente et véritablement utile, l'informatique doit devenir transparente : les ordinateurs doivent être enfouis dans les objets du quotidien, communiquer, se configurer et fournir des services sans que l'utilisateur s'en aperçoive. Les ParcTabs [8] sont un exemple historique parmi tant d'autres [1] de mise en oeuvre de l'*Ubiquitous Computing*.

Les réseaux domestiques sont une forme particulière d'*Ubiquitous Computing* se focalisant sur les lieux de résidence et leurs habitants. Plusieurs visions coexistent actuellement. Elles sont basées soit sur des technologies informatiques, soit sur des technologies issues de l'électronique grand public.

1.2 Technologies actuelles

Une première approche envisage les réseaux domestiques comme une évolution des LANs autour d'un élément fédérateur : l'ordinateur domestique. C'est l'approche de solutions telles que *Universal Plug'n Play (UPnP)* et *Rendezvous*. Elles permettent de gérer automatiquement l'installation et la configuration de nouveaux appareils fonctionnant sur IP et celle des services associés (partage de fichiers, d'imprimantes, de connexions à l'Internet) de manière transparente pour l'utilisateur.

Une autre approche est plutôt héritée de l'électronique grand public [7] : Bell et Gemmell prédisent l'avènement d'une nouvelle manière de gérer l'interaction des différents équipements audiovisuels chez le particulier. Les dispositifs sont tous connectés entre eux par un bus numérique (typiquement un bus IEEE 1394) au travers duquel ils échangent contenus, accès et traitements. Via ce bus, les téléviseurs, enregistreurs numériques, chaînes Hi-Fi, etc. s'interconnectent facilement et adaptent conjointement leur configuration. La technologie HAVi (*Home Audio-Video Interoperability*) par exemple permet ce type d'interactions.

Les deux approches ne s'excluent pas mutuellement, et certains efforts de recherche s'orientent aujourd'hui vers des approches hybrides.

1.3 Caractéristiques

Pour remporter l'adhésion du grand public et rendre pleinement les services attendus, les réseaux domestiques doivent respecter un certain nombre de contraintes. En particulier, l'accent doit porter sur l'automatisation des procédures d'installation et de configuration complexes permettant aux dispositifs de collaborer. A titre d'exemple, on peut citer l'obtention d'une adresse valide sur le réseau, la déclaration de la disponibilité sur un bus ou dans une cellule hertzienne, l'annonce des services proposés, etc.

Plus globalement, les réseaux domestiques doivent :
- Permettre une grande hétérogénéité.
- Tolérer des comportements très dynamiques.
- Fonctionner en l'absence d'administrateur.

1.3.1 Hétérogénéité

Alors que les réseaux informatiques classiques sont formés de composants similaires (principalement des ordinateurs) comparables en termes de puissance, de stockage, de mémoire, de bande passante, etc., les réseaux domestiques sont composés de dispositifs

hétérogènes. Ils comportent aussi bien du matériel Hi-Fi que des dispositifs mobiles et du matériel informatique. Cette hétérogénéité a des conséquences sur les réseaux domestiques : à titre d'exemple, on ne peut demander au petit processeur placé dans un lecteur MP3 de réaliser les mêmes opérations qu'un ordinateur de bureau. D'autres dispositifs tels que les assistants numériques personnels présentent de meilleures capacités de traitement mais souffrent d'un stockage limité.

Une autre différence porte sur l'hétérogénéité des moyens de communication. Grâce à IP, les réseaux informatiques classiques sont extrêmement homogènes : une technologie de communication physique est choisie, les appareils sont équipés de cartes réseau en conséquence, et l'administrateur effectue les tâches de configuration. À l'inverse, les moyens de communication utilisés par les réseaux domestiques sont hétérogènes dans la mesure où les appareils domestiques sont fournis tels quels : un enregistreur numérique n'utilise pas la même connectique qu'un assistant numérique personnel, et encore moins les mêmes protocoles de communication.

1.3.2 Dynamicité

Les réseaux domestiques sont aussi particulièrement dynamiques, et ce à plusieurs titres.

Tout d'abord, un réseau domestique donné évolue dans le temps en fonction des dispositifs qui le constituent. Il commence avec le premier appareil communicant que l'utilisateur achète et évolue au gré des achats, ventes, pannes, pertes, etc.

Un réseau domestique est aussi physiquement dynamique. Les éléments qui le composent ne sont pas interconnectés en permanence : les appareils peuvent être mobiles, sous tension ou non ; les canaux de communication peuvent être bruités ou temporairement indisponibles.

Plus généralement, aucune supposition ne peut être faite sur la disponibilité d'un dispositif dans le réseau à un instant donné. Par conséquent, aucun constituant du réseau domestique ne doit être indispensable au fonctionnement global. C'est une grande différence par rapport aux réseaux informatiques classiques, pour lesquels on peut raisonnablement estimer que certains des dispositifs sont disponibles en permanence (solutions haute disponibilité, locaux sécurisés, courant secouru, supervision).

1.3.3 Absence d'administrateur

Enfin, les utilisateurs des réseaux domestiques sont très différents de ceux des réseaux informatiques. L'utilisateur d'un réseau informatique a souvent bénéficié d'une formation adaptée. Les appareils sont pour lui des outils de travail et il est prêt à suivre quelques procédures fastidieuses pour les maintenir en fonction. Pour les opérations complexes, un administrateur possède le niveau d'expertise adapté et des ressources dédiées.

A contrario, l'utilisateur d'un réseau domestique interagit avec des objets du quotidien. Il n'a généralement bénéficié d'aucune formation particulière et peut tout ignorer du fonctionnement d'un réseau. Quand bien même il disposerait de compétences adaptées, il ne consacrera probablement pas les ressources nécessaires pour configurer, administrer, et superviser régulièrement le sien.

Les réseaux domestiques présentent donc des différences essentielles par rapport aux réseaux informatiques traditionnels, résumées dans le tableau ci-dessous.

Réseaux informatiques	Réseaux domestiques
Dispositifs similaires.	Dispositifs hétérogènes.
Moyens de communication rationalisés.	Moyens de communication hétérogènes.
Évolutions du réseau rares et maîtrisées.	Évolutions du réseau fréquentes et non-rationnelles.
Interconnexion supposée permanente des dispositifs.	Interconnexion erratique des dispositifs.
Utilisateurs formés et actifs.	Utilisateurs non-formés et passifs.
Administrateurs.	Pas d'administrateur.

Tab. 1. Différences entre réseaux informatiques et réseaux domestiques .

En quoi ces différences influent-elles sur les besoins et les modèles de sécurité ? Des solutions de sécurisation éprouvées pour les réseaux informatiques classiques sont-elles toujours valides en environnement domestique ?

2 Besoins de sécurité

Les réseaux domestiques seront attaqués s'ils recèlent de la valeur (contenus, ressources, usages), et ce avec succès s'ils présentent des faiblesses exploitables.

2.1 Menaces

Aujourd'hui, il existe déjà des menaces contre les ordinateurs personnels connectés à l'Internet, qui peuvent être considérés comme des réseaux domestiques limités. Les attaques auparavant uniquement portées contre les sociétés et les centres militaires ou universitaires frappent maintenant les particuliers [17].

Les mobiles pour attaquer de tels équipements domestiques sont nombreux :

- Le jeu : de nombreux attaquants sont simplement des passionnés d'informatique souhaitant étrenner leurs connaissances des réseaux et des systèmes [2].
- Le vandalisme.
- Le vol des contenus (données, films, musiques, images, etc.) de l'utilisateur.
- L'atteinte à la vie privée de l'utilisateur, par exemple en accédant à ses e-mails, ou en surveillant le trafic IP arrivant à son ordinateur.
- Le vol de ressources (bande passante, CPU ou disque) : certains pirates déposent par exemple des données, légales ou non, sur les ordinateurs qu'ils ont attaqués, et les rendent accessibles *via* un serveur `FTP` ou `HTTP`.
- Le rebond vers d'autres cibles plus importantes : une fois que l'attaquant a pris le contrôle d'une machine, il peut s'en servir pour en attaquer d'autres, réduisant ainsi le risque d'être découvert.
- L'utilisation dans le cadre d'une attaque par refus de service distribué (*DDoS, Distributed Denial of Service*).

Ces mobiles sont toujours valides, et sont probablement accentués dans le cas des réseaux domestiques complets, notamment dans le cas de l'atteinte à la vie privée : en permettant aux utilisateurs de gérer l'intégralité de leurs informations personnelles (e-mails, photos et films personnels, données bancaires et commerciales, etc.), les réseaux domestiques seront profondément liés à leurs vies quotidiennes. Ils seront donc une cible de choix pour quiconque voudra porter atteinte à la vie privée de leurs propriétaires, réaliser des opérations bancaires à leurs dépens, etc.

Si, en plus de mobiles, un attaquant dispose d'opportunités pour attaquer les réseaux domestiques, ceux-ci seront effectivement attaqués. La seule présence d'ordinateurs dans les réseaux domestiques rend des attaques connues techniquement possibles. Les attaques contre les ordinateurs domestiques sont actuellement très courantes, et d'autant plus faciles que ces ordinateurs utilisent majoritairement la même architecture et le même système d'exploitation. En utilisant des outils d'attaque automatisés exploitant des failles connues des logiciels ou des systèmes d'exploitation les plus courants, l'attaque d'une machine peut se faire d'un simple clic.

En outre, les ordinateurs domestiques bénéficiant rarement d'un administrateur compétent, ils sont en général moins bien protégés que leurs homologues professionnels : leurs utilisateurs n'appliquent que très rarement les mises à jour de sécurité. Ainsi, des vulnérabilités pour lesquelles il existe pourtant des correctifs perdurent [5]. À cela s'ajoute l'installation chaotique d'applications éventuellement corrompues par des logiciels malicieux (virus et autres chevaux de Troie).

Les ordinateurs personnels ne sont pas les seules cibles à considérer. En effet, tout équipement communiquant peut faire l'objet d'attaques, à l'instar de celles réalisées contre des dispositifs fonctionnant sous des systèmes d'exploitation embarqués.

Lors de la conférence Black-Hat 2002, Davis et Higbee [6] ont présenté une attaque utilisant des dispositifs électroniques grand public disposant d'interfaces Ethernet ; Ils ont transformé une console DreamCast de SEGA et un assistant personnel I-Paq de Compaq en analyseurs de réseau en se servant de portages spécifiques de Linux[4], puis les ont utilisés pour contourner un *firewall*.

Plus récemment, une attaque portant contre le Nokia 6210 a été publiée [3]. Ce téléphone mobile offre un service de gestion de cartes de visite électroniques *vCard*, qui peuvent notamment être échangées par SMS. En envoyant une *vCard* au format incorrect par SMS, un attaquant peut provoquer un dépassement de tampon [13], causant un refus de service sur le téléphone.

Si ces attaques contre des dispositifs particuliers restent pour l'instant l'apanage de spécialistes, il est très probable que des outils d'attaque automatisés existeront contre l'ensemble des dispositifs lorsque les réseaux domestiques seront largement déployés.

Les éléments portatifs présentent également un risque important de perte, de vol, ou de modification à l'insu de la victime. Outre le préjudice immédiat en cas de vol, il y a une augmentation du risque d'attaque avec privilèges si l'appareil n'est pas révoqué rapidement.

[4] La Dreamcast se base sur un processeur Hitachi SH4, et l'I-Paq sur un StrongARM. Un portage de Linux existe pour chacune de ces architectures.

Il existe donc simultanément des mobiles et des opportunités d'attaques contre les réseaux domestiques : toutes les conditions sont réunies pour qu'ils soient attaqués avec succès. Seule la mise en œuvre de mesures de prévention spécifiques permettra de réduire le risque à un niveau acceptable pour les utilisateurs.

2.2 Frontière

Une première étape vers la sécurisation des réseaux domestiques consiste en la définition et la mise en place d'une frontière. La frontière d'un réseau domestique marque la séparation entre les équipements placés sous la responsabilité de l'utilisateur et le monde extérieur. Son implémentation doit présenter les services de sécurité suivants :

- Authentification des dispositifs : les dispositifs situés à l'intérieur de la frontière peuvent s'authentifier comme tels.
- Confidentialité des communications : seuls les éléments situés à l'intérieur de la frontière ont accès aux messages circulant dans le réseau domestique.
- Authenticité des communications : les dispositifs du réseau domestique peuvent vérifier si un message provient bien d'un dispositif situé à l'intérieur de la frontière.

Le problème à résoudre réside dans l'établissement d'une frontière offrant ces services et maintenable au gré des évolutions du réseau domestique, en accord avec les caractéristiques d'hétérogénéité, de dynamicité et d'absence d'administration.

En premier lieu, la frontière doit être non-ambiguë. Un utilisateur connaît les dispositifs appartenant au réseau domestique. La frontière doit refléter cette appartenance : deux dispositifs mis en présence doivent pouvoir déterminer avec certitude s'ils appartiennent au même réseau domestique.

La frontière doit aussi suivre l'évolution du réseau domestique. Un réseau domestique commençant par le premier appareil communiquant acquis par l'utilisateur, la frontière doit être initialisable. Elle doit aussi être incrémentale et décrémentale : un utilisateur peut ajouter un élément au réseau domestique, comme il peut aussi exclure un dispositif d'un réseau, éventuellement en l'absence de ce dispositif (révocation en cas de vol).

Enfin, la frontière doit être tolérante au fractionnement, pour supporter la dynamicité physique du réseau domestique. En particulier, deux partitions d'un même réseau domestique peuvent évoluer séparément, et doivent pouvoir synchroniser leurs évolutions respectives (par exemple l'acquisition d'un nouveau dispositif par l'une des parties) lorsqu'elles seront de nouveau réunies.

3 Insuffisance des solutions traditionnelles

Une approche pour mettre en œuvre la frontière et sécuriser les réseaux domestiques consiste à se servir des solutions existant dans les réseaux informatiques. Actuellement, trois types de solutions y sont utilisés :

1. Le cloisonnement.
2. Les solutions centralisées.
3. La protection des communications.

3.1 Cloisonnement

Les solutions de cloisonnement se servent de l'organisation physique du réseau pour en assurer la sécurité. Pendant longtemps, ces solutions étaient les plus répandues pour sécuriser les réseaux informatiques : seuls les ordinateurs physiquement connectés au médium étaient considérés comme faisant partie du réseau, à l'exclusion de tous les autres. Dans la mesure où seuls des utilisateurs légitimes pouvaient accéder physiquement au réseau, celui-ci était réputé sûr. Plus tard, lorsque ces réseaux ont été interconnectés, la mise en place d'un *firewall* permettait de délimiter, au moins partiellement, la frontière du réseau local vis-à-vis de l'Internet.

La situation est différente dans les réseaux domestiques : la possibilité d'accéder au médium ne suffit pas à décider de l'appartenance au réseau, du fait notamment de l'utilisation éventuelle de réseaux sans fil[5]. De plus, la très grande dynamicité physique des réseaux domestiques, ainsi que la possibilité d'usages multi-sites (correspondant au besoin de relier les résidences principale et secondaire notamment) réduisent grandement l'intérêt d'un *firewall* classique.

3.2 Solutions centralisées

Une autre manière de définir l'appartenance à un groupe dans les réseaux informatiques repose sur des solutions centralisées : un serveur en ligne est capable d'authentifier les dispositifs autorisés, à l'image de Kerberos.

Ces solutions sont difficilement envisageables pour les réseaux domestiques. Du fait de leur dynamicité, on ne peut pas supposer qu'un dispositif est joignable en permanence : certains dispositifs pouvant se retrouver momentanément déconnectés (par exemple, un assistant numérique personnel, un ordinateur portable, un téléphone mobile, etc.), l'utilisation d'un dispositif central de sécurité les empêcherait de fonctionner.

Une alternative consisterait à utiliser un serveur centralisé n'ayant pas à être joignable en permanence car il serait capable de certifier les dispositifs en se servant d'algorithmes de cryptographie asymétrique. Un dispositif appartenant au réseau domestique disposerait donc d'un certificat signé par ce serveur, ainsi que de la clé publique de celui-ci, lui permettant de vérifier les certificats.

Cependant, si cette alternative permet l'authentification mutuelle de deux dispositifs même lorsque ceux-ci ne peuvent communiquer avec le serveur central, elle requiert malgré tout que celui-ci soit présent lorsqu'il s'agit de faire évoluer le réseau, et n'est donc pas totalement adaptée.

3.3 Protection des communications

Construire un Réseau Privé Virtuel (*Virtual Private Network* ou *VPN*) entre les dispositifs du réseau domestique pourrait être une autre manière de mettre en place la frontière d'un réseau domestique. Un VPN simule un réseau privé en utilisant la

[5] Les attaques par *war-driving* et l'engouement récent pour le *war-chalking* sont particulièrement symptomatiques de cet état de fait.

cryptographie pour sécuriser les communications : les messages qui circulent entre les membres du VPN sont chiffrés et authentifiés. IPSec est aujourd'hui le protocole le plus utilisé pour la mise en place de VPNs sur l'Internet. D'autres protocoles sont aussi utilisés : TLS/SSL (*Transport Layer Security / Secure Socket Layer*) est une proposition générique de sécurité au niveau Socket, alors que SSH (*Secure Shell*) est dédié à la sécurisation des protocoles d'interpréteurs de commande à distance.

Les protocoles de réseaux locaux sans-fil bénéficient eux aussi de mécanismes pour assurer la sécurité des communications. IEEE 802.11 par exemple a successivement connu différents mécanismes de sécurité, WEP étant le plus déployé.

La sécurité des communications est donc un axe de recherche qui a déjà été largement exploré. De nombreux algorithmes de chiffrement et autres protocoles de communication existent et ont jusqu'à présent bien résisté aux attaques. Bien que ces mécanismes soient une partie essentielle de la sécurité des VPNs, ils n'en sont qu'un aspect. En effet, la mise en place et la gestion d'un VPN entre les différents dispositifs d'un réseau requièrent de longues et complexes étapes d'installation et de configuration. La sécurité résultante est donc fortement dépendante de la compétence de l'administrateur [16].

Cette très forte implication de l'utilisateur dans la configuration de la sécurité n'est pas acceptable pour les réseaux domestiques. À quoi bon en effet disposer de réseaux qui s'auto-configurent dynamiquement s'il faut consacrer beaucoup de temps à la génération des clés, à leur gestion, à leur mise en place, etc.

De plus, les utilisateurs, et *a fortiori* les utilisateurs des réseaux domestiques, sont souvent considérés comme le maillon faible de la sécurité. Parce qu'ils ne sont pas formés aux règles de la sécurité, il peuvent être victimes du *social engineering*.

Enfin, les utilisateurs ne doivent surtout pas être impliqués dans la gestion de la sécurité de leur réseau domestique. En effet, l'expérience montre [10] que lorsqu'on laisse la sécurité entre les mains des utilisateurs, ils ont tendance à y préférer la facilité d'utilisation. Ils n'activent pas les mécanismes de sécurité, ou utilisent des modes de sécurité faibles (mots de passe triviaux par exemple).

4 Vers des réseaux domestiques sécurisés

Les solutions traditionnelles utilisées pour les réseaux informatiques ne peuvent donc être utilisées simplement pour sécuriser les réseaux domestiques, qui présentent des particularités incompatibles avec les pré-requis des mécanismes à implémenter.

4.1 Compromis

La mise en œuvre de la sécurité dans les réseaux domestiques doit être imposée, faute de quoi les utilisateurs ne l'activeront pas. À titre d'exemple, citons WEP qui malgré ses défauts apporte une sécurité minimale, et qui est pourtant rarement déployé.

Contrairement à d'autres services, la configuration de la sécurité ne peut pas être entièrement automatisée : les tâches d'autorité doivent être prises en charge par l'utilisateur, qui est le seul à pouvoir décider des dispositifs appartenant à son réseau domestique.

Jusqu'à présent, nous avons supposé que l'utilisateur n'était jamais impliqué dans la configuration du réseau et de sa sécurité. Cependant, un compromis entre la facilité d'utilisation et la sécurité est indispensable.

Par opposition aux réseaux domestiques, les *réseaux domestiques sécurisés* doivent :

- Supporter une grande hétérogénéité.
- Autoriser des comportements extrêmement dynamiques.
- *Assurer par défaut un certain niveau de sécurité.*
- Fonctionner sans administration, *à l'exception de l'expression de l'autorité par l'utilisateur.*

De la même manière qu'ils déchargent l'utilisateur de l'installation des dispositifs et de leur configuration sur le réseau, les réseaux domestiques sécurisés doivent décharger les utilisateurs des manipulations fastidieuses concernant la mise en œuvre de leur sécurité : la tâche d'administration est réduite à une tâche d'autorité. Du point de vue de la frontière, l'utilisateur doit simplement indiquer les dispositifs qui font partie du réseau domestique, et ne doit pas être impliqué dans la manipulation de matériel cryptographique, de fonctions de chiffrement, de mots de passe complexes, etc.

Plusieurs travaux ont déjà été publiés dans le domaine des réseaux *ad hoc* et de l'*Ubiquitous Computing* ayant pour but de mettre en œuvre des mécanismes de sécurité en ne requérant qu'une très faible implication de l'utilisateur.

4.2 Premiers résultats

Frank Stajano expose dans [1] la problématique de la sécurité pour l'*Ubiquitous Computing* et l'état de l'art des propositions de sécurité. Il décrit notamment le modèle du "*Resurrecting Duckling*" [14] qui permet de mettre aisément en place des relations sécurisées point-à-point temporaires.

Dans ce modèle, chaque dispositif peut être alternativement dans deux états différents : vierge ou marqué. Un dispositif B passe de l'état vierge à l'état marqué lorsqu'un autre dispositif A le "marque de son empreinte" et en prend le contrôle. Pendant cette opération, A transmet à B *via* un canal sûr (par exemple un contact physique, lorsque l'utilisateur place les deux dispositifs l'un contre l'autre) ses données cryptographiques, qui permettront par la suite de communiquer de manière sécurisée. Tant qu'il reste marqué, B obéit à A, et ne peut plus être marqué par aucun autre dispositif. Le passage de l'état marqué à vierge se fait lorsque A "libère" B et lui ordonne de se ré-initialiser. B regagne alors son état initial, et sa relation temporaire avec A est dissoute.

Dans le modèle initial, les relations sont strictement point-à-point. En particulier, seul l'appareil ayant marqué l'autre peut lui restituer son état vierge, ce qui renforce légèrement la sécurité au prix d'une contrainte d'ergonomie.

Dans [12], Stajano étend le modèle du "*Resurrecting Duckling*"en permettant à un dispositif A marquant un dispositif B de lui transmettre optionnellement une politique de sécurité en même temps que les informations cryptographiques. Ainsi, il peut par exemple l'autoriser à échanger des informations avec d'autres dispositifs, et éventuellement à accepter des politiques de sécurité venant d'autres dispositifs.

Divers travaux se sont inspirés du "*Resurrecting Duckling*", dans le domaine des réseaux *ad hoc* notamment. Balfanz et al. [4] s'en servent pour garantir la sécurité des communications entre des dispositifs devant interagir ponctuellement (par exemple, un ordinateur portable et une imprimante "publique" dans un aéroport). Ils proposent notamment différents algorithmes permettant la transmission du matériel cryptographique en utilisant un canal de communication présentant des propriétés plus faibles que le "*Resurrecting Duckling*". Par exemple, l'un d'entre eux ne requiert pas que le canal soit confidentiel.

Feeney et al. [15] proposent quant à eux d'utiliser des canaux de communication localisés pour sécuriser les "réseaux spontanés". Un réseau spontané est formé de dispositifs (typiquement des ordinateurs portables et des assistants numériques personnels) mis en présence temporairement en vue d'activités collaboratives. Pour assurer l'authenticité et la confidentialité des communications entre ces dispositifs, les participants s'échangent au début de la réunion (par exemple par infrarouge) une clé de session générée aléatoirement. Comme ils ne manipulent jamais directement de matériel cryptographique et n'en ont même pas connaissance, la mise en œuvre de la sécurité ne nuit pas à la facilité d'utilisation.

Le gestionnaire de sécurité (*security manager*) de Bluetooth est une autre approche, plus industrielle, de la prise en charge de la sécurité par les dispositifs. Chaque module Bluetooth dispose d'un gestionnaire de sécurité qui libère l'utilisateur de la plus grande partie de la gestion des données cryptographiques. Il assure tout d'abord le contrôle d'accès au dispositif local et à ses ressources. Tout autre dispositif souhaitant y accéder doit connaître une information secrète (d'une longueur maximale théorique de 16 octets, mais en pratique souvent réduite à un code PIN de 4 chiffres). Le gestionnaire de sécurité vérifie que le dispositif à l'origine de la requête connaît ce secret, et, le cas échéant, lui permet d'accéder à la ressource.

Le gestionnaire de sécurité se charge aussi de sécuriser les communications avec les autres dispositifs Bluetooth, en générant avec chaque dispositif une clé partagée obtenue par dérivation de l'information secrète (i.e. le code PIN) nécessaire pour accéder à la ressource. Cette clé servira à assurer l'intégrité et la confidentialité des messages.

Enfin, le gestionnaire de sécurité mémorise les opérations de sécurité (contrôle d'accès et mise en place de clé partagée) déjà réalisées. Ainsi, l'utilisateur d'un dispositif A, demandant à accéder à plusieurs reprises à un service fourni sur le dispositif B, ne rentre le code PIN requis par le gestionnaire de sécurité de B qu'une fois. Par la suite, B sait que A est autorisé à accéder au service.

Toutes ces propositions visent la mise en œuvre d'une sécurité transparente pour l'utilisateur. Toutefois, il n'existe pour l'instant aucune proposition permettant de mettre en œuvre une frontière telle que nous l'avons définie. Une recherche plus poussée est donc nécessaire.

5 Conclusion

Il est maintenant clair que lorsque les réseaux domestiques seront démocratisés, ils seront victimes d'attaques multiples :

- Héritées des attaques informatiques classiques concernant la partie IP / PC.
- Liées à des conditions de voisinage concernant les communications sans fil.
- Liées à la valeur des contenus multimédias et à leur manipulation dans un contexte *consumer electronics*.
- Liées à l'utilisation des appareils et des services par des non-administrateurs.

Ils doivent donc être sécurisés pour en faciliter l'adoption. L'établissement d'une frontière délimitant le réseau domestique est la première étape d'une sécurisation raisonnée et acceptable.

Les solutions de sécurité éprouvées en contexte professionnel ne sont pas adaptées en contexte domestique, soit parce qu'elles sont inopérantes en l'état (*firewall* IP), soit

parce qu'elles contraignent trop fortement le réseau (serveur centralisé), soit encore parce qu'elles requièrent un utilisateur expert.

Il s'agit donc de mettre en place un mécanisme permettant d'établir et de maintenir la frontière de manière transparente à l'utilisateur.

Plusieurs travaux ont déjà proposé des mécanismes permettant de mettre aisément en œuvre des relations sécurisées entre différents dispositifs. Toutefois, ces relations sont le plus souvent point-à-point, et/ou temporaires. À notre connaissance, aucun résultat n'a été publié sur la gestion de groupes dynamiques à long terme.

Dans cet esprit, nous menons actuellement des recherches pour permettre la mise en place distribuée d'une frontière garantissant la cohérence du réseau domestique après des opérations de fractionnement et de fusion, et autorisant la révocation d'appareils volés ou compromis, tout en tenant compte de la facilité d'accès à ces fonctions par des non-administrateurs.

Remerciements

Les auteurs tiennent à remercier Jean-Pierre Andreaux, Eric Diehl et Valérie Gayraud pour leurs contributions précieuses. Ils tiennent aussi à remercier les relecteurs, et particulièrement Nicolas Fischbach, pour leurs remarques pertinentes et constructives.

Références

1. Frank Stajano, *Security for Ubiquitous Computing*, John Wiley and Sons, 2002, *ISBN 0-470-84493-0, http ://www-lce.eng.cam.ac.uk/ fms27/secubicomp/*
2. The Honeynet Project, *Know Your Enemy*, Addison Wesley, 2001.
3. *www.@stake.com*, @stake security advisory, Nokia 6210 DoS SMS Issue, 2003, *http ://www.atstake.com/research/advisories/2003/index.html#022503-1*
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, 2002, *citeseer.nj.nec.com/balfanz02talking.html*
5. William A. Arbaugh, William L. Fithen et John McHugh, *Windows of Vulnerability : A Case Study Analysis*, IEEE Computer", IC-33, no 12, pp 52-59, 2000.
6. Chris Davis et Aaron Higbee, *DC Phone Home*, BlackHat Conference, 2002.
7. Gordon Bell et Jim Gemmell, *A call for the home media network*, Communications of the ACM, Vol. 45, no. 7, 2002, pp. 71–75, *http ://doi.acm.org/10.1145/514236.514237*, ACM Press.
8. R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis et M. Weiser", *An overview of the* `PARCTAB` *ubiquitous computing experiment*, IEEE Personal Communications, Vol. 2, no. 6, pp. 28-33, 1995, *citeseer.nj.nec.com/want95overview.html*
9. Médiamétrie, *Les Baromètres Multimédia*, Février 2003, *http ://www.mediametrie.fr/web/resultats/barometre/resultats.php ?id=632*
10. J. Jeff, Y. Alan, R. Ross et A. Alasdair, *The Memorability and Security of Passwords – Some Empirical Results.* Technical Report No. 500, Computer Laboratory, University of Cambridge, 2000, *http ://www.ftp.cl.cam.ac.uk/ftp/users/rja14/tr500.pdf*

11. Mark Weiser, *The computer for the 21st century*, ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 3, no. 3, 1999, pp. 3–11, *http ://doi.acm.org/10.1145/329124.329126*, ACM Press.
12. Frank Stajano, *The Resurrecting Duckling – What Next ?*, Lecture Notes in Computer Science, Vol. 2133, pp. 204–211, 2001, *citeseer.nj.nec.com/stajano00resurrecting.html*
13. Aleph One, *Smashing the Stack for Fun and Profit*, Phrack, Vol. 49, 1996, *http ://www.phrack.org/show.php ?p=49&a=14*
14. Frank Stajano et Ross Anderson, *The Resurrecting Duckling : Security Issues for Ad-hoc Wireless Networks*, 7th International Workshop on Security Protocols, pp. 172–194, 1999, *citeseer.nj.nec.com/stajano99resurrecting.html*
15. L. Feeney, B. Ahlgren et A. Westerlund, *Spontaneous networking : an application-oriented approach to ad hoc networking*, IEEE Communications Magazine, June 2001, *citeseer.nj.nec.com/feeney01spontaneous.html*
16. Bruce Schneier, *Secrets and Lies : Digital Security in a Networked World*, John Wiley & Sons, 2000.
17. CERT Coordination Center, Carnegie Mellon University, *CERT/CC Overview Incident and Vulnerability Trends*, 2002, *http ://www.cert.org/present/cert-overview-trends/*

Poste de travail léger sécurisé

Laurent CABIROL

Commissariat à l'Énergie Atomique,
Direction des Technologies de l'Information,
Bat 474 91191 Gif sur Yvette cédex
laurent.cabirol@cea.fr

Résumé Le Commissariat à l'Énergie Atomique se propose de déployer dans ses centres de recherche des applications nationales très fortement sécurisées dans un mode client-serveur. Après un rappel de la problématique, l'exposé décrira les règles qu'il s'impose pour la protection des informations sensibles manipulées par ces applications et les menaces contre lesquels il se propose de se protéger. Nous passerons ensuite à une description sommaire de la solution retenue basée sur un client léger Linux sans disque dur et avec lecteur de carte à puce utilisant un VPN pour communiquer avec le serveur. L'exposé s'attardera sur les mécanismes de boot, puis de chargement de système et d'applications en présentant différents choix possibles de sécurisation selon les étapes.

1 Contexte du déploiement d'applications sécurisées

Le CEA est un établissement public de recherche dont les laboratoires sont répartis au sein d'une douzaine d'établissement différent sur le territoire national. Ses recherches couvrent un champ assez vaste, des travaux nécessaires à la maîtrise de la production d'énergie nucléaire à la recherche fondamentale, notamment dans les domaines de la physique théorique et des sciences du vivant, en passant par des actions nombreuses de recherche technologique, par exemple dans le domaine des matériaux, de la micro électronique ou des nanotechnologies, sans oublier les travaux nécessaires au maintien de la capacité de dissuasion nationale. Le CEA regroupe environ 15000 salariés auxquels s'ajoutent de nombreux thésards, stagiaires et collaborateurs extérieurs (*http ://www.cea.fr/*).

Son parc informatique comprend environ 18000 postes de travail bureautique, de très nombreux serveurs de puissance variés et un centre de calcul à hautes performances. Touts ces ordinateurs sont reliés par un réseau très performant connecté à Internet via le réseau RENATER.

Pour le déploiement d'applications nationales (comme son système de gestion par exemple), le CEA a choisi d'utiliser un modèle d'architecture client-serveur et a établi des règles formalisées dans un recueil normatif interne (Normacs). Le CEA fait évoluer ce modèle pour prendre en compte les évolutions techniques (J2E, web services, portail, etc...).

Le déploiement d'applications sécurisées devra impérativement prendre en compte ce contexte dans une optique d'intégration plus rapide au moindre coût de ces applications au sein du système d'information du CEA.

2 Les menaces supplémentaires

Nous prenons pour hypothèse de travail que l'environnement informatique et réseau standard du CEA est à un niveau de sécurité conforme à l'état de l'art.

Par rapport aux applications standards déjà déployées, les applications fortement sécurisées doivent proposer des fonctionnalités supplémentaires pour prendre en compte de nouvelles menaces. Celles-ci sont détaillées dans la suite de ce papier. Nous ne décrivons pas les améliorations à faire ou à prendre en compte du coté "serveurs". Nous faisons l'hypothèse de travail qu'ils bénéficient d'une protection physique suffisante et de personnels d'administration sûrs.

2.1 Confidentialité des échanges

Les échanges entre les postes de travail et les serveurs centraux utilisent le réseau CEA. Même si le personnel intervenant au CEA fait l'objet de contrôle strict de sécurité, l'application plus rigoureuse de la règle du "*besoin d'en connaître*" impose une séparation du trafic de façon à ne pas permettre à un éventuel intrus de "sniffer" les échanges et de disposer d'informations qu'il n'aurait pas à connaître. Pour ne pas avoir à déployer un second réseau physique, il devient donc nécessaire de chiffrer les échanges entre client et serveur.

2.2 Authentification forte des utilisateurs des applications sécurisées

L'application stricte de la règle du "*besoin d'en connaître*" nous conduit également à vouloir authentifier de façon sûre l'utilisateur d'une application sécurisée. En effet, l'usage classique d'une authentification par "*login/password*" n'est pas jugé suffisamment forte dans ce contexte.

2.3 Protection physique du poste client

Les postes clients en ce qu'ils permettent l'accès aux applications sécurisées sont eux- même sensibles. Ils peuvent être volés afin de récupérer les informations qu'ils contiennent, que ces informations proviennent des applications elles-mêmes ou permettent par leur connaissance de les attaquer.

Il convient donc de protéger les postes de travail pour se prémunir de ce type d'attaque. La méthode classique consiste à établir un ou plusieurs périmètres de protection physique autour des ressources sensibles. Le coût de cette protection étant très élevé, il est nécessaire de le diminuer (et si possible de s'en affranchir) et donc de trouver des moyens de rendre le vol d'un poste de travail sans conséquence.

2.4 Intrusion dans le poste client

Disposer d'un poste client permet de l'étudier en détail et d'en déduire des informations qui permettrait de diminuer la sécurité du système. Des mesures prises en conséquence seront détaillées au chapitre suivant.

3 Éléments de solution

3.1 Authentification des utilisateurs

Le CEA dispose d'une infrastructure de gestion de clés et est à même de délivrer à ses personnels un certificat X509 stocké dans une carte à puce. Ce mode d'authentification sera donc imposé pour l'accès aux applications sécurisées. Le renforcement de l'authentification, et donc de la sécurité, vient de la nécessité de disposer d'un objet (la carte à puce) et de connaître le pin-code associé pour pouvoir accéder à ces applications.

3.2 Confidentialité des échanges

Pour répondre à la contrainte de chiffrement des échanges, nous mettons en place un VPN IPSEC. L'échange initial de clés se fera sous le contrôle du certificat X509 de l'utilisateur contenu dans sa carte à puce.

3.3 Protection physique, vol du poste de travail

Pour cela, nous nous imposerons de ne pas avoir de données rémanentes de l'application sur le poste de travail et donc de ne pas avoir de disque dur ou de dispositif de stockage amovible (disquettes, dongles USB, etc.).

3.4 Intrusion dans le poste client

Pour limiter les conséquences de la "possession" par un intrus d'un poste client sur la sûreté du système, des mesures anti-intrusion sont bien sûr mises en place sur le boîtier même du poste.

Nous nous sommes imposés de ne pas avoir de données rémanentes de l'application. Il faut cependant pouvoir « booter » le poste de travail de façon sûre. Deux possibilités s'offrent à nous : télécharger l'application de façon sécurisée ou démarrer le poste de travail à partir d'un dispositif non-réinscriptible (cd-rom). C'est ce dernier choix que nous avons fait. Il entraîne bien sûr des mesures supplémentaires pour la protection du dispositif qui stocke l'application, le système d'exploitation et ses données de configuration.

Dans ce contexte, il paraît aussi souhaitable de contrôler l'utilisation du poste client au plus tôt dans le processus de démarrage pour éviter les possibilités d'intervention "malicieuses" lors de cette phase sensible.

Nous choisirons ici de démarrer le poste de travail sous le contrôle d'une carte à puce d'administrateur. Cela permet de séparer ce rôle de celui de l'utilisateur. Afin que le vol du poste ne permette pas de récupérer d'informations sensibles sur sa configuration, nous avons choisi de chiffrer celles-ci sur le cd-rom. Le déchiffrement se fait sous le contrôle de la carte à puce d'administrateur. Une fois le poste démarré, l'utilisateur peut alors ouvrir une session à l'aide de sa carte personnelle (voir figure 1).

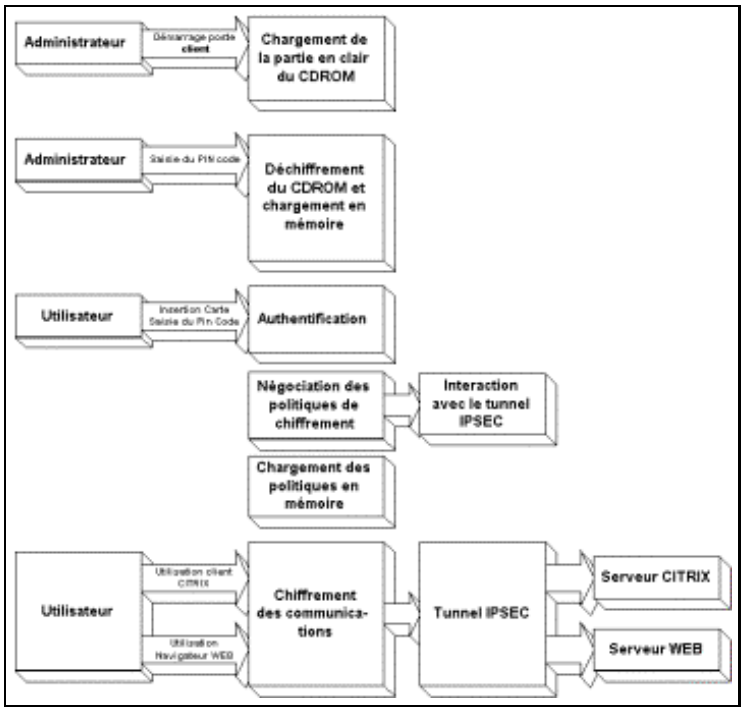


Fig. 1. Démarrage du poste de travail

4 Choix techniques

4.1 Matériel

Le poste de travail est constitué d'un PC standard auquel sont ajoutés un dispositif d'anti-intrusion, un lecteur de carte à puce avec clavier intégré. Cela permet de limiter le coût du matériel.

4.2 Logiciel

Le logiciel est constitué d'un système d'exploitation Linux configuré avec les seuls modules utiles et compilé spécifiquement pour ce PC après application des patchs de renforcement de la sécurité. Le mécanisme d'initialisation est spécifiquement modifié pour prendre en compte une authentification par carte à puce d'un administrateur avant le démarrage des services "réseau". Deux versions sont créées, l'une avec un client CITRIX, l'autre avec un navigateur standard.

4.3 Processus de démarrage du PC

Le mécanisme de boot du PC n'a pas été modifié. Le souhait de pouvoir contrôler l'utilisation du poste client au plus tôt dans le processus de démarrage (dès le boot) a conduit à examiner plusieurs choix possibles détaillés ci-après.

L'authentification au plus tôt lors du démarrage du PC impose de pouvoir ajouter des fonctionnalités supplémentaires comme la possibilité de dialoguer avec une carte à puce, de déchiffrer les informations de configurations du poste de travail, etc. Cela conduit à souhaiter l'extension soit du BIOS, soit du chargeur.

4.4 Modification de la ROM du BIOS du PC

Plusieurs possibilités existent pour remplacer le BIOS "standard" d'un PC, à savoir :

- Openfirmware,
- Linuxbios,
- Freebios.

La forte dépendance au matériel (chipset) de ces solutions ne permet pas de les envisager sans un effort jugé trop important à ce stade de notre projet.

4.5 Modification d'un chargeur

Les nombreux chargeurs du monde "propriétaires" n'ont pas été regardés pour des raisons évidentes de non-disponibilité des sources.

Dans le monde "libre", trois chargeurs principaux se partagent le "marché", à savoir :

- Syslinux/isolinux,
- LILO,
- GRUB.

La modification d'un chargeur pour des raisons équivalentes à la modification du BIOS n'a pas été retenue.

4.6 Modification de la procédure d'initialisation LINUX

Cette procédure a été modifiée afin de permettre l'authentification de la carte "administrateur" pendant son déroulement, et ainsi permettre le déchiffrement des informations confidentielles spécifiques du poste de travail et du réseau. Sont ensuite lancé directement le client nécessaire ainsi que le tunnel IPSEC après l'authentification réussi de l'utilisateur au moyen de son certificat X509 stocké sur sa carte à puce.

4.7 Quelques mesures de sécurité complémentaires

D'autres mesures ont été retenues de façons à rendre plus difficile le succès d'une attaque du poste client. Elles ne seront pas exposées. A titre d'exemple cependant, un mécanisme de couplage/identification réciproque du CD, du PC et de la carte administrateur a été mis en place.

5 Quelques exemple de scenarii malveillants

5.1 Vol du CD de démarrage

Les informations réellement importantes sont chiffrées sur le CD et donc inaccessibles. Le reste est du domaine public.

5.2 Démarrage du PC avec un autre CD

Cela nécessite l'ouverture du PC et donc la mise en œuvre des mesures anti-intrusion matérielles (blocage du boot du PC). Quand bien même, celles-ci seraient mise en défaut, l'attaquant devrait alors mettre en défaut le boîtier VPN protégeant les serveurs et serait repéré par le système de détection d'intrusion coté "serveurs".

5.3 Vol d'une carte "administrateur"

Il faudrait voler en plus le pin code de la carte. L'administrateur n'a pas pour autant l'autorisation de se connecter aux applications, il faudrait aussi voler une carte utilisateur et son pin code. Par ailleurs, il n'y a pas de "*shell*" sur le poste de travail, juste le client CITRIX ou le client WEB.

6 Conclusions

Les mesures rapidement décrites précédemment permettent de se protéger :
- de l'utilisation par des personnels non autorisés,
- de l'écoute sur le réseau,
- de l'intrusion sur le poste de travail,
- de limiter les conséquences d'un vol du poste de travail.

L'utilisation de composants "libres" permet de construire le poste client léger souhaité à moindre coût en prenant en compte les contraintes de sécurité que nous nous sommes fixées.

Références

1. BIOS
   http ://www.openfirmware.org/
   http ://www.linuxbios.org/
   http ://freebios.sourceforge.net/

2. Loader
   http ://syslinux.zytor.com/
   http ://brun.dyndns.org/pub/linux/lilo/
   http ://www.gnu.org/software/grub/
   bibitemlinux Linux
   http ://diet-pc.sourceforge.net/
   http ://www.linuxfromscratch.org/

3. Cartes à puce et Linux
   http ://www.linuxnet.com/

Détection d'intrusions dans les résaux *ad hoc*

Jean-Marc Percher[1], Bernard Jouga[2]

[1] Ecole Supérieure d´Electronique de l'Ouest (ESEO),
4, rue Merlet de la Boulaye,
49000 Angers, France
jean-marc.percher@eseo.fr
[2] Supélec, BP 81127,
35511 Cesson Sévigné cedex, France
bernard.jouga@supelec.fr

Résumé Les réseaux sans fil ad hoc ou MANET, *Mobile Ad hoc NETwork*, sont des réseaux dont la topologie ne bénéficie d'aucune infrastructure préexistante. Elle se forme au gré de l'apparition et du mouvement des nœuds. Les participants d'une réunion, les intervenants des opérations de secours menées sur un site en cours d'exploration, les éléments engagés sur un champ de bataille peuvent tirer profit des caractéristiques de tels réseaux pour échanger de l'information. L'évolution rapide des performances des réseaux locaux sans fil et leur utilisation de plus en plus importante par les utilisateurs mobiles devraient bénéficier au développement des MANET. Si les MANET se différencient des réseaux classiques, cellulaires ou filaires, par les caractéristiques de leur topologie, les services demandés au réseau par les utilisateurs restent identiques, notamment en matière de sécurité. Dans cet article nous proposons une architecture de sécurité pour les réseaux ad hoc. Les mécanismes de sécurité sont renforcés par un système de détection d'intrusions distribué et coopératif. Chaque nœud est équipé d'un IDS local et des agents mobiles autonomes sont mis en œuvre, si nécessaire, pour collecter les informations stockées sur les autres nœuds. Nous validons cette architecture d'IDS, *Intrusion Detection System*, construite autour d'une plate-forme à agents mobiles à l'aide d'un prototype et de tests en laboratoire.

1 Introduction

Nous proposons pour les réseaux *ad hoc* la définition suivante : "Un réseau ad hoc est un réseau sans fil capable de rendre transparentes aux utilisateurs mobiles les modifications de topologie qu'il subit". Le groupe MANET de l'IETF fournit une définition plus précise en introduction de la RFC 2501 [1] :

> *Un réseau ad hoc comprend des plates-formes mobiles (par exemple, un routeur interconnectant différents hôtes et équipements sans fil) appelées nœuds qui sont libres de se déplacer sans contrainte. Un réseau ad hoc est donc un système autonome de nœuds mobiles. Ce système peut fonctionner d'une manière isolée ou s'interfacer à des réseaux fixes au travers de passerelles. Dans ce dernier cas, un réseau ad hoc est un réseau d'extrémité.*

Il n'en reste pas moins que la terminologie "réseau ad hoc" est relativement peu explicite. C'est sans doute la raison pour laquelle la communauté scientifique la remplace parfois par celle de "réseau spontané", traduction de *spontaneous network*.

A partir de cette définition générale, il est intéressant de mettre en avant les caractéristiques principales qui différencient un réseau ad hoc d'un réseau classique.

- *Mobilité.*- La mobilité des noeuds constitue à l'évidence une caractéristique très spécifique des réseaux ad hoc. Cette mobilité est intrinsèque au fonctionnement du réseau. Elle se distingue de la nomadicité (mobilité des seuls noeuds terminaux) ou de l'itinérance (équipements statiques mais pouvant être déplacés). Dans un réseau ad hoc, la topologie du réseau peut changer rapidement, de façon aléatoire et non prédictible et les techniques de routage des réseaux classiques, basées sur des routes préétablies, ne peuvent plus fonctionner correctement.
- *Equivalence des nœuds du réseau.*- Dans un réseau classique, il existe une distinction nette entre les nœuds terminaux (stations, hôtes) qui supportent les applications et les nœuds internes (routeurs par exemple) du réseau, en charge de l'acheminement des données. Cette différence n'existe pas dans les réseaux ad hoc car tous les noeuds peuvent être amenés à assurer des fonctions de routage.
- *Liaisons sans fil.*- Les technologies de communication sans fil sont indispensables à la mise en place d'un réseau ad hoc. Malgré des progrès très importants, leurs performances restent et resteront en deçà de celles des technologies des réseaux filaires. La bande passante est moins importante, alors que le routage et la gestion de la mobilité génèrent davantage de flux de contrôle et de signalisation que dans une architecture de réseau filaire. Ces flux doivent être traités de façon prioritaire pour prendre en compte rapidement les modifications de topologie.
- *Autonomie des nœuds.*- La consommation d'énergie constitue un problème important pour des équipements fonctionnant grâce à une alimentation électrique autonome. Ces équipements intègrent des modes de gestion d'énergie et il est important que les protocoles mis en place dans les réseaux ad hoc prennent en compte ce problème.
- *Vulnérabilité.*- Les réseaux sans fil sont par nature plus sensibles aux problèmes de sécurité. Pour les réseaux ad hoc, le principal problème ne se situe pas tant au niveau du support physique mais principalement dans le fait que tous les nœuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau. Les possibilités de s'insérer dans le réseau sont plus grandes, la détection d'une intrusion ou d'un déni de service plus délicate et l'absence de centralisation pose un problème de remontée de l'information de détection d'intrusions.

L'article est organisé comme suit : la section 2 présente les caractéristiques des protocoles de routage ad hoc. La section 3 présente les caractéristiques des systèmes de détection des intrusions et les principaux IDS distribués. La section 4 décrit les caractéristiques de l'architecture proposée. La section 5 présente une partie des tests fonctionnels réalisés. Enfin nous concluons et proposons des suites possibles aux travaux réalisés.

2 Le routage dans les réseaux ad hoc

2.1 Taxonomie des protocoles de routage pour les réseaux Ad Hoc

Les protocoles de routage des réseaux ad hoc s'appuient sur deux modèles de fonctionnement : les protocoles proactifs et les protocoles réactifs. On peut les différencier

par la méthode utilisée pour découvrir le chemin entre le nœud source et le nœud destination. Pour maintenir leur table de routage, les protocoles proactifs recherchent à intervalle régulier les différentes routes disponibles dans le réseau. Quand un paquet doit être transmis, sa route est alors connue à l'avance et peut ainsi être immédiatement utilisée. Les protocoles réactifs entreprennent la recherche d'une route uniquement avant de transmettre un paquet. La figure 1 présente une taxonomie des protocoles de rou-

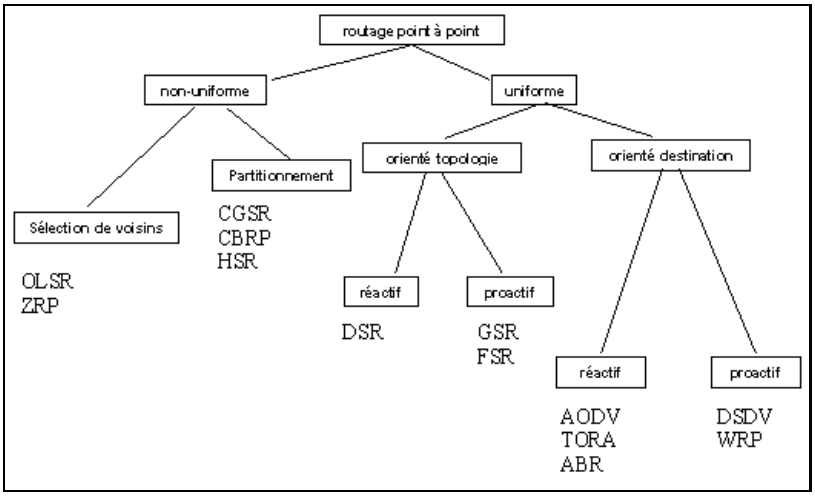


Fig. 1. Taxonomie des protocoles de routage pour les réseaux ad hoc.

tage pour les réseaux ad hoc. Ces protocoles se différencient d'abord par le niveau d'implication des nœuds dans le routage. Ils sont dits *uniformes* si tous les nœuds du réseau jouent le même rôle pour la fonction de routage. Ils peuvent à l'inverse être *non-uniformes* si une structure hiérarchique est donnée au réseau et que seuls certains nœuds assurent le routage. Ainsi, dans les protocoles à *sélection de voisins*, chaque nœud sous-traite la fonction de routage à un sous ensemble de ses voisins directs. Pour les protocoles à *partitionnement*, le réseau est découpé en zones dans lesquelles le routage est assuré par un unique nœud maître.

Les protocoles de routage uniformes peuvent également être regroupés selon les données qu'ils utilisent pour effectuer leur tâche. Dans les protocoles *orientés topologie*, plus connus sous le nom de *link-state protocols*, chaque nœud utilise comme données l'état de ses connexions avec ses noeuds voisins ; cette information est ensuite transmise aux autres nœuds pour leur donner une connaissance plus précise de la topologie du réseau. Les protocoles *orientés destinations*, plus connus sous le nom de *distance vector protocols*, maintiennent pour chaque nœud destination une information sur le nombre de nœuds qui les en séparent (la distance) et éventuellement sur la première direction à emprunter pour y arriver (le vecteur).

Avec un protocole proactif, les routes sont disponibles immédiatement. Cependant, le trafic induit par les messages de contrôle et de mise à jour des tables de routage peut être important et partiellement inutile. De plus, la taille des tables de routage croit

linéairement en fonction du nombre de nœuds. A l'opposé, dans le cas d'un protocole réactif, aucun message de contrôle ne charge le réseau pour des routes inutilisées. Mais, pour ces derniers, la mise en place d'une route par inondation peut être coûteuse et provoquer des délais importants avant l'ouverture de la route.

En terme de performances, les protocoles orientés topologie (*link state*) convergent plus rapidement que les protocoles orientés destination (*distance vector*). Cependant, dans le cas de réseaux à forte mobilité, le trafic induit par les fréquents messages de contrôle est souvent pénalisant.

D'une façon générale, les protocoles de routage proactifs "plats", qu'ils soient orientés destination ou topologie, ne se sont pas adaptés aux réseaux de taille importante (nombre de nœuds supérieur à 100) et à forte mobilité. Une première solution pour ce type de réseau est l'utilisation de protocoles dits hiérarchiques (tels que HSR, FSR, etc.). Une seconde solution peut être d'utiliser un protocole réactif. Ce type de routage permet de gérer de très gros réseaux si la mobilité des nœuds reste faible; le trafic reste faible s'il est dirigé vers un nombre réduit de destinations. D'autre part, le calcul d'une route sur demande est très pénalisant pour du trafic multimédia demandant des garanties en matière de qualité de service.

Des études comparatives montrent que certains protocoles sont plus performants que d'autres selon les caractéristiques du réseau. Par exemple, AODV, DSR ou GSR sembleraient convenir à des réseaux à forte mobilité, tandis que TORA est mieux adapté à des réseaux plus statiques. Par ailleurs, TORA et OLSR gèrent efficacement les grands réseaux à forte densité, AODV semble quant à lui performant dans les réseaux de faible densité, et DSR est bien adapté aux petits réseaux.

A l'heure actuelle, aucun protocole de routage dans les réseaux ad hoc n'a été adopté au sein de l'IETF. Les différents protocoles présentés sur la figure 1 ne sont encore que des *drafts* et restent en cours de développement et/ou de spécification. Certaines propositions ont été abandonnées et les plus abouties sont AODV, DSR et OLSR. Les deux premiers protocoles sont parmi les plus anciens et ils ont fait l'objet de simulations comparatives détaillées [2].

2.2 Exemple du protocole de routage OLSR

OLSR, *Optimized Link State Routing Protocol*, est un protocole proactif, non uniforme et basé sur la sélection de voisins [3]. OLSR s'appuie sur le concept de Relai Multi Point (*Multi Point Relay*, MPR). Les MPR d'un nœud correspondent à l'ensemble des voisins qui permettent d'atteindre tous les nœuds situés à deux sauts. La diffusion des différents messages de contrôle ne se fait que vers les MPR (voir la figure 2), réduisant ainsi les répétitions inutiles. D'autre part OLSR distingue les liens unidirectionnels des liens bidirectionnels, seuls utilisés pour le routage. Chaque nœud maintient de l'information sur les nœuds qui l'ont élu en tant que MPR. Ceci est fait grâce à des message de présence (*Hello messages*) envoyés par chaque nœud à ses voisins. Ces messages contiennent :

- la liste des nœuds avec lesquels l'émetteur a des liens bidirectionnels,
- la liste des nœuds que l'émetteur peut entendre (ils entretiennent un lien unidirectionnel vers lui)
- la liste des nœuds que l'émetteur a choisi comme MPR.

La diffusion de ces messages de présence permet aux nœuds du réseau de stocker, dans leur table des voisins, une vision à deux sauts de leur voisinage et de calculer l'ensemble de leurs MPR. Cet ensemble est recalculé dès qu'un changement est détecté dans le voisinage à deux sauts.

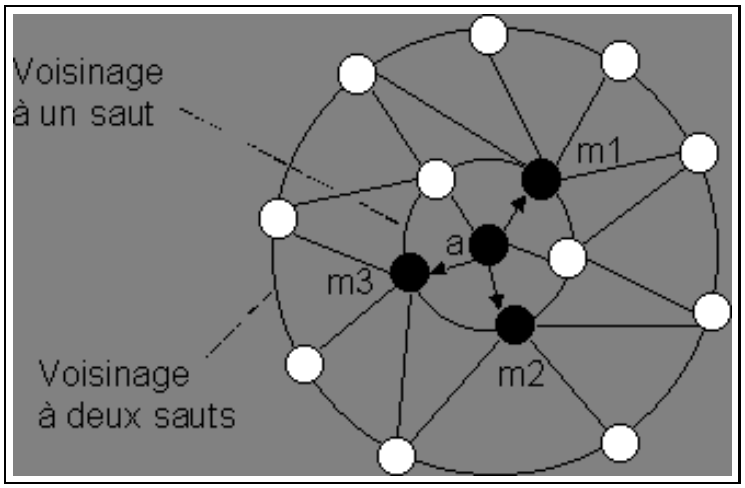


Fig. 2. Relais Multi Point. La station A a choisi m_1, m_2 et m_3 comme relais multi point. Quand A émet un message TC (*Topology Control*), il est seulement retransmis par m_1, m_2 et m_3, qui le retransmettent à leur tour vers leur MPR.

La diffusion sur la totalité du réseau (via les MPR) de messages de contrôle de topologie (*Topology Control messages, TC messages*) donne l'information topologique nécessaire au routage. Ces messages contiennent, pour chaque MPR, la liste des nœuds qui l'ont choisi. Grâce à ces messages, les nœuds peuvent maintenir une table de topologie (*Topology Table*), indiquant le dernier saut pour chaque destination.

Un algorithme de plus court chemin, appliqué à la table des voisins et à la table de topologie, permet de construire la table de routage de chaque nœud. Cette table mémorise, pour tous les nœuds du réseau, le nombre de sauts et le premier saut pour l'atteindre. Elle doit être recalculée dès que l'une des deux tables sources est modifiée.

OLSR fournit des routes optimales en nombre de sauts. Il convient pour des grands réseaux grâce à son mécanisme de MPR, mais est sans doute moins efficace pour de petits réseaux.

3 Détection d'intrusions dans les réseaux ad hoc

3.1 Le constat

Les réseaux sans fil sont par nature plus sensibles aux problèmes de sécurité. L'intrusion sur le support de transmission est plus facile que pour les réseaux filaires et il est possible de mener des attaques par déni de service en brouillant les bandes fréquences utilisées. Le contexte ad hoc augmente le nombre de failles de sécurité potentielles. Etant par définition sans infrastructure, les réseaux ad hoc ne peuvent bénéficier des services de sécurité offerts par des équipements dédiés : pare feux, serveurs d'authentification, etc. Les services de sécurité doivent être distribués, coopératifs et compatibles avec la bande passante disponible. Le routage pose aussi des problèmes spécifiques :

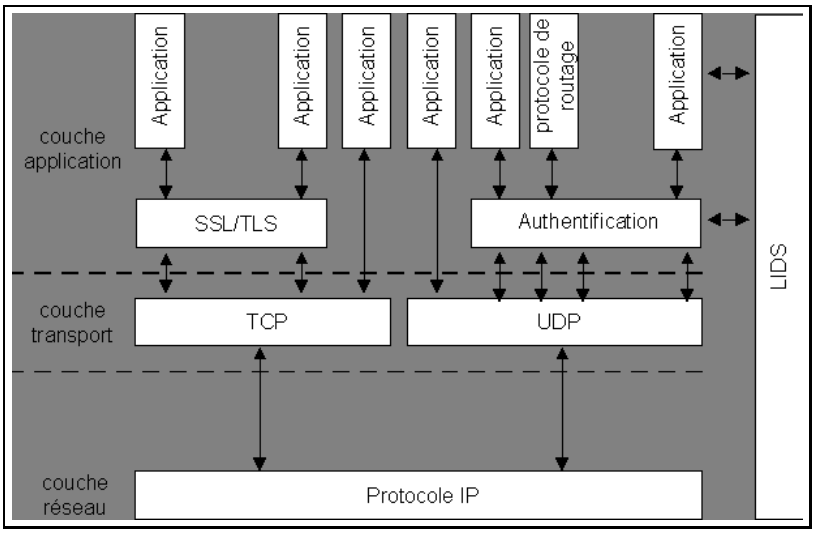


Fig. 3. Modèle de sécurité pour les réseaux ad hoc.

chaque station du réseau peut servir de relai et a donc la possibilité de capturer ou bien de détourner le trafic en transit. Des attaques en déni de service sont également possibles. Un routeur malicieux peut aussi générer des messages de routage dans le but de submerger les vrais routeurs. Nous avons mené dans [4,7] une étude sur les vulnérabilités du protocole de routage OLSR. Les quatre types d'attaques recensées reposent essentiellement sur l'émission de messages fallacieux par un nœud malveillant inséré dans le réseau ad hoc. Cette possibilité étant offerte par presque tous les autres protocoles de routage ad hoc, des attaques de même nature peuvent être développées. Les attaques décrites ont été implémentées et une plate- forme expérimentale a permis de jouer les attaques et de constater les effets attendus. Les signatures de deux de ces attaques ont été déterminées afin qu'elles puissent ensuite être détectées par un système de détection d'intrusions réparti.

Aujourd'hui la sécurisation des MANET s'appuie sur les mécanismes de sécurité du lien radio (WEP) et des échanges IP (Ipsec-VPN). Dans l'attente de solutions de sécurité adaptées aux caractéristiques des MANET comme, par exemple, les infrastructures d'authentification distribuées [5,6], la gestion des clés de codage doit être adaptée aux différents cas d'usage.

Nous avons proposé dans [4] une architecture de sécurité pour les réseaux ad hoc, présentée figure 3, dans laquelle les mécanismes de sécurité des MANET sont renforcés par un système local de détection des intrusions (*LIDS, Local Intrusion Detection System*).

3.2 Rappels sur les systèmes de détection d'intrusions

Nous considérons ici comme intrusion toute action visant à compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource. L'observation des attaques dirigées

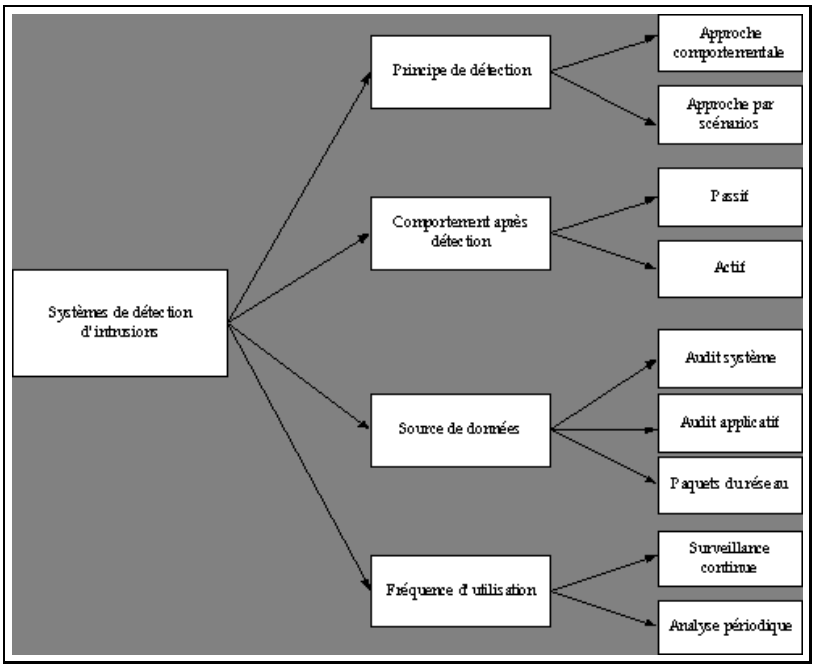


Fig. 4. Taxonomie des IDS.

vers les systèmes d'information nous montre que, quelles que soient les techniques de prévention mises en place contre les intrusions, il existe toujours des failles exploitables pour celui qui les traque. L'attaque d'un système peut même être réalisée simplement à partir de l'enchaînement d'opérations élémentairement autorisées ; elle ne nécessite donc pas toujours de contourner les mécanismes de sécurité. La détection d'intrusions peut donc être considérée comme une action complémentaire à la mise en place des mécanismes de sécurité. En effet, si une tentative d'intrusion est détectée suffisamment tôt, les réponses du système peuvent permettre de limiter les conséquences d'une attaque. Dans le cas d'une attaque de type déni de service, la réponse d'un système de détection d'intrusions (souvent abrégé IDS, traduction de *Intrusion Detection System*) pourrait être de refuser les nouvelles demandes de connexions au-delà d'une certaine fréquence estimée normale.

Pour présenter les caractéristiques des systèmes de détection d'intrusions nous retenons la classification proposée par L. Mé et C.Michel [8]. Celle-ci, représentée sur la figure 4, utilise les critères de classification suivants :

- Le principe de détection des intrusions,
- Le comportement après détection,
- La source des données,
- La fréquence d'utilisation.

Le principe de détection des intrusions. Les deux approches utilisées à ce jour sont l'approche comportementale et l'approche par scénarios. La première se base sur l'observa-

tion du comportement de l'utilisateur et sur la détection d'un comportement déviant par rapport à ses habitudes. Cette modification du comportement peut traduire une tentative d'intrusion, due soit à une usurpation de l'identité de l'utilisateur, soit à l'exécution par celui-ci de commandes non autorisées. La seconde approche consiste à identifier chaque attaque par une signature propre et ensuite à rechercher dans les fichiers d'audits du système les traces de ces signatures. On peut noter que cette approche par scénarios nécessite de connaître la signature d'une attaque avant de pouvoir la détecter. L'approche comportementale permet de détecter des attaques inconnues, mais la définition du comportement normal de utilisateur demeure la principale difficulté.

Le comportement après détection La nature de la réponse apportée par les systèmes après détection d'une intrusion peut aussi être utilisée pour classifier les IDS. Deux types de réponses sont implémentées à ce jour, soit une réponse passive, par exemple l'émission d'une alarme à l'administrateur, soit une réponse active, comme la mise en place de nouvelles règles de filtrage sur un pare-feu.

La source des données Les IDS peuvent être classés en fonction de la provenance de leurs données d'audit, selon qu'elles viennent du système, des applications, des paquets du réseau ou encore d'un autre IDS.

La fréquence d'utilisation La fréquence d'analyse des données d'audit est aussi un élément distinctif des systèmes de détection d'intrusions. Certains IDS peuvent surveiller en permanence le système d'information tandis que d'autres se limitent à une analyse périodique.

3.3 Motivation des travaux

Les techniques de détection d'intrusions utilisées dans les réseaux traditionnels ne peuvent être exploitées telles quelles dans les réseaux ad hoc. Comparées aux architectures des réseaux filaires où la supervision du trafic est généralement réalisée au niveau des commutateurs, des routeurs ou des passerelles, celles des réseaux ad hoc ne possèdent pas de point de concentration. En l'absence de ce point de concentration, la détection d'intrusions doit être distribuée sur l'ensemble des nœuds actifs à l'intérieur du réseau. De plus, dans un réseau sans fil ad hoc chaque nœud possède une vision limitée de l'activité du réseau. Cette limite dépend essentiellement des caractéristiques des stations en terme d'émission-/réception et constitue une autre contrainte importante pour les algorithmes de détection des intrusions. Sans possibilité d'analyse globale des activités du réseau, la détection des intrusions est plus difficile.

Il est donc nécessaire de définir un modèle d'architecture d'IDS distribué adapté aux spécificités des réseaux ad hoc. Dans ce type d'architecture, basée sur un modèle distribué et coopératif, chaque nœud est indépendant et responsable localement de la détection d'éventuels signes d'intrusions. La prise en compte des informations collectées, ou des intrusions identifiées par les nœuds voisins situés dans sa zone d'émission-/réception, permet à un nœud d'obtenir une vision plus large de l'état du réseau. Nous présentons dans la partie suivante une synthèse des architectures des principaux IDS distribués et coopératifs.

4 Les IDS distribués et coopératifs

Nom	Type de détection	Prétaitement des données	Analyse données (détection)	Type de réponse	Origine
AAFID [9]	scénarios	distribué	centralisée	passive	système
CSM [10]	comportementale	distribué	distribuée	active	système
DIDS [11]	hybride	distribué	centralisée	passive	système/réseau
DPEM [12]	comportementale	distribué	centralisée	passive	système
GrIDS [13]	hybride	distribué	centralisée	passive	système/réseau
IDA [14]	scénarios	agents mobiles	centralisée	passive	système
JiNao [15]	hybride	distribué	centralisée	passive	MIB/réseau
MICAEL [16]	scénarios	agents mobiles	distribuée	passive	MIB
SPARDA [17]	scénarios	agents mobiles	distribuée	passive	système/réseau

Tab. 1. Synthèse des caractéristiques des IDS distribués.

Les architectures d'IDS présentées dans la table 1 ne sont pas spécifiques aux environnements des MANET. Leur point commun est un pré-traitement des informations à analyser dès leur collecte mais, à l'exception de CSM, MICAEL et SPARTA, le processus de détection nécessite des organisations hiérarchiques autour d'un nœud central permanent.

L'architecture de CSM (*Cooperating Security Managers*) est totalement distribuée; l'IDS local installé sur chaque nœud coopère avec les autres IDS locaux pour identifier l'origine des connexions dans un réseau. Cet IDS a été étudié spécifiquement pour suivre les connexions à travers les réseaux.

SPARTA (*Security Policy Adaptation Reinforced Through Agents*) utilise des agents mobiles pour collecter les informations à analyser. L'architecture nécessite la présence d'un nœud central dont le rôle est de maintenir une base de connaissance des nœuds présents dans le réseau. Cette dernière caractéristique ne permet donc pas à SPARTA d'être déployé dans les réseaux sans fil ad hoc.

MICAEL utilise une plate-forme pour agents mobiles sur chaque nœud. La distribution de ces agents mobiles sur chacun des nœuds est dynamique et gérée par un nœud central.

Nécessitant la présence d'un nœud central permanent ces différents modèles d'architectures d'IDS distribués ne sont pas adaptés aux réseaux sans fil ad hoc.

5 Proposition d'architecture d'IDS distribué pour réseau sans fil ad hoc

5.1 Les spécifications

Les spécifications s'appuient sur les caractéristiques générales des IDS et sur les contraintes spécifiques imposées par les réseaux ad hoc. Les caractéristiques résumées ici ont été présentées dans les chapitres précédents.

- *Principes de détection* : l'architecture de l'IDS doit être indépendante de la méthode de détection.
- *Sources de données* : l'architecture de l'IDS doit être indépendante de la source des données.
- *Fréquence d'utilisation* : la détection des tentatives d'intrusions doit être réalisée en temps réel pour permettre aux utilisateurs de réagir immédiatement.
- *Comportement après détection* : sous le contrôle de l'utilisateur la réponse doit être active en local pour accroître le niveau de sécurité et informative vers les autres systèmes membres du même réseau.
- *Distribution des nœuds* : l'architecture de l'IDS doit prendre en compte le caractère spontané des réseaux ad hoc ainsi que l'absence de nœud central permanent.
- *Débits limités des liens inter-nœuds* : les technologies WLAN offrent encore aujourd'hui des débits inférieurs à ceux des LAN. L'IDS doit s'appuyer sur les technologies les moins consommatrices de ressources réseaux.
- *Mobilité des nœuds* : l'IDS doit posséder les mécanismes lui permettant de prendre en compte la mobilité des nœuds. Toutefois, différentes optimisations pourront être proposées selon les caractéristiques de mobilité et de densité des nœuds.
- *Caractéristiques des nœuds et portabilité de l'IDS* : la surcharge induite par la détection des intrusions ne doit pas altérer de plus de 10% les performances des systèmes et être indépendante des systèmes d'exploitation.
- *Normalisation* : l'architecture de l'IDS doit adopter les normes et standards afin de pouvoir coopérer avec d'autres IDS.

5.2 L'architecture globale de l'IDS distribué

Nous proposons d'équiper chaque nœud du réseau d'un système de détection local (*LIDS, Local Intrusion Detection System*). Les ressources réseaux ne sont utilisées que pour informer les autres nœuds d'une attaque détectée localement et, si nécessaire, pour collecter des informations complémentaires disponibles uniquement sur d'autres nœuds du réseau. L'architecture globale est représentée figure 5. Ce modèle d'architecture est basé sur une plate-forme pour agents mobiles. Un agent mobile est défini comme une entité logicielle qui fonctionne de manière autonome et continue dans un environnement particulier, capable de se déplacer et de s'adapter aux changements de l'environnement, de communiquer et de coopérer avec d'autres agents.

Dans notre modèle, les entités logicielles hébergées sur chaque nœud (LIDS) fonctionnent de manière indépendante et observent les activités locales. Le LIDS détecte les intrusions à partir de cette surveillance locale et peut amorcer des réponses en conséquence. Si une anomalie est détectée ou si les signes d'intrusion nécessitent d'être confirmés, des agents mobiles peuvent être créés et envoyés vers d'autres nœuds pour y effectuer des tâches spécifiques et ensuite retourner les informations recherchées.

La mise en œuvre des agents mobiles, qui permet de déplacer le code vers les données à analyser, est une alternative aux architectures clients/serveurs. Cette solution de communication entre les nœuds est efficace si le code de l'agent est moins volumineux que celui des données à analyser. Notre architecture nécessite uniquement une mobilité faible (codes et données) des agents. Les travaux présentés dans [16,17,18,19] donnent les avantages apportés par les agents mobiles dans les réseaux sans fil ad hoc :

- L'*asynchronisme* se traduit par la possibilité de déléguer une tâche à un agent mobile sans rester en attente du résultat.

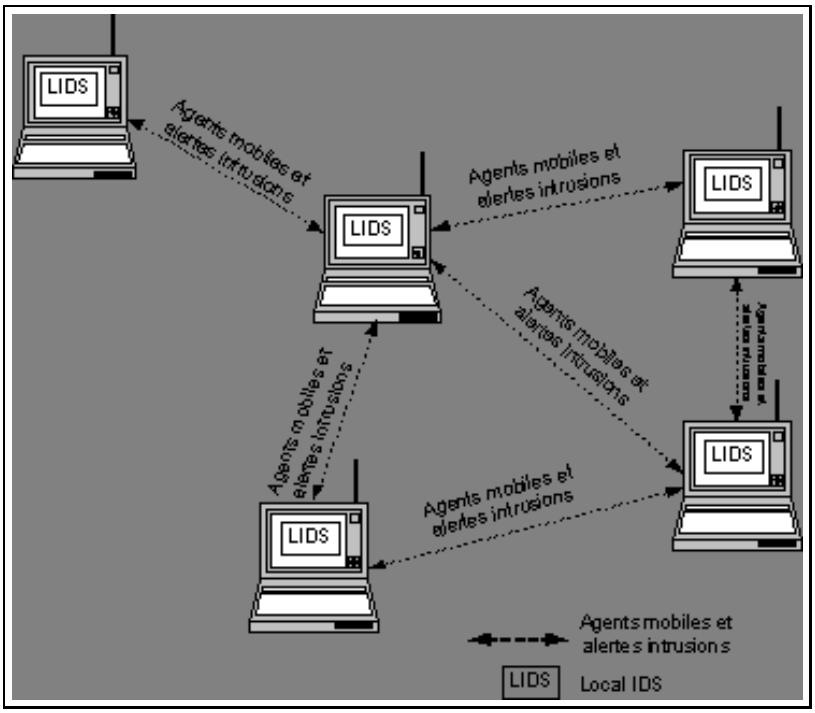


Fig. 5. Architecture globale de l'IDS.

- L'*autonomie* et l'*intelligence* permettent aux agents d'adapter leur comportement selon les informations collectées localement et éventuellement de poursuivre leurs déplacements au sein du réseau pour terminer leur mission.
- Le *temps de réponse des applications* et la *charge du réseau* peuvent être réduits par le déplacement du code vers les données plutôt que de déplacer les données vers les applications.

6 Architecture interne d'un LIDS

La figure 6 représente l'architecture interne d'un LIDS. La détection des intrusions est effectuée localement à partir des données collectées dans la MIB (*Management Information Base*).

- L'*agent IDS local* forme le cœur du système de détection d'intrusion. Il analyse les données collectées localement. Par défaut la détection s'appuie sur l'état des variables de la MIB II. Ces informations peuvent être complétées par des MIB spéciales, par exemple une extension de MIB spécifique au protocole de routage ad hoc, ou la MIB RMon pour prendre en compte les activités du réseau.
- L'*agent MIB local* a pour fonction de gérer les extensions de MIB spécifiques à la détection d'intrusions. Par exemple, pour utiliser une extension de MIB

spécifique au protocole de routage, un agent local sera chargé de la mise à jour des variables.
- La *plate-forme à Agents Mobiles* gère à la demande de l'agent IDS local l'ensemble des activités des agents mobiles (création, mobilité, sécurité, exécution, communication).
- L'*agent SNMP* est un agent SNMP standard dont la fonction est de répondre aux requêtes locales des LIDS et des agents mobiles.

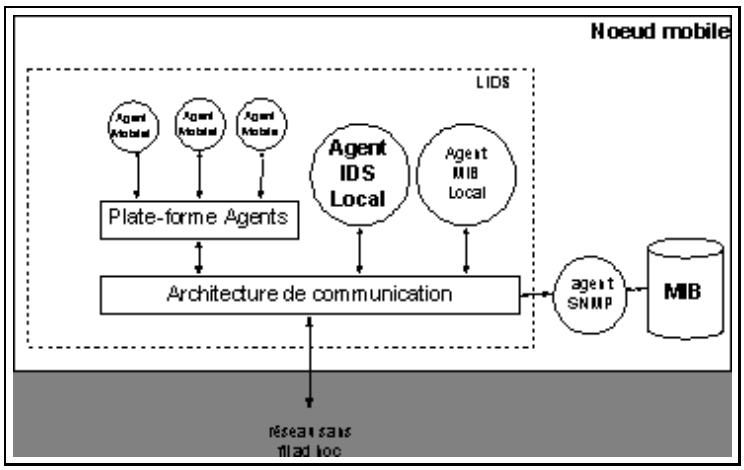


Fig. 6. Architecture d'un LIDS.

Architecture de l'agent IDS Local L'architecture interne retenue pour la conception des LIDS s'appuie sur la description des modules internes du modèle d'IDS proposé par l'IDWG (*Intrusion Detection Working Group* de l'IETF[3]). Celui-ci définit les trois composants de base d'un IDS, représentés sur la figure 7 :
- le *senseur* dont la fonction est de collecter et de mettre en forme les données brutes collectées dans le système ou sur le réseau,
- l'*analyseur* qui traite les informations générées par le senseur afin d'identifier les intrusions,
- le *manager* qui a pour fonction de gérer les alarmes après détection.

Notre modèle ajoute un nouveau type de message entre l'analyseur et le senseur. Ce message appelé *requête* permet à l'analyseur de demander et de collecter, s'il le souhaite, des informations complémentaires.

Architecture détaillée Le LIDS possède une architecture construite autour de quatre modules de base :

[3] Document officiel de l'IDWG : *http ://www.ietf.org/html.charters/idwg-charter.html*

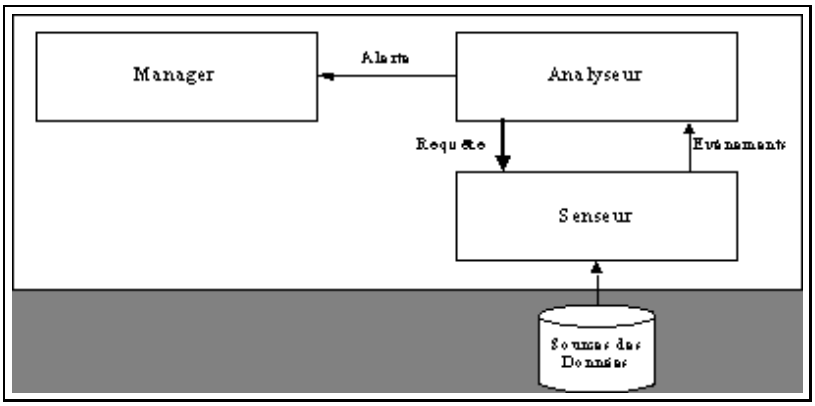


Fig. 7. Modèle IDS de l'IDWG, complété par le message requête.

- *IDS Framework* : ce module constitue le cœur du système de détection d'intrusion. Dans la première version, il utilise une détection par analyse de signatures. Chaque attaque est représentée par un diagramme d'états dont les transitions sont réalisées selon des évènements prédéfinis représentés par des valeurs de variables MIB.
- *Event Abstraction Framework* : le module d'abstraction des évènements a pour fonction de convertir un événement représentatif d'une signature en requête sur des variables de la MIB locale.
- *Mobile Agent Framework* : le module d'agents mobiles est constitué de la plateforme AGLET (sources IBM, *http ://www.trl.ibm.co.jp/agletts/*).
- *IDS communication Framework* : le module de communication avec des IDS externes a pour fonction d'assurer l'interopérabilité avec d'autres systèmes de détection d'intrusions.

Les modules externes *Attack Signatures, Event and Alert specification et Event Abstractor Database* sont des sources de données dont l'objectif est de faciliter les paramétrages du LIDS.

La MIB, *Management Information Base*, regroupe l'ensemble des variables de la MIB II et des variables complémentaires implémentées dans la MIB expérimentale.

Les différents composants du LIDS communiquent par les flots d'informations internes suivants :

- *La signature d'une attaque* est représentée par une machine d'états dans le module *IDS Kernel*. Chaque transition d'un état à un autre est la conséquence d'un événement.
- *La gestion des évènements* est assurée par le module *Audit Data Management*. Celui-ci reçoit du module *IDS Kernel* les caractéristiques des événements à détecter.
- *Le module Audit Data Management* demande soit au module *local Event Abstraction Framework* soit au *module Mobile Agent Framework* de détecter l'événement. Des alertes issues d'autres LIDS peuvent aussi être prises en compte par le module *Alert Management*. La recherche d'événement peut être réalisée à la demande ou de façon périodique selon les besoins de l'*IDS Kernel*.

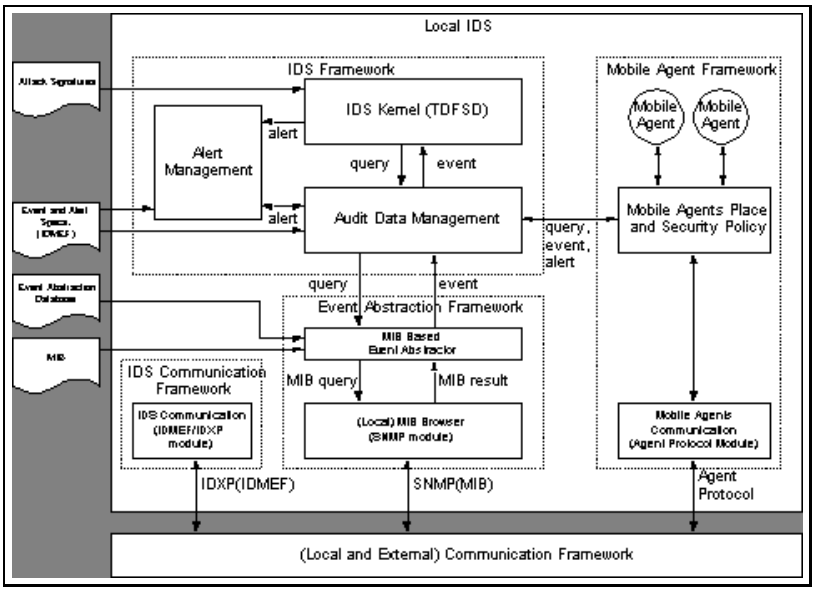


Fig. 8. Architecture modulaire des LIDS.

Le module MIB *Base Event Abstractor* a pour fonction d'associer à chacun des évènements les variables MIB correspondantes. Il s'appuie sur le module *Local MIB Browser* pour réaliser les requêtes SNMP (*GET et GETNext*).

Selon la valeur des variables MIB reçues, le module MIB *Base Event Abstractor* peut indiquer au module *Audit Data Management* l'occurrence d'un événement recherché. Celui-ci transmettra l'information au module *IDS Kernel*.

7 Tests de validation fonctionnelle

7.1 Description des tests réalisés

Seuls les modules du LIDS (figure 8) nécessaires aux tests de validation fonctionnelle ont été implémentés. Dans cette première phase de tests, dont l'objectif est de valider les choix techniques réalisés, nous avons sélectionné deux familles d'attaques représentatives des vulnérabilités inhérentes aux réseaux sans fil ad hoc :

- Attaques sur le protocole de routage OLSR.
- Déni de service N sauts : L'attaquant génère pour chaque message HELLO transmis par un nœud voisin un nouveau message HELLO dans lequel toutes les liaisons, contenues dans le message HELLO initial, sont annoncées asymétriques. Cette modification de l'état des liens a pour conséquence d'isoler le nœud émetteur du message HELLO initial et ainsi de perturber le routage.
- Déni de Service $1 + N$ Sauts, détournement de MPR : L'attaquant se fait élire MPR par ses nœuds voisins en annonçant dans de faux messages TC sa capacité

à atteindre des nœuds inexistants. Ses proches voisins vont ainsi lui transmettre l'ensemble des messages destinés à des nœuds distants.

– Attaques réseaux sur le protocole applicatif Telnet.

Ce type d'attaque, connu sous le nom de *Stepstone* [20], a pour objectif de masquer l'identité d'un attaquant en prenant le contrôle d'un nœud et en générant ensuite depuis celui-ci des attaques à destination d'autres nœuds. Une suite en cascade de connexions Telnet sur différents nœuds représente un exemple classique de ce type d'attaque.

Les nœuds d'un MANET, dépourvus des protections liées à l'infrastructure du réseau (Firewall, Proxy, Serveurs d'authentification, etc.) sont particulièrement exposés à ce type d'attaque.

Pour détecter les attaques sur le protocole de routage OLSR, les LIDS utilisent uniquement les variables spécifiques au protocole OLSR, mémorisées dans l'extension de la MIB locale. L'attaque de type *Stepstone* nécessite la mise en œuvre de la plate-forme à agents mobiles pour identifier l'origine des connexions Telnet.

Ces différentes attaques ont été réalisées et détectées sur la plate-forme de tests du projet RAHMS. Nous décrivons dans la section suivante la détection de l'attaque par rebonds Telnet.

7.2 Description et détection de l'attaque Stepstone

Principe de la détection Pour expliquer le mécanisme de la détection des attaques par rebonds Telnet, nous représentons sur la figure 9 les seuls modules des LIDS impliqués dans cette détection.

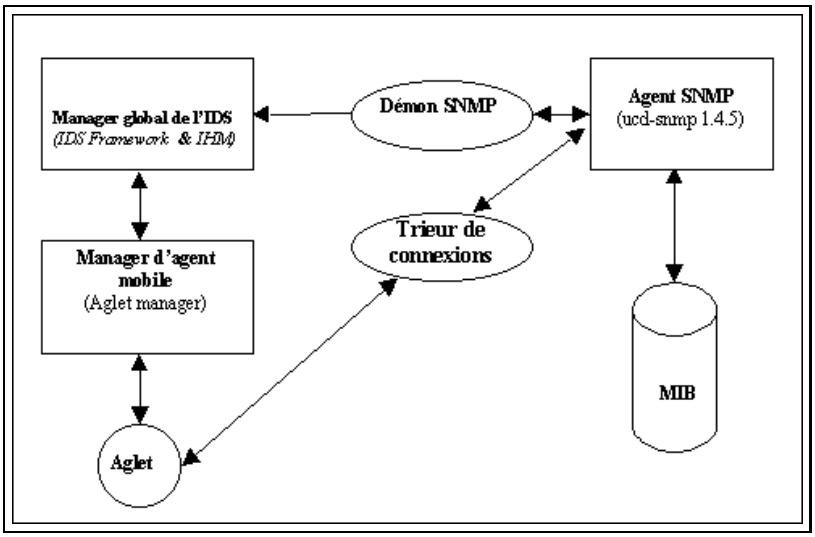


Fig. 9. Modules LIDS impliqués dans la détection de l'attaque Stepstone.

- Le *manager global* est chargé de détecter l'origine les connexions. Pour cela il s'appuie sur les informations collectées par les agents mobiles. Une interface homme machine (IHM) a été développée pour visualiser le suivi des connexions. La réponse à la détection d'une attaque *Stepstone*, i.e. la fermeture de la connexion, peut-être réalisée à l'initiative de l'opérateur.
- L'*agent SNMP* est utilisé pour collecter les informations dans la MIB.
- Le *démon SNMP* a pour fonction de détecter toute nouvelle connexion entrante ou sortante sur le nœud protégé. Il scrute la MIB à intervalles réguliers.
- Le *gestionnaire d'agents mobiles* contrôle l'activité des agents mobiles (création, accueil, sécurité, arrêt).
- Le *module trieur de connexions* a pour fonction de communiquer à un agent mobile la source locale de la connexion détectée sur le nœud suivant.

Détection de l'attaque Nous considérons ici une connexion Telnet réalisée en cascade sur trois nœuds équipés d'un LIDS et d'un agent SNMP. Cette chaîne de connexions est représentée sur la figure 10. Nous nous intéressons au LIDS du nœud

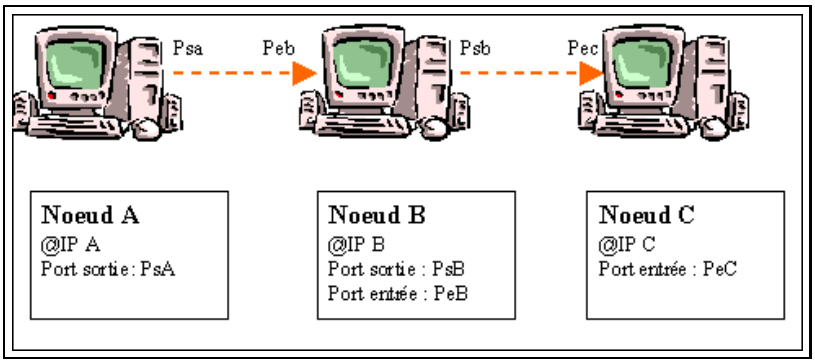


Fig. 10. Chaîne de connexions Telnet.

C. Le comportement des LIDS situés sur les autres nœuds est identique.

- *Première étape* : sur le nœud C, le démon SNMP informe le manager global de l'arrivée d'une connexion TCP. Il transmet l'adresse IP et le port à l'origine de cette connexion sur le nœud distant. Dans notre exemple @IPB et PsB.
- *Deuxième étape* : sur le nœud C, le manager global ne peut détecter une chaîne de connexion sans informations complémentaires, non disponibles dans la MIB locale. Il transmet les informations collectées (@IPB et PsB) au manager d'agents mobiles. Ce dernier va alors créer une Aglet et l'envoyer sur le nœud B pour collecter des informations complémentaires.
- *Troisième étape* : sur le nœud B, l'Aglet, est authentifiée et accueillie par la plateforme à agents locale. Elle s'adresse alors au module trieur de connexions pour lui demander de lui fournir l'origine de la connexion dont le port de sortie est PsB, suite à la requête effectuée dans la MIB par l'agent SNMP pour identifier

l'origine de la connexion. Dans notre exemple, le trieur de connexions transmet @IPA et PsA à l'Aglet.

- *Quatrième étape* : l'Aglet se déplace alors du nœud B vers le nœud A. Sur ce dernier le trieur de connexions indique à l'Aglet que le nœud est l'origine de la connexion.
- *Cinquième étape* : l'Aglet se déplace vers le nœud C avec les données collectées sur les différents nœuds impliqués dans la connexion. Le manager d'agents mobiles transmet les données au manager global du LIDS qui, après analyse, les transmet au module IHM, représenté sur la figure 11. La chaîne de connexions Telnet peut ainsi être visualisée par l'utilisateur du nœud C qui pourra décider de l'interrompre ou non.

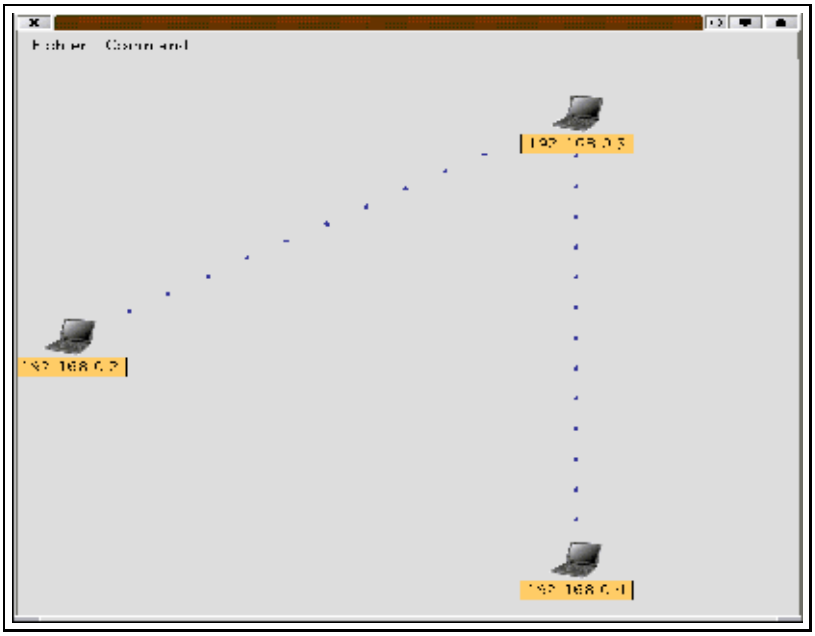


Fig. 11. Interface LIDS.

8 Conclusions et travaux futurs

Nous avons présenté dans cet article l'architecture d'un IDS distribué pour les réseaux sans fil ad hoc. Les tests fonctionnels réalisés en laboratoire sont positifs : les attaques implémentées sont détectées en temps réel. L'utilisation des agents mobiles dans un environnement sans fil a été validé par le biais d'un prototype, mais des mesures de performances restent toutefois à réaliser, notamment dans un réseau constitué d'un grand nombre de nœuds et avec différentes conditions de trafic. Cette prochaine

étape nécessitera peut-être la définition d'une nouvelle plate-forme à agents mobiles optimisée pour la détection d'intrusions et les réseaux sans fil. Dans notre prototype, la sécurité des LIDS s'appuie sur les mécanismes de sécurité du langage Java et de la plate-forme Aglet. L'ajout d'un agent mobile chargé d'évaluer l'intégrité globale du système constitué par l'ensemble des LIDS est actuellement en cours d'étude. La conception modulaire des LIDS nous permet aussi d'évaluer les performances d'une détection comportementale ou mixte et de différentes sources d'information. La détection de l'attaque par rebonds Telnet est adaptée à la détection de tous types de connexions TCP, néanmoins les possibilités de détection des LIDS devront être étendues à un plus grand nombre d'attaques.

9 Remerciements

L'architecture distribuée du système de détection d'intrusions pour les réseaux sans fil ad hoc, présentée dans cet article, a été réalisée dans le cadre du projet RNRT pré-compétitif RAHMS (Réseau Ad Hoc Multiservice Sécurisé). Les auteurs tiennent ici à remercier le RNRT et les ministére ministères de l'industrie et de la recherche pour leur soutien. En outre, les auteurs remercient l'ensemble des participants au projet et tout particulièrement Patrick Albers et Olivier Camp (ESEO), Ludovic Mé et Ricardo Puttini (Supélec).

Références

1. Scott Corson, Joseph Macker. *RFC 2501, Mobile Ad hoc Networking (MANET).* IETF (1999)
2. Samir R. Das, Charles E. Perkins, et Elizabeth M. Royer. *Performance comparison of two on-demand routing protocols for ad hoc networks.* In Proceedings of the IEEE Conference on Computer Communications, INFOCOM (2000).
3. Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot. draft-ietf-manet-olsr-09.txt. *Optimized Link State Routing Protocol.* IETF (2003).
4. *RAHMS, Réseau Ad Hoc Multiservices Sécurisé.* Projet RNRT 1999 numéro 65. Dossier d'étude de l'architecture de sécurité. Rapport interne (2002).
5. L.Zhou and Z. Haas. *Securing Ad Hoc Networks.* IEEE Network, Vol 13, Nov.-Dec. 1999, pp.24-30 (1999).
6. H. Luo, P. Zerfos,J. Kong, S. Lu, and L.Zhang. *Self Securing Ad Hoc Wireless Networks.* Proceedings of the Seventh International Symposium on Computers and Communications, ISCC'02 (2002)
7. Patrick Albers, Olivier Camp, Jean-Marc Percher, Bernard Jouga, Ludovic Mé, and Riccardo Puttini. *Security in Ad hoc Networks : a General Intrusion Detection Architecture Enhancing Trust Based Approaches.* WIS 2002, Ciudad Real, Spain (2002).
8. Ludovic Mé et Cédric Michel. *La détection d'intrusions : bref aperçu et derniers développements.* Actes du congrès EUROSEC'99 (1999).
9. Jai Sundar Balasubraniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, Diego Zamboni. AAFID, *Autonomous Agents For Intrusion Detection.* Technical report 98/05, COAST Laboratory, Purdue University (1998).

10. Gregory B White, Eric A. Fish and Udo Pooch. *CSM - Cooperating Security Managers : a Peer Based Intrusion Detection System.* IEEE Networks, pp.20-23, January/February (1996).
11. Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukkherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. *DIDS, Distributed Intrusion Detection System.* Computer Security Laboratory, Department of Computer Science, University of California, Davis. (1992).
12. C. Ko, M. Ruschitzka, and K. Levitt. *Execution Monitoring of Security-Critical Programs in Distributed Systems : A Specification-based Approach.* Proceedings of the 1997 IEEE Symposium on Security and Privacy (1997)
13. Stuart Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle. *GrIDS, A Graph Based Intrusion Detection System for Large Networks.* Computer Security Laboratory, Department of Computer Science, University of California, Davis (1996).
14. Midori Asaka, Atsushi Taguchi, Shigeki Goto. *The Implementation of IDA : An Intrusion Detection Agent System.* IPA, Waseda University (1999).
15. Y.F Fou, F. Gong, C. Sargor, X. Wu, S. F. Wu, H. C. Chang, F. Wang. *JINAO, Design and Implementation of a Scalable Intrusion Detection System for the OSPF Routing Protocol.* Advanced Networking Research, MCNC Computer Science Dept, NC State University (1999).
16. J. Duarte de Queiroz, L. Fernando Rust da Costa Carmo and L. Pirmez. *Micael : An Autonomous Mobile Agent System to Protect New Generation Networked Applications.* Nucleo de Computacao Eletronica UFRJ, Brazil. In the Proceedings of RAID'99 (1999).
17. C. Krugel and T.Toth. *Flexible, Mobile Agent Based intrusion Detection for Dynamic Networks.* European Wireless 2002, Florence, Italy (2002).
18. G. Helmer, J. S. K. Wong, V. Honava,. L. Miller and Y. Wang. *Lightweight Agents For Intrusion Detection.* Journal of Systems and Software, *http ://citeseer.nj.nec.com/helmer00lightweight.html*
19. Nadia Boukhatem. *Les agents mobiles et leurs applications.* DNAC'99, Paris (1999).
20. Stuart Staniford-Chen and Todd Heberlien. *Holding Intruders Accountable on the Internet.* Proceeedings of the 1995 IEEE Symposium on Security and Privacy, pp.39-49, Oakland (1995).

La Sécurité dans les Réseaux Sans Fil Ad Hoc

Valérie Gayraud[1], Loutfi Nuaymi[2], Francis Dupont[2], Sylvain Gombault[2], and Bruno Tharon[2]

Thomson R&I, Security Lab,
1, Avenue de Belle-Fontaine,
35551 Cesson Sévigné
valerie.gayraud@thomson.net
ENST Bretagne,
2, Rue de la Châtaigneraie,
CS 17607, 35576 Cesson Sévigné
{loutfi.nuaymi, francis.dupont, sylvain.gombault,
bruno.tharon}@enst-bretagne.fr

Résumé Cet article présente une étude sur la sécurité des réseaux sans fil ad hoc. La première partie définit la notion de réseaux sans fil ad hoc et leurs contextes d'utilisation. Une synthèse d'analyse de risque de haut niveau menée sur ce type particulier de réseaux fait l'objet de la seconde partie. La troisième partie se focalise sur le routage, une fonction particulièrement sensible des réseaux sans fil ad hoc. Le fonctionnement général des protocoles de routage et les attaques correspondantes sont présentés. La partie quatre concerne l'état de l'art actuel des solutions proposées par les différentes équipes de recherche travaillant sur ce sujet à travers le monde. Nous concluons sur la sécurisation des réseaux sans fil ad hoc et indiquons les axes futurs de développement qui nous semblent intéressants.

1 Les Réseaux sans fil Ad Hoc

Les réseaux sans fil ad hoc sont composés de systèmes informatiques divers, plus ou moins complexes, appelés **nœuds** par la suite, ayant la possibilité de communiquer de manière autonome par ondes radio. Les nœuds interagissent et peuvent coopérer pour s'échanger des services. Un nœud peut à la fois communiquer directement avec d'autres nœuds ou servir de **relais**. Un relais permet à des nœuds se trouvant hors de portée radio les uns des autres de communiquer. Ces réseaux sont dits ad hoc dans la mesure où ils ne nécessitent pas d'infrastructure fixe. Ils peuvent exister temporairement pour répondre à un besoin ponctuel de communication. Le mode de fonctionnement **ad hoc** se distingue du mode **infrastructure** dans lequel les nœuds du réseau communiquent entre eux via un point d'accès, aussi appelé base, qui peut être relié à un réseau fixe. La figure 1 montre la différence d'utilisation des réseaux sans fil en mode infrastructure fixe et en mode ad hoc. Les applications de tels réseaux sont nombreuses et tendent à se multiplier avec la miniaturisation des processeurs et la diversité des terminaux. Les situations à caractère exceptionnel se prêtent bien à l'utilisation de réseaux sans fil ad hoc : opérations militaires, couvertures d'évènements sportifs, opérations de secours. Plus simplement, une réunion de travail peut demander la création ponctuelle d'un réseau informatique entre ses participants. Les réseaux domestiques représentent aussi

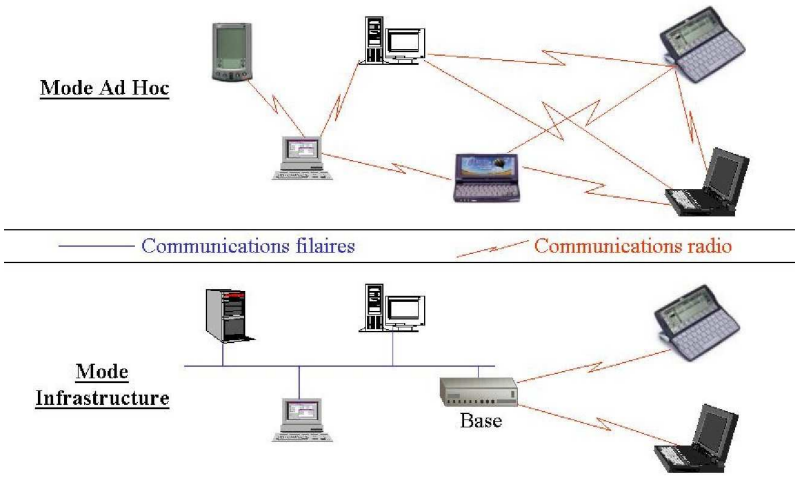


Fig. 1. Mode infrastructure versus mode ad hoc

un vaste champ d'application en plein devenir. L'organisation d'une soirée de jeux vidéo en réseaux, où chacun apporte son matériel, illustre une utilisation possible dans ce milieu.

L'**IETF** (*Internet Engineering Task Force*) a créé une entité spécifique pour les réseaux mobiles ad hoc, le groupe de travail **MANET** (*Mobile Ad hoc Networks*). Les membres de ce groupe soumettent régulièrement des propositions de protocoles de routage adaptés aux réseaux mobiles ad hoc.

Les réseaux sans fil ad hoc s'appuient sur les technologies sans fil conçues à l'origine pour des réseaux locaux et domestiques :

- Les technologies IEEE 802.11a, 802.11b (*Wireless Fidelity*, WiFi), 802.11g, HiperLan/1 (remplacé par HiperLan/2), HomeRF (SWAP) sont propres aux réseaux WLAN (*Wireless Local Area Network*).
- La technologie Bluetooth, pour les réseaux **WPAN** (*Wireless Personal Area Network*). Bluetooth fonctionne en mode point à point ou point à multipoint.
- Les technologies infrarouges (IrDA, *Infrared Data Association*), utilisées dans les télécommandes par exemple, peuvent aussi être considérées comme support des réseaux ad hoc. Mais ces technologies se limitent à des communications point à point.

Les technologies WiFi, IEEE 802.11g, HiperLan, HomeRF et Bluetooth opèrent dans la bande ISM (*Industrial, Scientific and Medical*) à 2.4 GHz alors que 802.11a opère dans la région des 5 GHz.

2 Les Risques Liés à la Sécurité Informatique

2.1 L'Analyse de Risque en Sécurité

L'analyse de risque est nécessaire pour bien appréhender la problématique de la sécurité dans les réseaux sans fil ad hoc. Elle suit les étapes suivantes :

1. Détermination des **fonctions** et **données** sensibles des réseaux sans fil ad hoc.
2. Recherche des **exigences de sécurité** par le biais des critères de sécurité que sont l'authentification, l'intégrité, la confidentialité, l'anonymat et la disponibilité.
3. Étude des **vulnérabilités**.
4. Étude des **menaces** et quantification de leur probabilité d'occurrence ou de leur faisabilité.
5. **Mesure du risque** encouru en fonction des vulnérabilités mises en lumière et des menaces associées.

À partir de ces différents points d'entrée, il est possible de déterminer quelles sont les parties critiques, en terme de sécurité, que les concepteurs, administrateurs, et utilisateurs de réseaux sans fil ad hoc doivent appréhender. La figure 2 retrace les différentes phases de ce processus.

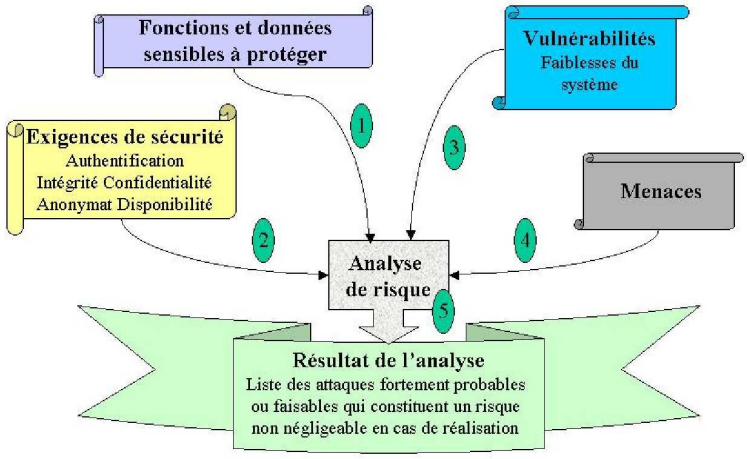


Fig. 2. Les étapes de l'analyse de risque

Il faut noter qu'une généralisation des besoins en sécurité faisant abstraction des contextes d'utilisation a été nécessaire pour mener à bien cette analyse de risque. En effet, une application commerciale civile, par exemple, n'aura pas les même contraintes qu'une application militaire. Un contexte militaire mettra en avant le fort besoin d'authentification, de furtivité et d'intégrité physique des éléments alors qu'une utilisation commerciale critique nécessitera de se focaliser sur la confidentialité des services. Selon

les cas, il peut donc être indispensable d'étudier des solutions appropriées au contexte d'utilisation à travers une analyse approfondie prenant en compte des contraintes spécifiques. L'analyse indiquée ici se veut générale et peut servir de base à une telle étude.

2.2 Fonctions et Données Sensibles

Les fonctions sensibles des nœuds d'un réseau sans fil ad hoc sont le **routage**, la **configuration**, la **gestion d'énergie**, et les **mécanismes de sécurité**. La plupart des données sensibles sont directement liées à ces fonctions puisqu'il s'agit :

- Des données relatives au routage (tables de routage et données de configuration des mécanismes de routage)
- Des mesures et données de configuration pour la gestion de l'énergie.
- Des données relatives à la sécurité (clés cryptographiques, mots de passe, certificats...)
- D'une manière générale tout ce qui concerne les données de configuration.

Les informations personnelles des utilisateurs doivent aussi être considérées comme des données sensibles.

2.3 Exigences de Sécurité des Réseaux sans fil Ad Hoc

Contraintes :

Déterminer les exigences de sécurité d'un système nécessite d'appréhender l'ensemble des contraintes qui pèsent sur ce système. Cette étape permet par la suite de quantifier les critères de sécurité. Les spécificités des réseaux sans fil ad hoc sont multiples. On peut les répartir en six grands thèmes traitant des caractéristiques des nœuds, de la gestion de l'énergie, des caractéristiques du réseau, des technologies sans fil sous jacentes, de la mobilité et de la configuration.

Caractéristiques des Nœuds :

- Les participants peuvent posséder des systèmes **hétérogènes** qui doivent s'interconnecter facilement.
- Certains éléments peuvent avoir de **faibles capacités de calculs**.

Gestion de l'Énergie :

- L'**énergie** doit être conservée au maximum pour éviter d'incessantes recharges du système qui diminuent sa mobilité.
- Les nœuds chercheront donc à se mettre en veille le plus souvent possible, ce qui provoquera alors une **diminution de réactivité** de l'ensemble du réseau.

Caractéristiques du réseau :

- La **charge** du réseau doit être **distribuée équitablement** entre les éléments en tenant compte de leur capacité respective.
- Chaque élément d'un réseau ad hoc est autonome et possède à la fois les fonctionnalités de **relais** et de **point de communication**. L'administration de ces éléments reste interne au réseau.
- L'**absence d'infrastructure centralisée** sera une contrainte très forte pour la gestion des accès aux ressources du réseau.

Technologie sans fil :

- Les perturbations dûes à l'environnement radio peuvent entraîner des **diminutions de débit et bande passante**.
- Les réseaux sans fil ad hoc héritent de l'architecture propre aux technologies WLAN et WPAN, et notamment des couches **physique** et **liaison de données** de ces technologies.

Mobilité :

- Les éléments étant fortement mobiles, leur **sécurité physique** est moins assurée que pour un poste de travail fixe, dans un bureau par exemple. Leur valeur marchande peut être d'importance non négligeable.
- La topologie du réseau peut changer d'autant plus rapidement que les nœuds sont **mobiles**.
- Des **liens asymétriques** peuvent se créer lorsqu'un élément muni d'un récepteur particulièrement sensible est capable de capter les émissions d'un autre nœud qui est hors de portée du premier élément.

Configuration :

- L'**auto configuration** permet aux nœuds de s'intégrer facilement dans un réseau. Elle facilite la gestion du réseau car l'interconnexion des éléments ne nécessite qu'un minimum d'intervention technique externe. Cette fonctionnalité est de plus en plus nécessaire pour un déploiement à grande échelle des réseaux sans fil ad hoc.

Authentification / Intégrité / Confidentialité / Disponibilité :

Coopérer au sein de tels réseaux présente un risque s'il n'y a aucun contrôle des participants. L'**authentification** des parties apparaît donc comme la pierre angulaire d'un réseau sans fil ad hoc sécurisé. En effet, comment assurer une quelconque confidentialité et intégrité des messages échangés si, dès le départ, on n'est pas sûr de communiquer avec la bonne entité ?

Contrairement au réseau filaire, il n'est pas nécessaire de pénétrer dans un local physique pour accéder au réseau. Si l'authentification est mal gérée, un attaquant peut s'attacher au réseau sans fil et injecter des messages erronés. L'**intégrité** des messages échangés est donc une exigence importante pour ces réseaux. L'intégrité des nœuds est, elle aussi, primordiale car les éléments d'un réseau ad hoc sont moins sujets à surveillance. En effet, ils ne sont pas confinés dans un bureau mais transportés par leur propriétaire et peuvent donc être momentanément égarés. Un attaquant peut subtiliser un appareil, le corrompre avec un cheval de Troie par exemple, avant de le restituer discrètement à son propriétaire.

Une fois les parties authentifiées, la **confidentialité** reste un point important étant donné que les communications transitent via les airs et sont donc potentiellement accessibles à tout possesseur du récepteur adéquat.

La **disponibilité** est une propriété difficile à gérer dans les réseaux sans fil ad hoc étant donné les contraintes qui pèsent sur ces réseaux :

- Topologie dynamique.
- Ressources limitées sur certains nœuds de transit.
- Communications sans fil pouvant être facilement brouillées ou perturbées.

Les applications sans fil en mode ad hoc ne devraient donc pas se focaliser sur ce critère.

Anonymat / Protection de la Vie Privée :
Certaines applications peuvent nécessiter la discrétion sur l'identité des participants qui collaborent au réseau sans fil ad hoc : par exemple un vote anonyme au cours d'une conférence. De plus, les différents gadgets électroniques qui formeront les nœuds des réseaux ad hoc de demain, auront en toute probabilité, la possibilité de garder la trace de nos préférences afin de nous faciliter le quotidien et de nous offrir des services toujours plus appropriés. Cette tendance va pourtant à l'encontre de la protection de la vie privée de tout un chacun. Qui a envie de voir diffuser sur les ondes ses goûts et affinités ?

2.4 Vulnérabilités

La première vulnérabilité de ces réseaux est liée à la **technologie sans fil** sous-jacente. Quiconque possédant le récepteur adéquat peut potentiellement écouter ou perturber les messages échangés. Et ceci, même s'il se trouve dans un lieu public, à l'extérieur du bâtiment où se déroulent les échanges.

Les **nœuds** eux-mêmes sont des points de vulnérabilités du réseau car un attaquant peut compromettre un élément laissé sans surveillance.

L'**absence d'infrastructure fixe** pénalise l'ensemble du réseau dans la mesure où il faut faire abstraction de toute entité centrale de gestion pour l'accès aux ressources.

Les **mécanismes de routage** sont d'autant plus critiques dans les réseaux ad hoc que chaque entité participe à l'acheminement des paquets à travers le réseau. De plus, les messages de routage transitent sur les ondes radio.

Enfin, ces réseaux héritent de toutes les vulnérabilités propres aux technologies sans fil WLAN et WPAN.

2.5 Menaces

On distingue les menaces de type passif, où l'attaquant est limité à l'écoute et l'analyse du trafic échangé, des menaces de type actif. Dans ce dernier mode, l'attaquant se donnera les moyens d'agir sur la gestion, la configuration et l'exploitation du réseau. Il peut injecter son propre trafic, modifier le fonctionnement d'un nœud, usurper l'identité d'un élément valide, rejouer des messages, modifier des messages transitant sur le réseau ou provoquer un déni de service. L'attaque passive prive le réseau de la confidentialité des messages échangés. Éventuellement, l'analyse du trafic représente un risque pour l'anonymat des participants et le respect de leur vie privée.

2.6 Résultat de l'Analyse de Risque

Après l'étude des besoins et exigences des réseaux sans fil ad hoc en terme de sécurité, puis corrélation avec les risques issus des vulnérabilités et menaces s'appliquant à ces réseaux nous avons pu dresser une liste des attaques fortement probables ou faisables et qui constituent un risque non négligeable en cas de réalisation.

Les dénis de services, *denial of services* (DoS), apparaissent comme les attaques les plus faciles à réaliser par un attaquant. La criticité de telles attaques dépend fortement du contexte d'utilisation mais n'est jamais complètement négligeable. Les modèles de dénis de services qui suivent se dégagent plus particulièrement dans le cas de réseau sans fil ad hoc :

- Brouillage du canal radio pour empêcher toute communication.

- Tentative de débordement des tables de routages des nœuds servant de relais.
- Non-coopération d'un nœud au bon fonctionnement du réseau dans le but de préserver son énergie. L'égoïsme d'un nœud est une notion propre aux réseaux ad hoc. Un réseau ad hoc s'appuie sur la collaboration sans condition de ses éléments. Un nœud refusant de jouer le jeu peut mettre en péril l'ensemble.
- Tentative de gaspillage de l'énergie de nœuds ayant une autonomie de batterie faible ou cherchant à rester autonome (sans recharge) le plus longtemps possible. Ces nœuds se caractérisent par leur propension à passer en mode veille le plus souvent possible. L'attaque consiste à faire en sorte que le nœud soit obligé de rester en état d'activité et ainsi de lui faire consommer toute son énergie. Cette attaque est référencée par Ross Anderson et Franck Stajano ([8,9])) sous l'appellation ***sleep deprivation torture attack***, un scénario de torture par privation du sommeil.
- Dispersion et suppression du trafic en jouant sur les mécanismes de routage.

Les **attaques passives d'écoute et d'analyse du trafic** constituent une menace certaine pour la confidentialité et l'anonymat.

L'**usurpation de l'identité d'un nœud** en leurrant les mécanismes de contrôle d'accès permet de nombreuses attaques actives rendant particulièrement critiques la protection des mécanismes de routage.

L'**attaque physique d'un élément valide** d'un réseau sans fil ad hoc, entraînant la compromission du nœud, se révèle comme étant un point faible de ces réseaux.

Enfin, il apparaît clairement que les attaques sur les **mécanismes de routage** sont particulièrement critiques. Ce papier accorde donc une large part à cette problématique dans la partie qui suit.

3 Le Routage dans les Réseaux sans fil Ad Hoc

Les protocoles de routage peuvent être classés en différentes familles selon le moment auquel ils initient la découverte de route, selon la manière dont les nœuds d'un réseau se partagent le travail de routage et selon la manière dont les informations de routage sont échangées.

3.1 Classification des Protocoles de Routage

Si un protocole initie la découverte de route lorsque le besoin s'en fait ressentir, c'est à dire lorsqu'un paquet doit être transmis vers une destination dont la route n'est pas connue dans la table de routage, il sera considéré comme faisant partie de la famille des protocoles **réactifs**. Si le protocole initie des découvertes de route régulièrement sans attendre qu'il y ait un paquet à transmettre, il sera dit **proactif**. Certains protocoles combinent ces deux manières d'initier des découvertes de routes, à la demande et en avance, et sont donc considérés comme **hybrides**. Certains protocoles de routage n'utilisent pas tous les nœuds d'un réseau pour faire transiter les messages, au contraire ils en sélectionnent certains, en fonction du voisinage ou pour former des cellules. Ces protocoles sont dits **non-uniformes**. Ceux qui utilisent tous les nœuds du réseau capables de router sont appelés **uniformes**. Les algorithmes permettant de maintenir la table de routage sont de deux types : les algorithmes basés sur le ***distance vector*** et ceux basés sur le ***link state***. Les protocoles de type *distance vector* n'ont qu'une vision partielle du réseau. Les protocoles de type *link state* maintiennent leur table

de routage à jour grâce à des annonces faites régulièrement par les différents nœuds reflétant l'état de l'ensemble des liaisons du réseau. Ces protocoles ont une vision totale du réseau. On peut citer quelques protocoles de routage proposés au sein du groupe de travail MANET :

- **AODV** (*Ad hoc On demand Distance Vector*) est un protocole réactif, uniforme, de type *distance vector*.
- **DSR** (*Dynamic Source Routing*) est réactif, uniforme, de type *link state*.
- **OLSR** (*Optimized Link State Routing*) est un protocole non-uniforme, proactif, de type *link state*.
- **FSR** (*Fisheye State Routing*) est proactif, uniforme, de type *distance vector*.

3.2 Le Routage de Paquets

Afin de comprendre les attaques sur les protocoles de routage, il est nécessaire de comprendre leur fonctionnement global. Lorsqu'un nœud dans un réseau veut émettre un message vers un autre nœud, il regarde dans sa table de routage si une route existe pour ce nœud. Si elle n'existe pas, il initie une découverte de route, ***route discovery***, en diffusant sur le réseau, dans les airs pour les accès sans fil, un message de type ***route request***. Le message de *route request* contient l'adresse du nœud émetteur, l'adresse du nœud destinataire, un marqueur permettant d'identifier la découverte de route et une liste initialement vide à remplir par les nœuds intermédiaires. Lorsqu'un nœud intermédiaire reçoit ce paquet, s'il n'en est pas le destinataire et si sa table de routage n'indique pas de chemin pour le nœud recherché, il diffuse à son tour le paquet de type *route request* en rajoutant son adresse à la liste de nœuds intermédiaires. Dans le cas où le nœud intermédiaire possède dans sa table de routage un chemin pour le nœud destinataire, la majorité des protocoles prévoit que le nœud intermédiaire renvoie directement un message de type ***route reply*** à l'émetteur en indiquant ce chemin. Lorsqu'un paquet de requête atteint son destinataire, ce dernier émet un paquet de réponse du type *route reply*. Ce paquet transite par les nœuds intermédiaires de la liste. La figure 3 schématise l'évolution des messages lors de la découverte de route.

Lorsque la réponse atteint l'initiateur de la découverte de route, ce dernier met à jour sa table de routage avec cette nouvelle route, qui consiste en la liste des nœuds intermédiaires avec un coût associé. Le coût sert aux nœuds à effectuer un choix entre deux routes menant à la même destination. Il peut être basé sur le nombre de nœuds intermédiaires traversés ou sur des critères plus complexes comme le débit, la fiabilité des liaisons ou la taille des paquets. Si l'initiateur reçoit ultérieurement une indication comme quoi cette destination peut-être jointe avec un coût plus faible par un autre chemin, la table sera mise à jour avec la route ayant le coût le plus faible. Une fois une route établie, un protocole de routage doit aussi mettre en œuvre un mécanisme de maintenance des routes pour gérer les évènements comme la coupure d'un lien entre deux nœuds par lesquels transitent des messages. Lorsqu'un nœud reçoit un paquet de données pour une destination vers laquelle il ne peut plus émettre, il renvoie un message d'erreur de type ***route error*** vers la source du paquet de données. La route doit alors être supprimée de la table de routage. Des optimisations existent permettant à un nœud d'écouter les routes échangées par les autres nœuds et de mettre à jour sa table de routage en conséquence.

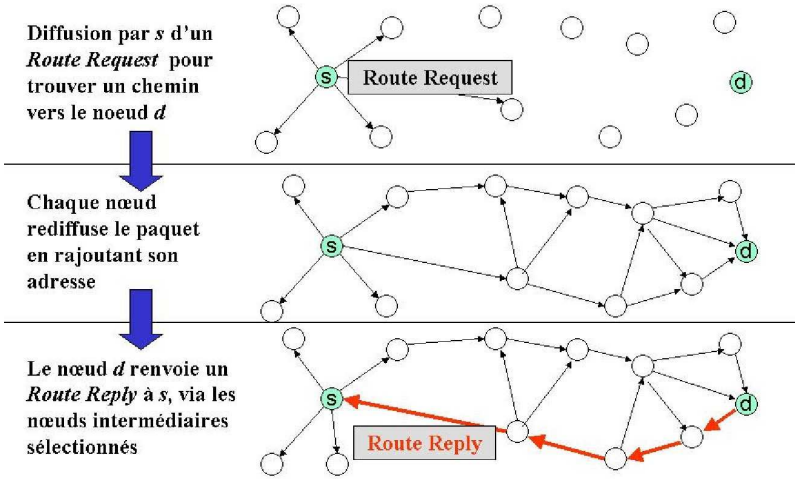


Fig. 3. Découverte de route initiée par le protocole de routage

3.3 Les Attaques Liées aux Protocoles de Routage

Si aucun contrôle n'est fait sur la provenance et l'intégrité des messages de routage du réseau ad hoc, un nœud malicieux pourra facilement causer des perturbations au réseau. Cela lui sera d'autant plus facile que les réseaux sans fil ad hoc n'ont pas de barrière physique pour se protéger et que tous les éléments peuvent potentiellement participer au mécanisme de routage.

Si un nœud malicieux a la capacité d'usurper l'identité d'un nœud valide du réseau, il peut lors du mécanisme de découverte de route répondre au nœud initiateur avec un message de type *route reply* en annonçant un chemin, avec un coût minimal, vers le nœud demandé. Le nœud émetteur mettra alors sa table de routage à jour avec cette fausse route. Les paquets de données du nœud émetteur vers le nœud destinataire transiteront par le nœud malicieux qui pourra tout simplement les ignorer. Cette attaque est appelée trou noir, ***black hole***[1]. Les paquets sont captés et absorbés par le nœud malicieux. La figure 4 illustre cette attaque.

Une variante est appelée ***grey hole***, seuls certains types de paquets sont ignorés par le nœud malicieux. Par exemple, les paquets de données ne sont pas retransmis alors que les paquets de routage le sont. Un attaquant peut aussi créer des **boucles infinies** dans le réseau ou imposer aux paquets de faire des **détours** consommant la ressource radio inutilement. Un nœud malicieux ayant usurpé l'identité d'un nœud valide peut aussi générer des messages d'erreurs de type *route error*, de manière aléatoire, pour perturber le fonctionnement du mécanisme de maintenance des routes.

[1] Cette attaque n'est pas spécifique aux réseaux sans fil, on la retrouve aussi dans les réseaux filaires

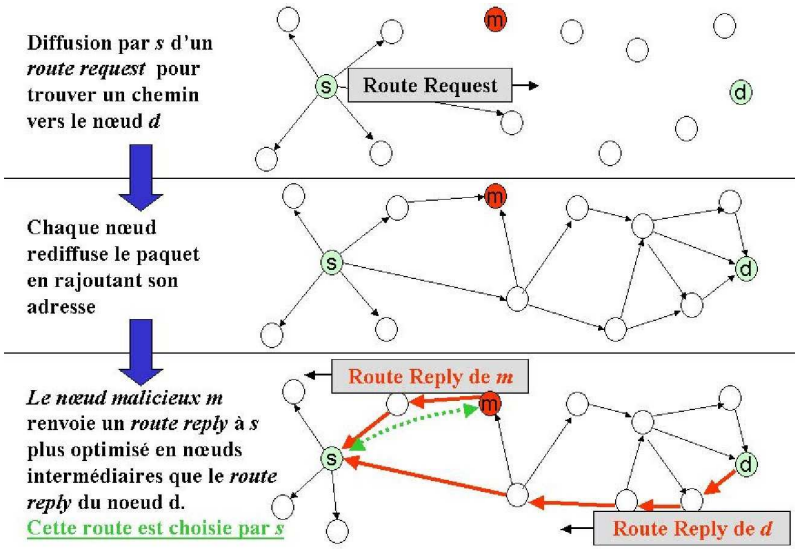


Fig. 4. Attaque *black hole* : Le nœud malicieux m capte le trafic dédié au nœud d

4 État de l'Art des Solutions

4.1 Solutions pour l'Authentification

L'absence d'infrastructure centralisée dans les réseaux sans fil ad hoc compromet l'utilisation directe des systèmes d'authentification basés sur la cryptographie à clé publique. En effet, ces systèmes d'authentification supposent l'utilisation de **certificats** établis par une **autorité centrale**. Le certificat, signé par l'autorité centrale, permet de garantir qu'une clé publique appartient bien à son propriétaire et non à un usurpateur. L'opération de vérification de certificat ne se limite pas à contrôler la signature de l'autorité centrale. Il est aussi nécessaire de s'assurer que le certificat est toujours en cours de validité et qu'il n'a pas été révoqué. Une révocation de certificat est indispensable si la clé privée du propriétaire a été volée ou divulguée. Il existe trois grands courants dans le domaine de l'authentification pour les réseaux sans fil ad hoc. Deux de ces orientations se basent sur l'établissement d'une clé secrète permettant par la suite l'authentification des participants. Toute la complexité réside en la manière d'établir cette clé. Les deux modèles basés sur une clé secrète sont :

- ***The key agreement*** : Les participants s'entendent sur une clé secrète.
- ***The Duckling Security Policy Model*** : Le modèle d'authentification élaboré par Ross Anderson et Franck Stajano [9].

Le troisième axe de recherche pour l'authentification au sein de réseaux sans fil ad hoc, se base sur la cryptographie à clé publique et cherche à s'affranchir du besoin d'une entité centrale de certification :

- **L'infrastructure à clé publique auto-organisée** : Modèle proposée par Hubaux *et al.* [5].

Clé Secrète Commune (*Key Agreement*) :

Les recherches en matière de *key agreement* dans les réseaux ad hoc se focalisent sur la manière d'établir une clé commune entre plusieurs participants qui ne se connaissent pas à priori. La mise en place de protocoles assurant l'authentification mutuelle n'est pas abordée dans le cadre de ce papier car ces problématiques ne sont pas typiques au réseau ad hoc. Les participants établissent entre eux une clé secrète leur permettant de s'authentifier afin de communiquer de manière sécurisée. La mise en place de cette clé peut se faire de manière **distribuée**. Dans ce cas, la clé secrète est fournie aux participants du réseau ad hoc via un canal supposé sûr. C'est le cas lorsque des collègues souhaitant établir une communication sûre entre eux à l'occasion d'une réunion dans une salle de conférence close, distribuent un mot de passe inscrit sur un morceau de papier qui fait le tour de la salle. Seules les personnes présentes dans la salle en ont connaissance. Une clé forte peut être dérivée du mot de passe à l'aide d'une fonction de hachage. La difficulté de ce mode de fonctionnement est de trouver un canal sécurisé pour distribuer la clé. Une autre manière d'établir une clé secrète commune est de faire en sorte que chaque participant apporte sa **contribution** à la clé finale. Lorsqu'il n'y a que deux nœuds, le protocole de **Diffie-Hellman** [2] peut être utilisé.

Méthode de Diffie-Hellman

Alice et Bob se mettent d'accord sur un entier N et un générateur α du groupe cyclique fini d'ordre N (ce groupe est constitué de tous les entiers positifs ou nul strictement inférieur à N, les calculs dans le groupe cyclique se font modulo N). Alice et Bob choisissent chacun un nombre **secret** utilisé comme exposant. Le secret d'Alice est a, et celui de Bob est b. Alice envoie alors $\alpha^a modulo(N)$ à Bob et Bob envoie $\alpha^b moduloN$ à Alice :

$$Alice \longrightarrow Bob : \alpha^a modulo(N)$$

$$Bob \longrightarrow Alice : \alpha^b modulo(N)$$

Une fois que Bob a reçu $\alpha^a modulo(N)$ de la part d'Alice, il peut utiliser son nombre secret b pour calculer :

$$(\alpha^a modulo(N))^b modulo(N) \Longrightarrow (\alpha^a)^b modulo(N) \Longrightarrow (\alpha^{a.b}) modulo(N)$$

De son côté, Alice peut calculer :

$$(\alpha^b modulo(N))^a modulo(N) \Longrightarrow (\alpha^b)^a modulo(N) \Longrightarrow (\alpha^{b.a}) modulo(N)$$

La **clé résultante**, secret partagé par Alice et Bob, sera $\alpha^{a.b} modulo(N)$. Un attaquant qui a la possibilité d'écouter les échanges entre Alice et Bob, ne pourra pas deviner la clé car il est très difficile (en terme de puissance de calcul), lorsque N, a et l'exposant sont suffisamment grands, de calculer le nombre secret a connaissant N et α. L'opération revient à calculer un logarithme discret dans le groupe cyclique fini d'ordre N.

Il faut noter que le protocole de Diffie-Hellman tel que présenté ici ne résiste pas aux attaques du type *man in the middle*, il faut donc rajouter des éléments dans les échanges permettant une authentification mutuelle des participants. Lorsqu'il y a plus de deux nœuds, le protocole de **Diffie-Hellman doit être généralisé à de multiples participants**. Chaque nœud i possède son exposant secret e_i, la clé résultante sera $\alpha^{e_1.e_2.\ldots.e_n}$ pour n participants. La mise en pratique du protocole de Diffie-Hellman à de multiples participants n'est pas si simple et fait l'objet de nombreuses recherches. En effet, il ne suffit pas que chacun envoie la valeur α^{e_i} aux autres pour que tous les nœuds aient la possibilité de déterminer la clé. Le protocole de Diffie-Hellman s'appuie

sur le fait que les participants calculent la clé à l'aide de la valeur reçue des autres participants et leur exposant secret. Il faudrait donc que le nœud i reçoive la valeur $\alpha^{e_1 \cdot e_2 \cdot \ldots \cdot e_{i-1} \cdot e_{i+1} \cdot \ldots \cdot e_n}$ pour être capable de calculer la clé. La mise en pratique peut se faire comme indiqué dans la figure 5.

1 Le premier nœud envoie α^{e_1} au second nœud

2 Le deuxième nœud envoie $\alpha^{e_1 \cdot e_2}$ au troisième nœud

3 Le (n -2)-ième nœud envoie $\alpha^{e_1 \cdot e_2 \cdot \ldots \cdot e_{n-2}}$ au (n -1)-ième

4 Le (n -1)-ième nœud envoie la valeur $\eta = \alpha^{e_1 \cdot e_2 \cdot \ldots \cdot e_{n-1}}$ à tous les autres nœuds

5 Le dernier nœud n peut alors déterminer la clé secrète partagée,

$$k = \alpha^{e_1 \cdot e_2 \cdot \ldots \cdot e_n} = \eta^{e_n}$$

6 Chaque nœud i envoie la valeur $\eta^{\frac{E_i}{e_i}}$ au nœud n, (E_i est un facteur de brouillage choisi par i)

7 Le nœud n renvoie à chacun des nœuds la valeur $(\eta^{\frac{E_i}{e_i}})^{e_n} = \eta^{e_n \cdot \frac{E_i}{e_i}}$

8 Les nœuds i enlèvent le facteur de brouillage $\frac{E_i}{e_i}$ pour retrouver la clé $k = \eta^{e_n}$

$$(\eta^{e_n \cdot \frac{E_i}{e_i}})^{\frac{e_i}{E_i}} = \eta^{e_n}$$

Fig. 5. Exemple d'application du protocole de Diffie-Hellman à de multiples participants

Cette application de Diffie-Hellman à de multiples participants a pour inconvénient de nécessiter d'établir au préalable une relation d'ordre entre les nœuds du réseau. De plus, il faut que le pénultième et l'antépénultième éléments aient la possibilité de communiquer avec tous les autres nœuds, ce qui est une contrainte beaucoup trop forte pour les réseaux sans fil ad hoc. D'autres possibilités ont été étudiées. Elles se basent sur le principe d'établir deux clés secrètes indépendamment entre deux groupes parallèles de deux entités à l'aide du protocole de Diffie-Hellman. Les deux clés secrètes sont ensuite utilisées comme exposant secret pour établir une troisième clé secrète entre les deux groupes de deux entités. La figure 6 illustre ces échanges.

Le problème est que ces méthodes exploitent des organisations particulières de réseau, en cube par exemple. Un travail supplémentaire est donc nécessaire pour trouver le moyen d'appliquer le protocole de Diffie-Hellman aux réseaux à topologie dynamique et arbitraire que sont les réseaux ad hoc. Maarit Hietalahti [3] a, au cours de sa thèse à l'université d'Helsinki, travaillé sur un protocole mettant en œuvre une généralisation du protocole de Diffie-Hellman à de multiples participants par le biais du protocole **AT-GDH**, pour *Arbitrary Topology Generalisation of Diffie-Hellman*, dont le but est de résoudre ces problèmes. La solution proposée est basée sur la construction au préalable d'une topologie de type *spanning tree* et ensuite d'appliquer le protocole de Diffie-Hellman de la même manière que précédemment en balayant l'arbre des feuilles à la racine.

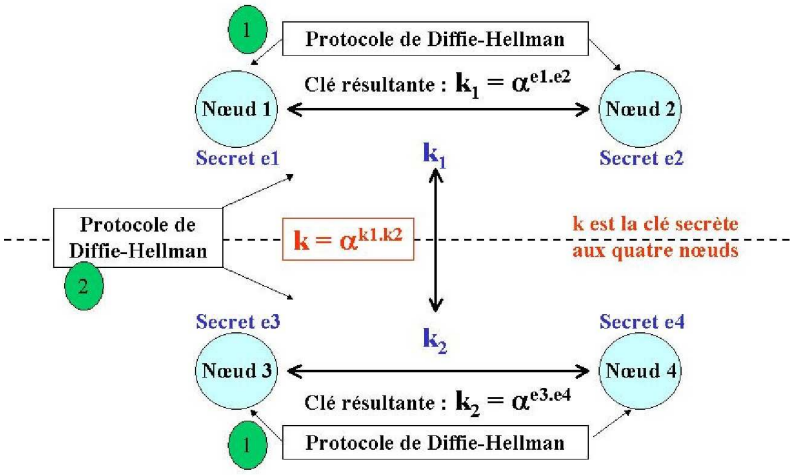


Fig. 6. Diffie-Hellman à quatre participants, configuration en cube

Relation de Type Maître-Esclave (*Duckling Security Policy Model*) :
Le travail de Ross Anderson et Franck Stajano s'inscrit dans la notion d'informatique omniprésente, ***ubiquitous computing***. Dans ce modèle, la plupart des objets nous entourant sont voués à recevoir un processeur leur permettant d'effectuer des comptes rendus de mesure (ce serait le cas pour un thermomètre dans le milieu médical par exemple) ou de rendre des services à d'autres objets de type maître comme le PDA (*Personal Digital Assistant*) d'un médecin. Les communications entre objets s'établissent via le canal radio. Le modèle d'authentification élaboré par Ross Anderson et Franck Stajano, *The Duckling Security Policy Model*, est basé sur une relation de type **maître-esclave**. Lors de sa première utilisation, un objet doit être marqué, ***imprinting***, par son propriétaire. Lors de cette opération une clé secrète est échangée entre les deux entités via un canal supposé sûr. Il peut s'agir d'un contact physique entre le maître et l'esclave. Un point important à soulever dans cette politique est la **gestion de clés**. Doit-elle se faire par une entité centralisée, qui liste les objets sous son contrôle associés à leur propriétaire et à la clé correspondante ou bien par l'utilisateur lui-même. L'option centralisée semble tentante pour éviter aux utilisateurs la charge de gestion de clés, néanmoins elle implique un fort risque de violation de la vie privée en cas de divulgation par l'entité centrale de la liste objet-propriétaire-clé.

L'Infrastructure à Clé Publique Auto-Organisée :
Le troisième courant pour l'authentification, développé par Hubaux *et al.* [5], se base sur des infrastructures à clé publique, *public key infrastructure* (PKI), autogérées au sein du réseau ad hoc. Chaque nœud établit des certificats pour les nœuds en qui il a confiance. Lorsque deux éléments d'un réseau veulent communiquer sans connaissance au préalable l'un de l'autre, ils s'échangent leur liste de certificats et vont essayer de créer une **chaîne de confiance** entre eux. Supposons qu'un élément a veuille com-

muniquer avec un nœud c, si a fait confiance en un troisième élément b et c fait aussi confiance en b, alors une chaîne de confiance entre a et c pourra être établie via b.

4.2 Solutions pour l'Intégrité Physique des Nœuds

L'intégrité des nœuds du réseau dépend fortement des capacités physiques de ce nœud à résister à des attaques qui permettraient à un attaquant de modifier le fonctionnement du nœud afin de le corrompre. Avoir la possibilité de booter sur la disquette ou le CD-ROM d'un PC en libre service dans un réseau ouvert corrompt fortement l'intégrité de ce nœud. En effet, l'OS (*Operating System*) du nœud peut alors être échangé par un OS corrompu. L'intégrité physique d'un système informatique, ***tamper resistance***, est une propriété très difficile à mettre en œuvre par les fabricants. Un moyen de contourner ce problème peut se faire en mettant en place des fonctions permettant de mettre en évidence une attaque physique sur un élément. Une telle fonction peut être comparée à un sceau assurant l'inviolabilité d'un courrier. Elle met en œuvre des notions de traçabilité de l'attaque.

4.3 Solutions pour l'Intégrité et l'Authentification des Messages

Les moyens classiques pour assurer l'intégrité et l'authentification des messages échangés par les nœuds d'un réseau sont l'utilisation de **signatures numériques** ou de **MACs** (*Message Authentication Code*). Les signatures numériques s'appuient sur la cryptographie à clé publique. Un nœud possède une clé publique qui sert à ses correspondants pour chiffrer des messages lui étant destinés et le nœud déchiffre les messages qu'il reçoit avec sa clé privée. Dans le cas de la signature, le nœud utilise une clé privée (dédiée à la signature) pour signer un message. Le destinataire du message déchiffre la signature avec la clé publique de l'émetteur et est convaincu que ce dernier est bien l'auteur du message car lui seul connaît sa clé privée. De plus, le message est bien intègre car ne connaissant pas la clé privée de l'émetteur un attaquant ne pourra pas modifier le message original. L'utilisation de la cryptographie asymétrique n'est pas une solution préférée dans les réseaux ad hoc car certains nœuds peuvent avoir des capacités de calcul réduites. Or, la cryptographie asymétrique demande plus de temps de calcul que la cryptographie symétrique.

Une autre solution consiste à utiliser des MACs. Il s'agit de fonctions mathématiques à sens uniques dépendant d'une clé secrète qui, à partir du message original, produisent un condensé de ce message. Ces fonctions mathématiques sont telles qu'il est difficile de retrouver un message à partir de son condensé ou de produire deux messages ayant le même condensé. De plus, la moindre modification du message original entraîne un changement dans le condensé. L'inconvénient de cette solution est qu'il est nécessaire au préalable d'établir autant de clés secrètes que de paires de nœuds susceptibles de vouloir communiquer ensemble. D'autres solutions doivent donc être envisagées pour les réseaux ad hoc.

TESLA (*Time Efficient Stream Loss-tolerant Authentication*) a été proposé par Perrig *et al.* [7]. Il s'agit d'une extension au protocole décrit en [1], dit *Guy Fawkes' protocol*. TESLA permet d'authentifier les messages avec un MAC dépendant d'une clé secrète qui n'est divulguée par l'émetteur du message qu'après un délai d'attente δ. La valeur δ est calculée de manière à ce qu'on soit sûr que le destinataire a reçu le message avant la divulgation de la clé, cette condition garantie l'intégrité du message. Le temps δ ne doit pas être trop important pour limiter les latences dans le réseau,

en effet un destinataire doit attendre la divulgation de la clé secrète avant de pouvoir effectivement traiter un message. La figure 7 décrit le fonctionnement de TESLA.

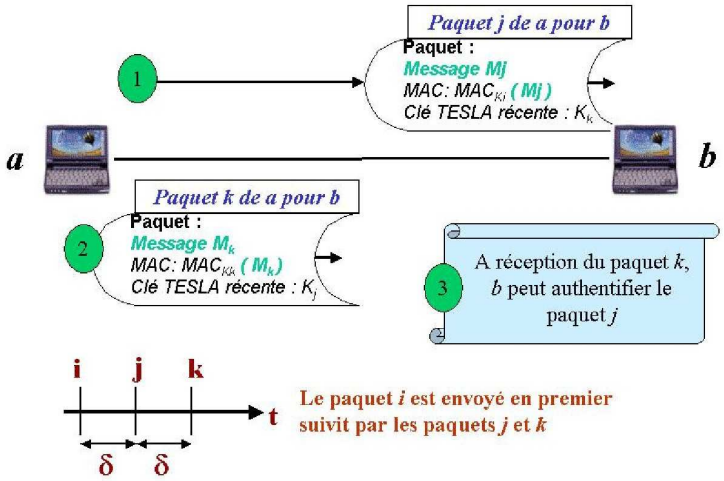


Fig. 7. Authentification de messages avec TESLA. Les messages M_i sont authentifiés à l'aide d'un MAC dont la clé k_i est divulguée dans un paquet subséquent

La clé secrète utilisée pour le MAC est issue d'une chaîne de clés. Un élément de la chaîne k_i est calculé de la manière suivante : $k_i = \hbar(k_{i+1})$ où $\hbar$ est une fonction de hachage. L'élément initial k_n est choisi par l'émetteur. Celui-ci va utiliser ces clés par ordre croissant c'est à dire en commençant par k_1. En réception, le destinataire pourra vérifier la relation suivante : $k_{i-1} = \hbar(k_i)$ où k_i est la clé dernièrement reçue et k_{i-1} correspond à la clé précédente. Cette condition assure que la clé k_i fasse bien partie de la chaîne de clé de l'émetteur, ce qui assure, en plus de l'intégrité, la propriété d'authentification du paquet. Il est à noter que ce processus doit être initialisé par l'authentification du premier paquet émis à l'aide d'une signature numérique.

Un exemple de protocole de routage sécurisé utilisant TESLA :

Une manière de contrer les attaques sur les mécanismes de routage consiste en l'authentification des messages de découverte et de maintenance de route. Hu, Perrig et Johnson ont développé un protocole de routage, **Ariadne** [4], basé sur le protocole DSR et qui implémente des mécanismes d'authentification des messages de routage en utilisant au choix un schéma de signature numérique, l'utilisation de MAC avec autant de clés secrètes établies que de paires de nœud ou bien en mettant en œuvre TESLA. Leur article se focalise sur cette dernière solution. Une conclusion intéressante de leur travail montre que l'introduction de mécanismes de sécurité dans un protocole de routage passe nécessairement par une diminution des performances de ce protocole. En effet, les entêtes des paquets de routage voient leur taille augmenter avec l'authentification des messages. De plus, une baisse de réactivité dûe au temps d'attente pour authentifier un message apparaît. Un compromis entre niveau de sécurité et per-

formance est donc clairement nécessaire. Il faut rester prudent avec les mécanismes d'authentification de messages. En effet, un attaquant peut générer une multitude de paquets sur le réseau avec une fausse authentification. Les nœuds sont alors submergés par un nombre trop important de paquets non-valides à authentifier, ce qui ralentit le fonctionnement du réseau jusqu'à provoquer un déni de service. Il est aussi important de prévoir des mécanismes permettant de se prémunir contre un attaquant cherchant à rejouer des messages authentifiés.

4.4 Solutions pour la Confidentialité

La confidentialité dans les réseaux sans fil ad hoc est d'abord traitée par l'utilisation de transmission par saut de fréquences, *frequency hoping*. Les données sont transmises sur une séquence de fréquences définies pseudo-aléatoirement. L'attaquant doit connaître cette séquence pour pouvoir se synchroniser en réception. Une fois l'authentification des participants clairement établie, les outils cryptographiques permettent de rendre les communications confidentielles. Toutefois, étant donné qu'une des contraintes des réseaux ad hoc est de devoir être adaptables à des nœuds ayant de faibles capacités de calcul, la cryptographie symétrique sera préférée à la cryptographie à clé publique, cette dernière nécessitant beaucoup plus de puissance de calcul.

4.5 Solutions pour l'Anonymat

L'anonymat est très difficile à obtenir dans des réseaux coopératifs. Il est préfèrable de parler de pseudo-anonymat. L'identité d'un participant est associée à un code à la manière d'un pseudonyme. Une autorité centrale de confiance est nécessaire pour stocker de manière sécurisée la correspondance identité / code.

4.6 Solutions pour la Disponibilité

Il n'existe aucun moyen de contrer un déni de service sur le canal radio provoqué par un attaquant puissant ayant les moyens de brouiller efficacement la totalité du spectre radio. Néanmoins, des techniques comme le saut de fréquence permettent de se prémunir contre des attaquants ayant des capacités plus réduites. En effet, ces techniques permettent une transmission des données sur un large spectre de fréquence. Pour être efficace un attaquant doit donc être capable de brouiller l'étendue des fréquences utilisées.

4.7 Solutions Complémentaires

Pour contrer les attaques sur les mécanismes de routage de type *black hole*, où un nœud malicieux prétend être un relais pour un autre nœud mais ne transmet pas les messages de données, Marti *et al.* [6] ont développé deux méthodes appelées ***watchdog*** et ***pathrater***. Le *watchdog* permet d'identifier les nœuds malicieux. Le *pathrater* est une technique permettant au protocole de routage d'éviter les nœuds corrompus inscrits dans une liste noire, *blacklist*. Il faut rester prudent quant à l'utilisation de ces mécanismes car ils peuvent être détournés par un attaquant. En effet, un nœud malicieux peut aussi faire en sorte qu'un nœud valide soit ajouté à la liste noire, l'isolant ainsi du réseau. L'utilisation de détecteurs d'intrusion dans les réseaux ad hoc est une

solution complémentaire faisant l'objet de recherches intensives. L'**IDS** (***Intrusion Detection System***) collecte et analyse les données du trafic afin de déterminer si des utilisateurs non autorisés sont connectés ou si certains nœuds ont des comportements anormaux. Un axe de recherche consiste à étudier la manière dont les protocoles de routage pourraient utiliser ces informations pour prévenir certaines attaques.

5 Conclusion

Cet article montre à quel point les réseaux sans fil ad hoc constituent, de par leur nature, un formidable challenge pour la sécurité informatique. Les spécificités de ces réseaux sont principalement :

- La transmission en milieu ouvert.
- Les topologies dynamiques.
- L'absence d'autorité centrale.
- La nécessité de bonne coopération des nœuds.
- L'hétérogénéité des participants avec pour certains des capacités restreintes.

Toutes ces contraintes concourent à rendre la sécurité des réseaux sans fil ad hoc difficile et complexe à appréhender. Ce sujet va devenir d'autant plus critique que le développement de tels réseaux va rapidement s'amplifier. En effet, les réseaux sans fil ad hoc sont stimulés par l'évolution rapide des technologies informatiques vers la miniaturisation et l'intégration. L'authentification des nœuds et des messages échangés, constitue le point de départ incontournable pour la sécurité des réseaux ad hoc. Il existe de nombreuses études théoriques mais finalement peu d'applications pratiques qui puissent satisfaire l'ensemble des contraintes inhérentes aux infrastructures sans fil ad hoc. Enfin, il apparaît clairement que les mécanismes de routage constituent un point sensible pour la sécurité des réseaux sans fil ad hoc. La profusion de propositions diffusées au sein du groupe de travail MANET de l'IETF montre clairement le manque de maturité sur ce sujet. Un risque est que l'accent soit mis sur la résolution des problèmes fonctionnels au détriment de la sécurité. Les deux axes de recherche que sont l'authentification des nœuds et messages d'un côté, les mécanismes de routage sécurisés de l'autre, apparaissent comme des directions de travail primordiales. La prise en compte de ces deux problématiques doit se faire rapidement afin d'assurer un déploiement des réseaux sans fil ad hoc fiables et sécurisés.

Remerciements :

Les auteurs tiennent à remercier Jean-Marie Bonnin et Bruno Stevant de l'ENST Bretagne pour leur participation à la mise en place des manipulations. Merci à Eric Diehl, Olivier Heen et Nicolas Prigent du Security Lab de Thomson pour leurs conseils avisés et leur précieuse contribution à la préparation de ce papier et de la présentation. Nous souhaitons aussi remercier Thierry Martineau et Franck Veysset pour leur relecture du document et les remarques constructives qui ont suivi.

Références

1. Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas and Roger Needham, A new family of authentication protocols, *ACM SIGOPS Operating Systems Review*, Vol. 32, no. 4, pp. 9–20, 1998.

2. Whitfield Diffie and Martin E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, IT-22, no. 6, pp. 644–654, 1976.
3. Maarit Hietalahti, Efficient Key Agreement for Ad Hoc Networks, Ph. D, Helsinki University of Technology, May 2001.
4. Yih Chun Hu, Adrian Perrig and David B. Johnson, Ariadne : A secure on-demand routing protocol for ad hoc networks, *Proceedings of The 8th ACM International Conference on Mobile Computing and Networking*, 2002, *citeseer.nj.nec.com/hu02ariadne.html*
5. Hubaux, J. P., Buttyan, L. and Capkun, S., The Quest for Security in Mobile Ad Hoc Networks, *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, 2001.
6. , Sergio Marti, T.J Giuli, Kevin Lai and Mary Baker, Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks,
7. Adrian Perrig, Ran Canetti, J.D. Tygar and Dawn Song, Efficient Authentication and Signing Multicasts Streams over Lossy Channels, *IEEE Symposium on Security and Privacy*, September 2002.
8. Frank Stajano, *Security for Ubiquitous Computing*, John Wiley and Sons, 2002, *ISBN 0-470-84493-0, http ://www-lce.eng.cam.ac.uk/ fms27/secubicomp/*
9. Frank Stajano and Ross Anderson, The Resurrecting Duckling : Security Issues for Ad-hoc Wireless Networks, *7th International Workshop on Security Protocols*, pp. 172–194, 1999.

JAB, une backdoor pour réseau Win32 inconnu

Nicolas Grégoire

Exaprobe
ngregoire@exaprobe.com,
WWW home page : http ://www.exaprobe.com

1 Introduction

Le but de cet article est de montrer les possibilités offertes par Internet Explorer et les objets OLE rattachés dans le cadre de la prise de contrôle à distance d'une machine Windows située dans un réseau totalement inconnu. Nous en profiterons pour présenter JAB, une backdoor utilisant Internet Explorer pour établir un canal de commandes et de données entre la machine compromise et l'attaquant.

Nous verrons tout d'abord les moyens employés jusqu'à présent pour piloter une backdoor Win32, ainsi que les différents outils exploitant la technique de manipulation d'objets OLE utilisée par JAB. Après avoir rapidement examiné le code Perl nécessaire à une requête HTTP minimaliste (GET statique) et donc survolé les fonctions intéressantes du paquetage `Win32 : :OLE`, nous aborderons le cycle de vie de la backdoor, depuis le choix des cibles et des vecteurs d'attaques jusqu'à la gestion de multiples instances via le serveur JABd.

Nous détaillerons ensuite le fonctionnement interne de JAB, dont principalement l'API de communication client/serveur et les techniques de transfert bidirectionnel de données binaires. Nous finirons par une présentation du mode *"command-line"* de la backdoor (utile pour le débugage et l'exfiltration de documents) et des limitations inhérentes à ce type de backdoor.

2 Techniques de contrôle distant de backdoors pour Windows

2.1 Historiques des moyens de contrôle

Le code-source de Windows n'étant pas publiquement disponible, l'évolution des moyens permettant de contrôler à distance une machine compromise est sensiblement différente de celle rencontrée sous Unix. En effet, très peu de *"magic flags"* comme ceux utilisés pour `/bin/login` sous Unix (variables d'environnement, nom d'utilisateur) existent, l'attention s'étant portée sur la fourniture d'un shell (`cmd.exe`, `command.com`) à l'attaquant. La première technique employée est la connexion d'un shell (ou du canal de commande pour *BO2K*) sur un port haut de la machine compromise, mais cela impose que cette machine soit accessible directement par l'attaquant (pas de "masquerading") et qu'il n'y ait pas filtrage INBOUND sur le port employé.

Ces conditions étant rarement réunies dans le cadre d'un réseau interne d'entreprise, la technique suivante consista à initier un reverse-shell directement vers une machine contrôlée par l'attaquant. Le seul pre-requis étant l'existence d'au moins un flux autorisé entre le LAN et l'Internet, cette technique est utile si nous ne sommes pas confrontés

à un réseau "strict", c'est-à-dire où les connexions vers l'extérieur doivent passer par des proxys, soit à des fins de lutte anti-virale, soit pour comptabiliser/authentifier les accès sortants.

Dans le cadre de ces réseaux "stricts" employant des proxys, l'attaquant cherche souvent à extraire depuis la base de registre les informations de connexion, afin de les utiliser pour établir son propre canal sortant. Mais plusieurs limitations apparaissent : le proxy peut nécessiter une authentification interactive, ou un firewall personnel peut détecter la tentative d'établissement de connexion via le proxy.

Pour contourner ces limitations, le moyen le plus intuitif est d'arriver à accéder au réseau via un processus autorisé aussi bien au niveau du firewall personnel que du proxy. C'est ainsi que nous avons vu apparaître des attaques portant sur l'injection de code dans l'espace mémoire de processus autorisés, ceci permettant de contourner les firewalls personnels. Mais ce type d'attaque reste assez technique, et n'est pas une solution simple (en terme de code) pour établir un véritable canal de communication entre la machine compromise et l'attaquant.

2.2 Outils utilisant IE/OLE

La première évocation publique de l'utilisation de Internet Explorer et des contrôles OLE pour piloter à distance une machine Windows a été le document "Hacking Guide" de SensePost (p.80) [1].

Depuis, plusieurs outils ont été développés dans ce sens, dont Setiri (par SensePost) [2] et Deep-Pentest (par HSC) [3]. Le premier a été présenté à Defcon 10, et le deuxième semble réservé à l'usage interne de HSC, et n'a donc pas été publié. Il apparaît donc que l'emploi d'Internet Explorer comme vecteur de communication pour une backdoor se développe, bien que nous n'ayons pas pour l'instant de trace d'outil "black hat" employant cette technique.

Le fonctionnement de JAB repose sur un client (situé du côté de la victime) et un serveur (situé côté pen-tester). Ces deux éléments sont codés en Perl, ceci permettant d'utiliser `pack()` pour la conversion de données et `eval()` pour l'interprétation à la volée de code fourni par le serveur. Ceci impose donc la transformation du script contenant le code "client" en exécutable (dans notre cas via *Perl2EXE*), le binaire obtenu étant d'une taille relativement conséquente (environ 900 Ko).

Les objectifs de JAB sont doubles :

- permettre le contournement de la majorité des protections pouvant exister dans un réseau d'entreprise, ceci justifiant l'emploi de Internet Explorer via les contrôles OLE comme moyen de communication,
- faciliter le travail côté pen-tester, c'est-à-dire principalement fournir une gestion aisée et asynchrone d'instances multiples. Ceci permet par exemple de contrôler des machines situées dans un fuseau horaire différent de celui du "maître", les ordres pouvant être récupérés sur le serveur sans intervention humaine.

Nous verrons par la suite les choix conceptuels et d'implémentation permettant de respecter ces objectifs.

3 Manipulation d'objets OLE, dont Internet Explorer : détails

Étant donné qu'un court morceau de code est parfois beaucoup plus explicite qu'un long discours, voici un programme minimaliste réalisant une requête statique :

```
use strict;
use Win32::OLE;

my $url = "http://www.exaprobe.org/owned.html?jobs_at_victime.fr"";
my $timeout = 60; my $time = 0;

# Creation de l'objet OLE
my $ie=Win32::OLE->new('InternetExplorer.Application') or die $!;
$ie->{Visible} = 0; $ie->{Offline} = 1; $ie->{Silent} = 1;

# Accès à l'URL
$ie->Navigate2($url,2);
while ($time <= $timeout) {
     Win32::Sleep(1000); $time++;
     last unless $ie->{Busy};
}
```

Ce type de code permet donc d'identifier les adresses email dont les destinataires ont exécuté notre code. Aucun "démon" n'est nécessaire côté serveur, une simple lecture des logs du serveur HTTP permettant de lister les adresses email ainsi remontées.

4 Cycle de vie de la backdoor

Cette partie décrit les différentes phases prenant place entre la décision d'attaquer une entité particulière et la fin de vie des processus constitués par les backdoors déployées.

4.1 Génération des binaires

Deux choix très importants pour la suite des évènements doivent être faits :
- allons-nous tenter de faire exécuter notre code via une faille de navigateur, une faille de client de messagerie ou de l'ingénierie sociale ?
- en cas d'envoi par mail, les destinataires finaux sont-ils clairement identifiés ou écrivons-nous à un alias de type "*tous@victime.fr*" ?

Des réponses apportées à ces questions, nous pourrons déduire l'icône la plus adaptée à la situation (Flash, Zip, Word) ainsi que le schéma de génération des identifiants (aléatoire dans le cadre d'un alias de messagerie global, fixe si très peu de comptes sont visés et que l'on désire une certaine traçabilité de nos binaires).

4.2 Déploiement des binaires

Dans le cas d'une attaque par ingénierie sociale, il suffit de composer un mail incitant son destinataire à exécuter le fichier présent en pièce jointe. Dans le cadre d'une exploitation de faille de navigateur, nous pouvons utiliser un outil comme IEsploit.tcl [4] pour compromettre chaque machine Windows non patchée accédant au site Web ainsi piégé.

4.3 Exécution de notre code offensif

Une fois notre binaire lancé par la victime, les actions d'initialisation suivantes sont entreprises :
- dépôt d'un "flag" permettant de prouver la compromission de la machine
- vérification de la connectivité Internet (*http ://www.google.fr*)
- enregistrement auprès du serveur JABd

On entre ensuite dans la boucle principale du programme, celle-ci consistant à :
- récupérer auprès du serveur JABd une liste d'ordres
- interpréter chaque ligne de la liste (cf. l'API pour plus de détails)
- exécuter l'ordre correspondant

4.4 Gestion côté serveur

La personne en charge du contrôle des backdoors peut définir des actions par défaut, consulter les informations remontées (fichiers, résultat d'exécution de commandes) ou décider de copier sur le poste de la victime des fichiers permettant par exemple d'y augmenter notre niveau de privilèges. La gestion asynchrone de la prise d'ordres et de la remontée d'informations permet au gestionnaire de ne pas faire que ça de ses journées, mais aussi de prendre un repos bien mérité même si les backdoors en cours de déploiement sont situées dans plusieurs fuseaux horaires.

5 Fonctionnement interne du client

5.1 API de communication "serveur vers client"

Nous listons ici les ordres pouvant être transmis au client, avec le type de paramètres attendus et la signification de chacune de ces commandes :

- SLEEP : endort le processus distant

  paramètres : nombre d'unité à attendre et unité de temps employée (1s, 30m, 12H)
- DIE : tue le processus distant

  pas de paramètre
- EXEC : exécute une commande DOS et envoie le résultat au serveur

  paramètres : commande à passer à "`CMD /C`"
- EVAL : télécharge du code Perl, l'exécute et envoie le résultat au serveur

  paramètres : nom du fichier contenant le code côté serveur
- DWLD : copie un fichier sur le disque de la victime

  paramètres : nom du fichier distant, nom du fichier local
- UPLD : copie un fichier vers le serveur JABd

  paramètres : nom du fichier local, nom du fichier distant
- CMD : télécharge un nouveau fichier d'ordres, les interprète puis les exécute

  paramètres : nom du fichier contenant les ordres côté serveur

5.2 Transfert de données

Pour le transfert "serveur vers client", un simple GET est suffisant. Les données à transférer sont UUencodées, certains caractères interprétables par Internet Explorer étant ensuite transformés en leur équivalent HTML ('`&`' converti en '`&amp ;`').

Pour le transfert "client vers serveur", les données sont transformées selon le principe vu ci-dessus, mais la remontée d'information est faite via des formulaires utilisant la méthode POST et soumis par un code Javascript `onLoad()`. Le découpage des données en fonction de la taille maximale d'un formulaire est fait automatiquement par le client, tout comme est fait automatiquement le réassemblage côté serveur.

5.3 Mode CLI

Le fait de lancer la backdoor avec en paramètre la chaîne "`--top_secret_option`" permet d'entrer en mode interactif. Ce mode permet de visualiser la configuration de la backdoor (URL du serveur JABd, identifiant, ...) et d'accéder à l'API comme si les ordres étaient communiqués depuis le serveur. Ce mode est principalement utile pour le débogage, ainsi que pour l'exfiltration de documents depuis une machine sur laquelle nous bénéficions d'un accès physique ou interactif (VNC, Terminal Server).

5.4 Limitations de JAB

Nous détaillons ici les différentes limitations actuelles de JAB, qu'elles soient liées à des choix conceptuels (utilisation de IE/OLE) ou d'implémentation.

La principale limitation liée à l'utilisation d'Internet Explorer comme moyen de communication est la nécessité de pouvoir accéder au Web soit :

- de manière directe
- via un proxy sans authentification
- via un proxy avec authentification transparente (par IP source, par NTLM)
- via un proxy avec authentification interactive et sur lequel l'utilisateur ciblé est déjà loggué

Les limitations liées à l'implémentation ou aux moyens mis en oeuvre sont :

- absence de persistance de la backdoor en cas de reboot (la mise en place de ce type de persistance imposerait des modifications en base de registres ou dans les fichiers de démarrage)
- pas de HTTPs (besoin d'un certificat reconnu par Internet Explorer pour éviter une popup demandant d'accepter le certificat proposé)
- authentification faible des clients sur le serveur (basée sur une clé statique transitant en clair sur le réseau)
- absence de chiffrement applicatif des données transférées (un simple UUencodage, légèrement modifié, est réalisé)

6 Conclusion

Il ressort de cet article que contre ce type d'attaque, les défenses traditionnelles sont inefficaces car la backdoor apparaît comme l'usage légitime d'un navigateur Web par un utilisateur autorisé. De plus, l'avenir nous promettant une interaction encore plus grande entre le système d'exploitation et les applications, les moyens d'accéder

à Internet via un autre processus se multiplieront. Les backdoors se tourneront donc soit vers un accès de très bas niveau (communication directe avec les drivers de la carte réseau) soit vers un accès par rebond à travers une application de confiance (contrôle externe via une API quelconque, injection en mémoire). Dans les deux cas, les défenses situées sur le poste de travail (anti-virus et firewalls personnels) devront évoluer fortement afin de limiter la menace.

Une des meilleures solutions est donc prévenir l'exécution initiale de la backdoor, c'est-à-dire filtrer les documents exécutables à l'entrée du réseau d'entreprise, patcher les machines (même "end user") malgré la charge de travail induite, et surtout éduquer les utilisateurs pour les sensibiliser quant à leur rôle à jouer dans l'application de la politique globale de sécurité. D'autres solutions, comme l'utilisation de "white lists", permettraient de restreindre les sites Web pouvant être accédés par les utilisateurs. Mais il faut garder à l'esprit que cette solution est très lourde et que l'accès par une backdoor IE/OLE à un site de type Caramail permettrait de toute façon la remontée de données via des messages émis à travers le webmail.

Références

1. Roelof W. Temmingh, SensePost : A guide for breaking into computer networks from the Internet
   http ://packetstormsecurity.org/papers/general/hackingguide3.1.pdf
2. SensePost : Defcon 10 Presentation of Setiri
   http ://packetstormsecurity.org/defcon10/dc10-sensepost-setiri.ppt
3. HSC : Deep-PenTest tool
   http ://www.hsc.fr/ressources/presentations/meleenumerique/img8.html
4. Truff, Projet7 : Upload and code execution on IE
   http ://projet7.tuxfamily.org/factory/exploits/IEsploit.tcl

Faiblesses des protections d'exécutable PE. Étude de cas : Asprotect

Nicolas Brulez

Cartel Sécurité
brulez@cartel-securite.fr
`http ://www.cartel-securite.fr`

1 Introduction

Asprotect est une protection d'exécutable PE. (Fichiers Windows 32 bit). Elle ne nécessite aucune modification du code source et travaille directement sur le binaire. Le fichier original est compressé, puis chiffré, mais reste néanmoins exécutable. Le désassemblage devient inutile puisque le code du programme est protégé. La protection utilise des techniques d'anti-debugging pour rendre l'analyse plus complexe.

L'étude de la protection complète a duré 2 heures. Je ne présenterai ici que les points importants de la protection et comment la retirer. Je donnerai bien sûr des explications sur son principe de fonctionnement et détaillerai les différentes protections rencontrées à savoir :

- comment trouver l'entry point original ;
- la protection des imports et ses faiblesses ;
- les *Stolen Bytes* et comment les retrouver ;
- la protection contre les breakpoints.

À la fin de ce tutoriel, nous obtiendrons un programme déprotégé permettant ainsi le désassemblage et l'analyse statique du binaire.

2 Présentation du format PE

Le format PE[1] est le format des exécutables windows 32 bit. Je vais présenter rapidement son architecture, ses structures afin d'aider à la compréhension de ce tutoriel. Pour plus d'information, lire une documentation plus complète sur le format PE [1].

Un fichier peut être représenté selon la figure 1.

2.1 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 ................
```

[1] Portable Executable

```
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....$.......
```

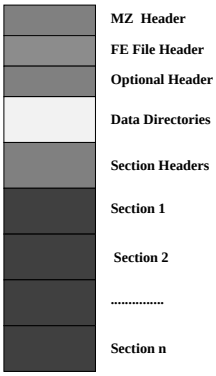


Fig. 1. Structure d'un fichier exécutable type PE

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`.

2.2 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;
```

2.3 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 *RVA*[2] 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`.

2.4 Le *DataDirectory*

Il s'agit d'un tableau de structures contentant des informations diverses telles que l'*import table*, l'*export table*, etc.

2.5 Les *Section Headers*

Ils sont accessibles à l'adresse *PE optional header* + `F8h`. Chaque section de l'exécutable est décrite par la structure suivante :

[2] Relative Virtual Address

```
+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)
```

3 Unpacking de l'exécutable

3.1 Trouver le point d'entrée

Comme nous venons de voir dans la présentation du format PE, il existe un champ de la structure *Optional Header* qui contient l'adresse relative du point d'entrée du programme, c'est-à-dire l'adresse de la première instruction à exécuter. Une fois protégé, notre programme n'aura plus le même point d'entrée. Celui-ci pointera vers le début de la protection. Pour différencier le point d'entrée du programme une fois protégé, et le point d'entrée originel du programme avant protection, on utilise de le terme *Original Entry Point* ou *OEP*.

La protection va d'abord déchiffrer puis décompresser le programme original en mémoire et donner la main au programme original en sautant à l'*OEP*. Le principe de fonctionnement est très similaire à celui des virus.

La première chose à faire lorsque l'on souhaite retirer une protection PE, est de déterminer l'*OEP*.

Pour Asprotect, il existe une méthode relativement simple que je vais présenter. Pour trouver l'*entry point* nous allons utiliser un débogueur. J'ai choisi *Ollydber* car celui-ci est gratuit, open source et contient de nombreuses options très intéressantes pour l'analyse d'une protection PE.

Commençons l'analyse. Premièrement, nous allons lancer le fichier protégé par Asprotect avec *Ollydbger* pour stopper au point d'entrée :

```
00401000  68 01706200    PUSH SystemCl.00627001
00401005  E8 01000000    CALL SystemCl.0040100B
0040100A  C3             RETN
0040100B  C3             RETN
```

Ici commence la protection.

```
00627001   60               PUSHAD
00627002   E8 03000000      CALL SystemCl.0062700A
00627007   E9 EB045D45      JMP 45BF74F7
0062700C   55               PUSH EBP
0062700D   C3               RETN
```

Après quelques minutes de traçage, on s'aperçoit que tracer toute la protection prendra énormement de temps. Afin d'éviter de perdre du temps, je décide de regarder les sections de mon programme à l'aide d'un éditeur de fichiers PE et j'obtiens ceci :

```
 Nb Sections: 11
Size of Code: 17E400
 Entry Point: 1000
  Image Base: 400000

        Vsize:17F000 RVA:1000   Psize:8B200 offset:400    Flags:C0000040
        Vsize:A000   RVA:180000 Psize:5600  offset:8B600  Flags:C0000040
        Vsize:15000  RVA:18A000 Psize:0     offset:90C00  Flags:C0000040
        Vsize:4000   RVA:19F000 Psize:3800  offset:90C00  Flags:C0000040
        Vsize:1000   RVA:1A3000 Psize:200   offset:94400  Flags:C0000040
        Vsize:1000   RVA:1A4000 Psize:0     offset:94600  Flags:C0000040
        Vsize:1000   RVA:1A5000 Psize:200   offset:94600  Flags:C0000040
        Vsize:18000  RVA:1A6000 Psize:0     offset:94800  Flags:C0000040
.rsrc   Vsize:69000  RVA:1BE000 Psize:12A00 offset:94800  Flags:C0000040
.data   Vsize:13000  RVA:227000 Psize:12E00 offset:A7200  Flags:C0000040
.adata  Vsize:1000   RVA:23A000 Psize:0     offset:BA000  Flags:C0000040
```

Le point d'entrée d'un programme se trouve en général dans la premiere section, c'est-à-dire la section `code`.

Avant d'utiliser le débogueur pour surveiller les appels à la première section, nous allons voir si la protection permet l'utilisation d'un débogueur simple, ou si celle-ci contient du code anti-débugging. Il est possible que celle-ci génère des exceptions pour ralentir l'analyse, accéder au *context*[3] de l'application et changer son déroulement, voire d'effacer les *debug registers*.

Context Structure

```
+0 context flags (utilisé lors de l'appel à GetThreadContext)

DEBUG REGISTERS
+4 debug register 0
+8 debug register 1
+C debug register 2
+10 debug register 3
+14 debug register 6
+18 debug register 7

FLOATING POINT / MMX registers
+1C ControlWord
+20 StatusWord
+24 TagWord
+28 ErrorOffset
+2C ErrorSelector
+30 DataOffset
+34 DataSelector
```

[3] Lorsque le système change de thread, tous les registres du thread en cours sont enregistrés dans la structure *context*. Chaque thread a sa propre structure. Cela permet au système de revenir à l'état où il était avant de changer de thread en restaurant les registres précédemment sauvegardés.

```
+38 FP registers x 8 (10 octets chaqun)
+88 CrONpxState

SEGMENT REGISTERS
+8C gs register
+90 fs register
+94 es register
+98 ds register

ORDINARY REGISTERS
+9C edi register
+A0 esi register
+A4 ebx register
+A8 edx register
+AC ecx register
+B0 eax register

CONTROL REGISTERS
+B4 ebp register
+B8 eip register
+BC cs register
+C0 eflags register
+C4 esp register
+C8 ss register
```

La touche `F9` permet de lancer l'application sans passer dans le mode pas à pas. Immédiatement après, le débogueur se bloque et nous sommes en présence d'une exception. Celle-ci est délibérément créée par la protection et cette pratique est très courante dans les protections d'exécutable.

Le code contient de l'obscurcissement et rend la compréhension du code plus compliquée. La présence de sauts au milieu d'instructions existantes fait changer le code lors du traçage.

Voici le code éclairci :

```
016D33C6   E83A000000     CALL 016D3405
        ; L'adresse de retour sert d'except handler
016D33CB   68D4336D01     PUSH 16D33D4
        ; Adresse de retour : 016D33CB
```

D'après le code en `16D3405`, la protection se sert de l'adresse de retour (`016D33CB`) comme *exception handler* :

```
016D3405   31C0           XOR EAX,EAX                ; EAX = 0
016D340B   64FF30         PUSH DWORD PTR FS:[EAX]
016D3411   648920         MOV DWORD PTR FS:[EAX],ESP
        ; met en place le SEH
016D3414   3100           XOR DWORD PTR DS:[EAX],EAX
        ; Génère une Exception
016D3419   648F0500000000 POP DWORD PTR FS:[0]
```

Le code suivant génère une exception en essayant d'écrire à l'adresse `00000000`. La protection utilisant un *SEH*[4], le programme exécute notre *exception handler*.

Exception Handler

```
016D33CB   68D4336D01     PUSH 16D33D4
       ; met l'adresse 16D33D4 sur la pile
016D33D0   FF0424         INC DWORD PTR [ESP]
       ; adresse = adresse+1
016D33D3   C3             RETN
       ; saute à l'adresse 16D33D4+1
016D33D4   DB             0xBC
       ; Sert seulement à tromper le désassembleur
```

Le programmeur a utilisé une petite astuce de programmation assembleur pour tromper les désassembleurs. Une adresse est poussée sur la pile, puis ensuite incrémentée. Un `RET` nous renvoie à l'adresse pointée sur la pile, c'est-à-dire celle qui vient juste d'être incrémentée. L'exécution continue finalement en `016D33D5`.

```
016D33D4   BC8B44240C     MOV ESP,0C24448B
```

Le `BC` est interprété par le désassembleur du débogueur comme un `MOV ESP`, alors que le programme va continuer son exécution un octet après, assemblant le nouveau code suivant :

```
016D33D4   BC             INVALID
       ; Octet inutile
016D33D5   8B44240C       MOV EAX,DWORD PTR [ESP+C]
       ; EAX = Context Structure
016D33DC   8380B800000002 ADD DWORD PTR [EAX+B8],2
       ; Context.EIP = Context.EIP+2
```

Le programme accède à la structure `Context`. `B8` représente l'offset de la sauvegarde du registre EIP au moment de l'exception. En ajoutant `2` à cette sauvegarde, le programme ajuste l'EIP de deux octets et le programme reprendra en *EIP sauvé*+2, soit l'instruction après celle qui a déclanché l'exception.

Pourquoi deux octets ? Tout simplement car l'instruction qui a génèré une exception faisait deux octets :

```
016D3414   3100           XOR DWORD PTR DS:[EAX],EAX
```

Après avoir mis à zéro le registre `EAX`, le programme reprend la main :

```
016D33FF   31C0           XOR EAX,EAX ; EAX = 0
016D3401   C3             RETN        ; On sort du Except handler
```

Comme nous venons de voir, le programme crée une exception pour accéder à la structure *Context*. Il modifie l'EIP du programme pour modifier son déroulement. L'interêt de ce code est de bloquer les personnes n'ayant jamais rencontré d'*exception handler* en assembleur et de déstabiliser certains traceurs/débogueurs. Il est important de noter

[4] Structured Exception Handler

que la protection accéde aux *debug registers* et les met à zero pour effacer les points d'arrêt matériels.

Olly nous permet de rendre le contrôle à l'application en pressant `Shift+F9`. Le programme se bloque une fois de plus sur une exception. À partir de maintenant, je décide de compter le nombre d'exceptions pendant le chargement du programme.

Toutes les exceptions ont le même code sauf la dernière qui sera légèrement différente. J'ai compté 27 exceptions avant le lancement de l'application. Ce nombre n'est pas une constante et est aléatoire. Afin de déterminer la dernière exception et de ne pas lancer l'application, il suffit de regarder le code qui suit l'exception.

Les 26 premières exceptions sont de type :

```
016D2293   3100            XOR DWORD PTR DS:[EAX],EAX
016D2295   EB 0x           JMP SHORT xxxxxxxx
```

Alors que la dernière exception est légèrement différente :

```
016D2D7A   3100            XOR DWORD PTR DS:[EAX],EAX
016D2D7C   648F05 00000000 POP DWORD PTR FS:[0]
```

Pour continuer à déboguer sans lancer l'exécution du programme, nous allons déposer un breakpoint en double cliquant sur la case correspondante aux opcodes (`648F050000--0000`) Nous pouvons maintenant presser `Shift+F9` une dernière fois. Le débogueur nous rend la main juste sur le `pop`. À partir de là, je décide de tracer un peu à la recherche d'un éventuel saut vers le début du programme déchiffré et j'arrive sur une longue boucle :

```
016E6A96   42              INC EDX
016E6A97   8BFA            MOV EDI,EDX
016E6A99   D1C7            ROL EDI,1
016E6A9B   81F7 A38FD7AC   XOR EDI,ACD78FA3
016E6AA1   3BFD            CMP EDI,EBP
016E6AA3   0F85 EDFFFFFF   JNZ 016E6A96
016E6AA9   03DA            ADD EBX,EDX
```

Afin de ne pas tracer toute la boucle à la main je dépose une fois de plus un point d'arrêt juste en dessous du `jnz` et presse sur `F9` pour continuer l'exécution du programme. Maintenant que nous avons passé les exceptions, ainsi que cette longue boucle, nous allons pouvoir utiliser l'option la plus intéressante de ce débogueur : les *tracing conditions*.

Comme précédemment vu, l'*OEP* se trouve dans la première section. Nous savons donc que la protection au moment de rendre la main au programme protégé exécutera du code dans la première section. Ollydbger nous permet de définir des bornes d'adresses à surveiller.

Nous allons tout simplement demander au débogueur de s'arrêter sur la première instruction executée dans la première section. Pour cela, il suffit d'aller dans le menu *Debug* et de choisir *Set condition* nous choisissons maintenant *EIP in range* et nous entrons comme valeur l'adresse du début de la première section `401000` et comme adresse de fin, l'adresse du début de la seconde section `580000` (voir figure 2).

Note.- Pour obtenir ces adresses il suffit d'ajouter l'*image base* aux *RVA* des sections que nous avons récupérées avec notre éditeur de fichiers PE.

Ensuite il ne reste plus qu'à tracer : menu *Debug*, *Trace Into*. Le débogueur trace pour nous la protection et loggue toutes les instructions exécutées. Nous verrons par

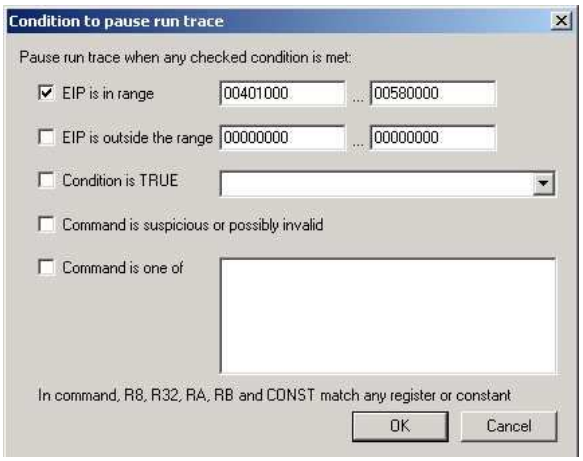


Fig. 2. Reprérage de la première section

la suite l'utilité de cette option. En bas à droite, nous pouvons voir le mot "Tracing" dans la barre d'état, il ne reste plus qu'à attendre.

Après quelques minutes, Ollydbger s'arrête sur la première instruction exécutée dans la première section, il s'agit de l'OEP. Notre programme est complètement déchiffré en mémoire.

```
0057F30F     E8            DB E8
0057F310     4C            DB 4C
0057F311     81            DB 81
0057F312     E8            DB E8
0057F313     FF            DB FF
```

Le débogueur n'a pas désassemblé le code, mais il s'agit bien de l'*OEP*. Notons l'adresse `57F30F`, elle nous servira par la suite. Il ne reste plus qu'à dumper le processus.

3.2 Dump du process décrypté

Pour cela, j'utilise le programme *LordPE*. Il suffit de sélectionner le processus de notre application et de faire un *full dump* (clic droit, *full dump*). Le programme déchiffré est écrit sur le disque. À ce stade de l'analyse, nombreuses sont les protections mises en echec. Asprotect étant plus résistante que la moyenne, nous allons maintenant nous occuper des imports.

3.3 Protection de l'*Import Table* et ses Faiblesses

Présentation Depuis quelques années, les protections d'exécutable détournent les appels aux API[5]. Au lieu d'appeler une API directement, un programme protégé commence par appeler une routine présente dans la protection et c'est celle-ci qui appelle l'API. Voici un petit schéma pour résumer le détournement d'une API :

[5] Application Programming Interface

– Programme normal :

$$\text{Programme} \xrightarrow{\textit{appelle}} \text{API}$$

– Programme protégé :

$$\text{Programme} \xrightarrow{\textit{appelle}} \text{routine Asprotect} \xrightarrow{\textit{appelle}} \text{API}$$

Quel est l'intérêt d'un détourment ? Les adresses présentes dans l'*import table* pointent vers la protection après le dump du programme. La protection n'étant plus active, les adresses n'existent plus et le programme affiche une erreur et s'arrête.

Nous allons passer en revue quelques méthodes utilisées par Asprotect pour rediriger les appels aux API.

Redirection simple

Dans un programme non protégé les API sont appelées comme ceci :

```
FF15D4B05300          CALL       [KERNEL32!GetVersionExA]
```

Ce `CALL` appelle une fonction à l'adresse `53B0D4`. Il suffit de retirer `FF15` dans l'opcode et d'inverser l'ordre des octets pour obtenir cette adresse.

Nous pourrions représenter l'appel de cette façon : `CALL DWORD PTR [53B0D4]`. Jetons un œil à cette adresse :

```
01AF:0053B0D4 0B 16 F9 BF 88 43 F7 BF-50 E1 F8 BF C6 20 F8 BF
01AF:0053B0E4 B1 EE F9 BF 08 2D F9 BF-B2 B9 F7 BF 8D B9 F7 BF
01AF:0053B0F4 BD C8 F7 BF 8E 5A F9 BF-32 60 FC BF DA C5 F8 BF
01AF:0053B104 FA AB F8 BF B1 42 F8 BF-AE 79 F7 BF D5 79 F7 BF
01AF:0053B114 F8 D4 F8 BF B1 6F F7 BF-6B 51 F8 BF B7 EE F9 BF
01AF:0053B124 EC 13 F8 BF 54 74 F7 BF-9F 7D F7 BF 3C C6 F9 BF
01AF:0053B134 9F FA F9 BF 22 0B F8 BF-2A 0A F8 BF 18 13 F9 BF
```

Nous sommes ici dans l'*IAT*[6]. C'est une table qui contient les adresses de toutes les API utilisées par le programme. Les adresses sont placées dans cette table lors de l'exécution du programme par Microsoft(r)Windows. L'adresse `BFF9160B` est donc l'adresse de l'API `GetVersionExA`. Les protections modifient l'*IAT* et remplacent les adresses des API par leurs propres routines. Voici un exemple très simple de détournement :

```
FF1512846000  CALL [00608412]
```

Regardons l'*IAT*.

```
01AF:00608412 75 20 85 21 41 52 85 21 11 74 85 21 73 98 85 21
```

Le programme appelle l'adresse : `21852075` (équivalent à : `CALL 21852075`)

```
21852075 PUSH BFF9162D
2185207A RET
```

Asprotect met sur la pile l'adresse de l'API à appeler, et ensuite utilise un `RET` pour s'y rendre. Ceci est la forme la plus simple du détournement d'API. Pour réparer ce détournement il suffirait de remplacer l'adresse dans l'*IAT* (`21852075`) par l'adresse de l'API : *BFF9162D*. Ce qui nous donnerait :

[6] *Import Address Table*

```
01AF:00608412 2D 16 F9 BF 41 52 85 21 11 74 85 21 73 98 85 21
```

Il suffirait d'appliquer la même méthode toutes les API. Il est évident qu'exécuter tout cela à la main prendrait réellement trop de temps. Il existe un outil pour reconstruire les tables d'imports qui permet de tracer les détournements et de remplacer automatiquement les adresses dans la l'*IAT* : *Imp Rec*.

Détournement avec pseudo-émulation

```
01AF:00442BAA  FF15D4504600      CALL    [004650D4]

01AF:004650D4 14 20 EE 00 28 20 EE 00-34 20 EE 00 40 20 EE 00
01AF:004650E4 4C 20 EE 00 58 20 EE 00-6C 20 EE 00 78 20 EE 00
01AF:004650F4 84 20 EE 00 98 20 EE 00-AC 20 EE 00 10 C9 EC 00
01AF:00465104 B8 20 EE 00 C4 20 EE 00-D4 20 EE 00 F0 20 EE 00
01AF:00465114 08 21 EE 00 14 21 EE 00-2C 21 EE 00 38 21 EE 00
```

La routine detournée appelle dans notre cas l'adresse `EE2014`. Juste pour information, voilà à quoi ressemble la routine :

```
01AF:00EE2014  55                PUSH      EBP
01AF:00EE2015  8BEC              MOV       EBP,ESP
01AF:00EE2017  83EC0C            SUB       ESP,0C
01AF:00EE201A  56                PUSH      ESI
01AF:00EE201B  57                PUSH      EDI
01AF:00EE201C  E9F2F50ABF        JMP       BFF91613 <= appel l'API.

:what bff91613
The value BFF91613 is (a) KERNEL32!GetVersionExA+0008
```

Cette routine appelle l'API `GetVersionExA`. Ce détournement est un peu plus complexe que le précédent, puisque la protection exécute les huits premiers octets (cinq instructions) de l'API dans sa propre routine et ensuite appelle l'adresse de l'API + 8. En exécutant les premières instructions d'une API dans sa propre routine, la protection arrivait à duper les premières versions des outils de reconstruction d'*IAT*. De nos jours, ces techniques sont reconnues comme obsolètes.

Reconstruction de l'Import Table

Nous allons utiliser *ImpRec* (*Import Reconstructor*) pour reconstruire notre *Import Table* (voir figure 3) Il suffit tout simplement de choisir le processus protégé et ensuite d'entrer quelques informations telles que le point d'entrée, l'adresse de l'*IAT* et sa taille. Nous avons trouvé l'*OEP* un peu plus tôt : `0057F30F`.

ImpRec attend comme valeur une *RVA*, il faut donc entrer : `17F30F` (*RVA* = *VA* - *Image base*). Nous pressons ensuite sur le bouton *IAT Autosearch*. Un message nous avertit et nous affiche l'adresse de l'*IAT* et sa taille mais celle-ci n'est pas correcte. ImpRec nous trouve :

`RVA IAT : 0019F350 Taille IAT : 000C`

Une *IAT* ne peut faire seulement douze octets, la taille est donc fausse. Substituons-le par `1000`. Il y a également de fortes probabilités que l'*IAT* soit fausse, nous rentrerons

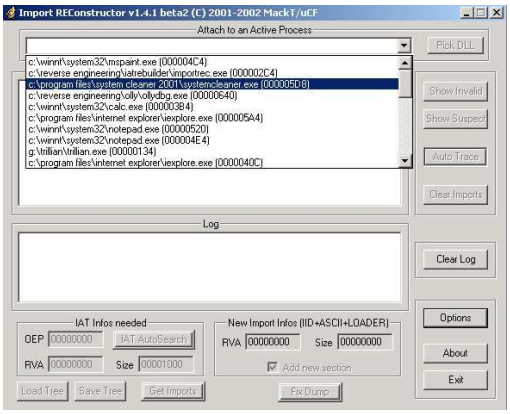


Fig. 3. Reconstruction de l'Import Table avec *Import Reconstructor*

donc `0019F000`. Nous pressons cette fois-ci sur *Get Import* pour obtenir les imports. Les adresses ne sont pas correctes ce qui veut dire que nous ne sommes pas encore dans l'*IAT*. En descendant un peu pour faire défiler les adresses nous obtenons la figure 4. J'ai souligné les adresses valides de l'*IAT*. Elles commencent toutes au même endroit. `16xxxxx`. Elles se trouvent juste au dessus des adresses non valides. À partir de ce moment-là, nous savons que le début de l'*IAT* est à l'adresse : `19F258`. Pour connaître la taille de l'*IAT*, il suffit de descendre plus bas et de voir la limite entre les bonnes et les mauvaises adresses : À partir de maintenant, nous savons que l'*IAT* se termine en `19FC60`. Pour obtenir la taille de celle-ci, il suffit simplement de soustraire l'adresse de début à l'adresse de fin : `19FC60 - 19F258 = A08`. Nous rentrons donc les informations correctes :

```
RVA IAT : 19F258 Taille IAT : A08
```

Nous allons pouvoir commencer la reconstruction de l'*import table*, mais juste avant de commencer, une pression sur *Clear imports* pour effacer les imports, suivie d'une pression sur le bouton *Get Imports* permet de lister l'*IAT* dans sa totalité. La première chose mise en évidence, c'est qu'il n'y a aucune DLL de listée. Asprotect insère des adresses invalides à la place des doubles mots nuls présents normalement pour indiquer le début d'une autre DLL. Nous allons commencer par retirer tous les doubles mots invalides en les sélectionnant grâce à la touche CTRL et la souris. Une fois toutes les adresses invalides sélectionnées, il suffit de faire un clic droit et d'utiliser l'option *cut thunks* comme ceci (voir figure 5). Aprés l'opération, nous avons bien maintenant notre *IAT* découpée par DLL et certaines d'entre elles contiennent toujours des API non réparées (voir figure 6). Il suffit simplement de faire un clic droit sur chaque adresse et de choisir l'option *TRACE LEVEL 1 (DISASM)*. Les noms des API apparaissent maintenant. Nous remarquons cependant des adresses ayant une adresse à peine différente de la majorité. Les adresses « normales » commencent par `16Exxxx` alors que certaines adresses sont de la forme `16Dxxxx`.

Si nous essayons de les tracer, notre outil se bloque et il faut tout recommencer depuis le début. Nous allons pour l'instant seulement nous préoccuper des adresses en

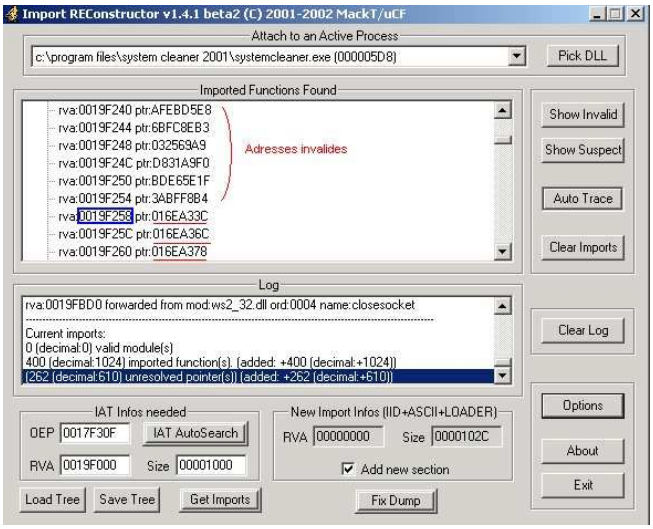


Fig. 4. Obtention des imports

`16Exxxx` avec le traceur level 1. Il suffit de sélectionner les adresses à l'aide de CTRL et de la souris comme nous l'avons fait tout à l'heure et de choisir le traceur.

Il reste à présent sept pointeurs non réparés (voir figure 7).

Emulation d'API

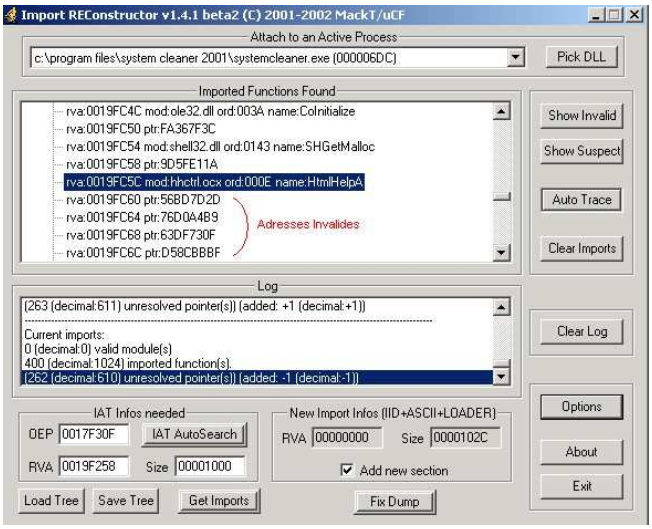
Asprotect émule certaines API et ne les appelle pas vraiment. ImpRec n'a donc pas su de quelle API il s'agissait.

J'ai du explorer chaque API à la main pour pouvoir écrire un plugin spécial Asprotect et l'utiliser avec l'outil de reconstruction d'*IAT*. Je vais maintenant montrer quelques exemples d'émulation et expliquer comment j'ai construit mon plugin.

Prenons l'exemple suivant :

```
001B:016D1408  6A00              PUSH    00
001B:016D140A  E8513DFFFF        CALL    KERNEL32!GetModuleHandleA
001B:016D140F  FF35E86C6D01      PUSH    DWORD PTR [016D6CE8]
001B:016D1415  58                POP     EAX
001B:016D1416  8B05F86C6D01      MOV     EAX,[016D6CF8]
001B:016D141C  C3                RET
```

La protection appelle en premier `GetModuleHandleA`, mais nous voyons que celle-ci pousse une valeur sur la pile et la récupere dans `EAX`. Nous voyons juste en dessous qu'une valeur précédemment sauvegardée est placée dans `EAX`. Le `RET` juste après nous indique que l'appel est terminé. le traceur level 1 va retourner l'API `GetModuleHandleA` alors qu'en fait il s'agit d'une autre API. Elle n'est pas appelée contrairement aux autres détournements. Pour savoir de quelle API il s'agit, il faudrait trouver le code



qui va placer sa valeur de retour dans une variable afin d'être utilisée plus tard lors de l'émulation.

Pour trouver de quelle API il s'agit, il suffit de déposer un point d'arrêt sur `GetVersion` dans notre débogueur et de relancer l'application protégée. Une fois que le débogueur s'arrête, il suffit de presser *F12* jusqu'à être dans notre processus.

À partir de là, nous pouvons voir comment Asprotect sauvegarde les valeurs retournées par les API, pour les utiliser plus tard lors de l'émulation.

```
001B:016D11ED  68F5116D01          PUSH      016D11F5
001B:016D11F2  EB6A                JMP       016D125E

001B:016D125E  5A                  POP       EDX
001B:016D125F  5B                  POP       EBX
001B:016D1260  6867126D01          PUSH      016D1267
001B:016D1265  C3                  RET

001B:016D1267  8943F6              MOV       [EBX-0A],EAX
      ; EBX-A = adresse de sauvegarde.
001B:016D126A  FFE2                JMP       EDX
001B:016D126C  61                  POPAD
001B:016D126D  C3                  RET
```

Continuons de tracer et nous arrivons ici :

```
001B:016D11F5  685E6C6D01          PUSH      016D6C5E
001B:016D11FA  6809126D01          PUSH      016D1209
001B:016D11FF  8B0582516C01        MOV       EAX,[016C5182]
001B:016D1205  FF20                JMP       [EAX]
```

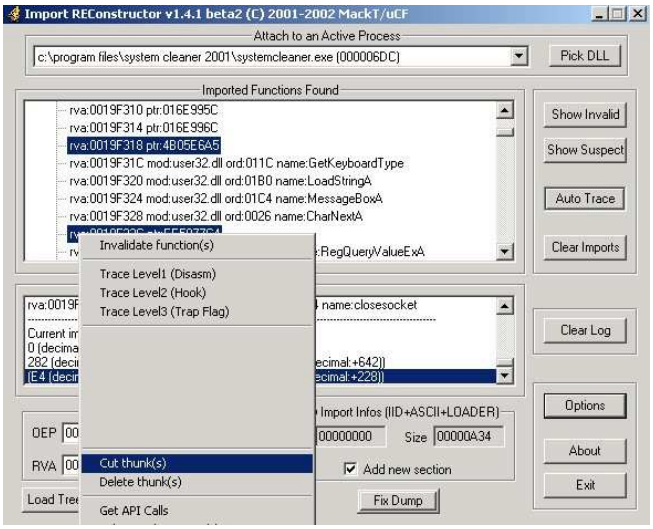


Fig. 5. Utilisation option *cut thunks*

`EAX` contient l'adresse de l'API à émuler. Il s'agit d'une boucle qui exécute toutes les API à émuler en prenant soin de sauvegarder leur résultat.

Exemple : Asprotect appelle `GetCommandLineA` et retourne dans `EAX` sa valeur de retour. Cet appel n'est pas détourné, c'est la protection qui appelle l'API normalement.

```
KERNEL32!GetCommandLineA
001B:77E7F07F  A194F6EC77          MOV       EAX,[77ECF694]
001B:77E7F084  C3                  RET
```

Ensuite le code que nous avons déjà vu est exécuté :

```
001B:016D1267  8943F6              MOV       [EBX-0A],EAX
001B:016D126A  FFE2                JMP       EDX
```

Nous pouvons voir où la valeur de retour est sauvegardée en regardant où pointe `EBX-0x0A`. Dans le dump ci-dessous, la valeur n'a pas encore été sauvegardée :

```
0023:016D6CF8 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ........
```

La valeur de retour est sauvegardée en `16D6CF8`, et l'API correspondante à cette adresse est `GetCommandLineA`.

Lors de l'émulation, la protection appelle le bout de code ci-dessous pour récuperer la valeur de retour de `GetCommandLineA` comme si cette API venait d'être utilisée.

Il n'y a aucune différence pour le systeme d'exploitation, mais les outils de reconstruction d'*IAT* sont impuissants face à cette émulation.

```
001B:016D1416  8B05F86C6D01        MOV       EAX,[016D6CF8]
001B:016D141C  C3                  RET
```

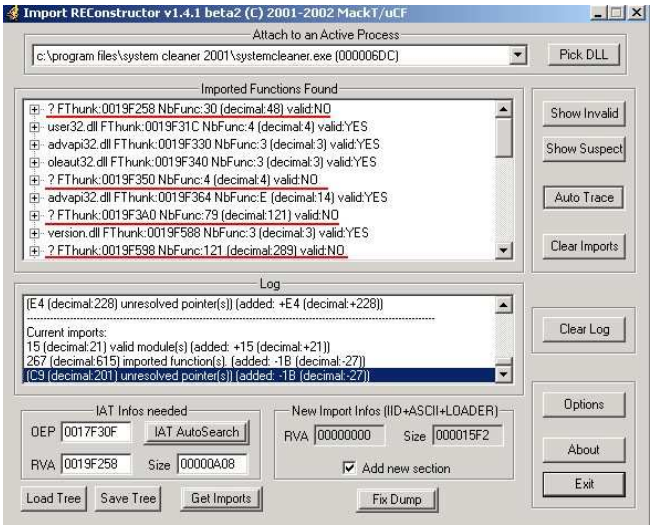


Fig. 6. API non réparées

Il est donc possible de réparer les dernières API de cette façon, mais ceci prends du temps. J'ai donc écrit un plugin à base de signatures pour reconnaître l'API émulée et la réparer.

Dans notre exemple avec `GetCommandLineA` le code complet de l'émulation etait celui-ci :

```
001B:016D1408  6A00              PUSH      00
001B:016D140A  E8513DFFFF        CALL      KERNEL32!GetModuleHandleA
001B:016D140F  FF35E86C6D01      PUSH      DWORD PTR [016D6CE8]
001B:016D1415  58                POP       EAX
001B:016D1416  8B05F86C6D01      MOV       EAX,[016D6CF8]
001B:016D141C  C3                RET
```

Mon plugin recherche à l'adresse d'une API émulée la signature suivante : `6A00E8`.

`GetCommandLineA` est la seule API émulée de cette facon, il n'y a aucun risque de confusion avec une autre émulation.

Le plugin recherche ces 3 octets et s'il les trouve, comme `GetCommandLineA`, déclare l'adresse.

Il suffit de faire correspondre toutes les émulations à une signature donnée pour écrire le plugin qui permettra de réparer par la suite n'importe quel programme protégé par Asprotect, tout cela en quelques clics de souris.

Notre *Import Table* est maintenant complètement réparée. Nous pouvons l'injecter dans notre dump. Il suffit de presser *Fix Dump*, de choisir le fichier et l'*IAT* sera insérée correctement dans le programme. À cette étape, les premières versions d'Asprotect étaient complètement unpackées. Aucune autre protection n'était présente, mais depuis peu, une nouvelle protection existe, *les stolen bytes*.

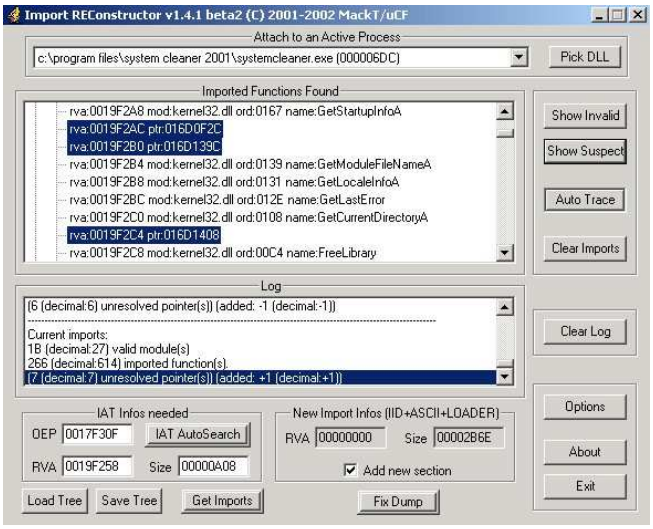


Fig. 7. Pointeurs non réparés

3.4 Protection *Stolen Bytes*

Présentation Si nous exécutons notre dump, il ne fonctionne pas et nous affiche des erreurs. En règle générale, un compilateur utilisera toujours un *stack frame* au point d'entrée. Dans notre fichier dumpé, la première instruction est un `CALL`. La protection *Stolen Bytes* consiste à exécuter les instructions au point d'entrée et de les effacer ensuite, afin de rendre tout dump non-opérationnel.

Pour une plus grande sécurité, ces instructions sont d'abord recopiées dans le code de la protection, puis mises à zéro lors de la protection du programme. À chaque exécution, les instructions protégées seront effacées avant l'appel du reste du programme.

Récupération des *Stolen Bytes* Tout à l'heure, nous avons utilisé le traceur pour nous rendre au point d'entrée. Le traceur a sauvegardé toutes les instructions que nous avons exécutées. *Menu View*, *Run Trace* pour afficher le listing :

```
016E7955   60              PUSHAD
016E7956   9C              PUSHFD
016E7957   FC              CLD
016E7958   BF 6B796E01      MOV EDI,16E796B
    ; EDI pointe vers l'adresse à effacer.
016E795D   B9 53010000      MOV ECX,153                ; nombre de bytes
016E7962   F3:AA            REP STOS BYTE PTR ES:[EDI]  ; overwrite..
016E7964   9D              POPFD
016E7965   61              POPAD
016E7966   E9 A479E9FE      JMP SystemCl.0057F30F    ; Saut à l'OEP
```

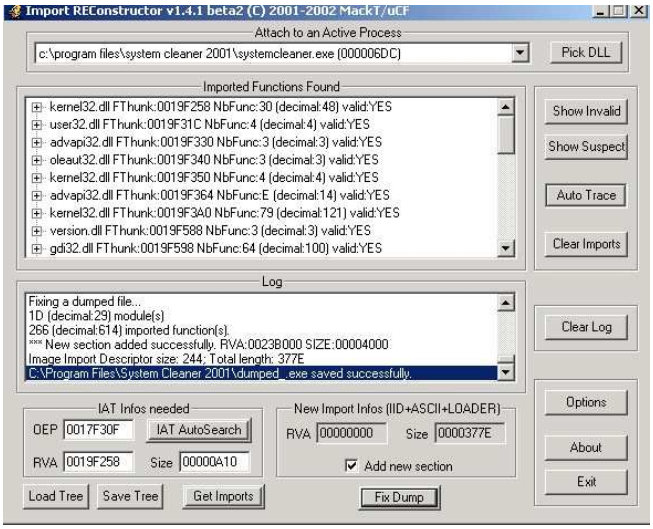


Fig. 8. Insertion de l'*IAT*

Asprotect, avant d'appeler le point d'entrée, efface `0x153 octets` du programme. Nos logs ne contiennent pas les octets avant effacement, nous allons donc devoir trouver un moyen de stopper avant leur effacement. Pour cela nous allons utiliser une autre condition de traçage. Quand *Ollydbger* s'arrête au point d'entrée, le registre `EAX` contient la valeur `57EBCC`. Gardons cette valeur de côté et relançons notre application sous *Ollydbger* (CTRL+F2). Procédons comme pour la recherche du point d'entrée, mais au lieu de fournir une fourchette d'adresses, donnons une condition basée sur le contenu d'EAX, ce qui nous permettra de stopper avant l'effacement de ces octets : *Menu Debug*, *set condition* et nous remplissons `EAX == 57EBCC` comme indiqué sur la figure 9. Ensuite dans le *Menu Debug*, il suffit de sélectionner *Trace into*. *Ollydbger* trace de la même manière, mais cette fois-ci, il s'arrête avant le point d'entrée, juste avant qu'Asprotect n'efface les octets nécessaires au programme :

```
016E5689   68 15566E01      PUSH 16E5615
016E568E   C3               RETN
```

Remontons un petit peu dans le désassemblage (voir figure 10).

```
016E567E   55               PUSH EBP
016E567F   8BEC             MOV EBP,ESP
016E5681   83C4 F0          ADD ESP,-10
016E5684   B8 CCEB5700      MOV EAX,57EBCC
016E5689   68 15566E01      PUSH 16E5615
016E568E   C3               RETN
```

Voici les octets que la protection exécute puis efface. Les quatre premières instructions sont importantes. Le `PUSH` et le `RET` servent à se déplacer dans la protection d'une manière non habituelle. Il ne nous reste plus qu'à insérer ces octets dans le programme

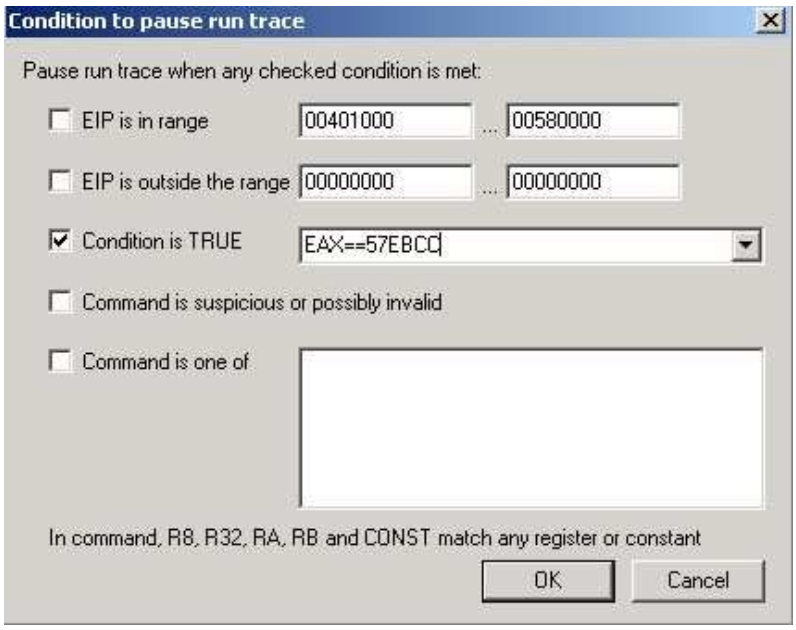


Fig. 9. Recherche sur la condition `EAX == 57EBCC`

et changer le point d'entrée pour avoir un exécutable complètement reconstruit et sans protection.

Insertion des *Stolen Bytes* Pour ajouter les octets manquants, nous allons utiliser un éditeur hexadécimal. N'importe quel éditeur fera l'affaire, j'utilise l'éditeur intégré à *Lord PE*. Il suffit de se rendre au segment du point d'entrée c'est à dire à *17F30F*.

Nous retrouvons bien notre `E8`, qui représente le call que nous avions au point d'entrée. Les octets avant notre point d'entrée ont été effacés (octets nuls) (voir figure 11). Nous insérons les octets manquants : `558BEC83C4F0B8CCEB5700` (voir figure 12). Nous sauvegardons, il ne nous reste plus qu'à changer le point d'entrée du programme et nous en avons terminé avec Asprotect.

Binaire final : mise à jour du point d'entrée Avec *Lord PE*, nous allons changer le point d'entrée de notre programme qui est pour l'instant fixé à `17F30F`. Nous venons de changer onze octets qu'il va falloir exécuter. Le point d'entrée de notre programme va être `17F30F-0x0B` soit : `17F304`. Nous entrons le nouveau point d'entrée, nous sauvegardons et lançons l'application pour vérifier. Celle-ci se lance sans problème.

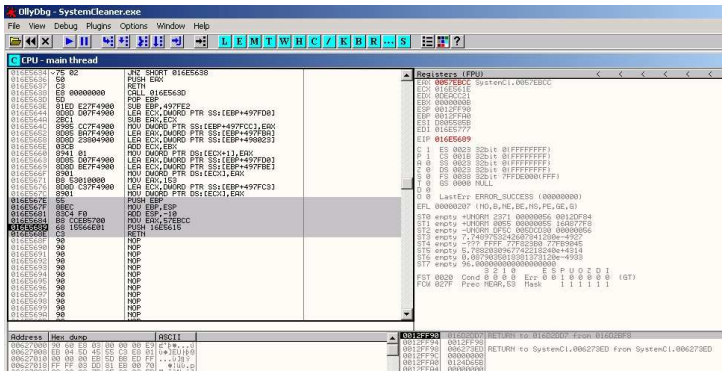


Fig. 10. Remontée dans le désassemblage

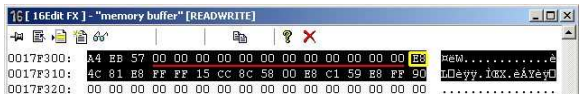


Fig. 11. Etat des octets avant le point d'entrée

4 Conclusion

Il est maintenant possible de désassembler notre binaire, d'analyser son fonctionnement ou même de le modifier. Dans le cas d'applications sensibles, dont le fonctionnement doit rester secret, un tel niveau d'insécurité est plus que problématique. En régle générale, il ne faut pas compter sur les protections d'exécutables pour cacher le fonctionnement d'une application. Un attaquant finira toujours par retirer la protection, ou en retirer suffisamment pour comprendre le fonctionnement du binaire protégé.

Les protections par dongle tel que *HASP* ou *SENTINEL* offrent aussi des enveloppes du même type. Ces protections d'exécutables s'avèrent de moins bonne qualité lorsque l'attaquant possède le dongle. Dans le cas des dongles *HASP*, il suffit de trouver l'*OEP* du programme, de dumper le processus et de corriger le point d'entrée pour obtenir un exécutable fonctionnel sans dongle, et désassemblable. De plus, certains outils permettent de retirer ces protections automatiquement, voire même de les émuler.

J'espère que ce tutoriel aura fait comprendre qu'une protection d'exécutable, aussi performante soit-elle, n'arrête jamais une personne déterminée, et que la meilleure

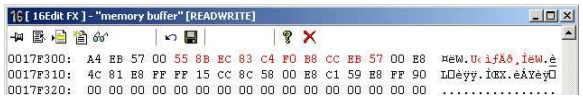


Fig. 12. Insertion des octets manquants

des protections reste encore la distribution trés limitée (et surveillée) des programme concernés.

Références

1. Documentation sur le format PE, `http://spiff.tripnet.se/~iczelion/files/pe1.zip`

La rétroconception : application à l'analyse logicielle

Serge Lefranc

Centre d'Électronique de l'Armement
Bruz CEDEX F-35174, France
serge.lefranc@dga.defense.gouv.fr

Résumé Cet article a pour but de présenter la rétroconception et son application à l'analyse logicielle. Pour cela, nous nous attacherons à présenter certains aspects juridiques relatifs à la rétroconception. Nous expliciterons ensuite ses buts afin de nous permettre d'élaborer une méthodologie associée à l'utilisation de certains outils. Après avoir donné quelques exemples de mécanismes de protection et les mesures de contournement correspondantes, nous essayerons d'illustrer cet article par un exemple.

1 Introduction

La rétroconception peut se définir comme l'action d'extraire une connaissance ou un savoir d'une réalisation.

Cette activité est pratiquée depuis très longtemps dans un grand nombre de domaines industriels comme l'automobile ou l'électronique, afin, notamment, de permettre l'accession à un savoir ou une technologie, à moindre frais.

L'informatique n'a pas été épargnée par ce phénomène. Tout possesseur d'un programme possède le code de l'application traduit dans le seul langage que le processeur comprenne, le langage machine. Il suffit ensuite d'avoir les compétences nécessaires pour obtenir un accès de bas niveau aux fonctionnalités du programme.

Historiquement, c'est dans les pays de l'ex Bloc de l'Est et dans certains pays d'Asie que les compétences en rétroconception sont les plus vivaces. N'ayant pas accès de façon simple aux logiciels des pays de l'ex Bloc de l'Ouest, principalement pour cause de restriction, ou d'isolement économique, une forte communauté de rétroconcepteurs s'est développée, d'autant qu'elle était encouragée par l'absence de sanctions et de législation au niveau étatique. C'est donc, tout naturellement, le monde du logiciel payant qui a été le plus victime de cette forme de piratage. Par conséquent, il est normal de trouver une importante communauté de rétroconcepteurs dans le monde Windows.

Ces dernières années ont cependant été marquées par une évolution des tendances. Des compétences en rétroconception ont été nécessaires, et acquises, de la part de la communauté Open Source afin d'identifier les mécanismes de fonctionnement des différents vers qui ont touché le monde du logiciel libre ou d'extraire de l'information des traces binaires retrouvées sur une machine compromise.

Notre exposé va donc tenter, après avoir abordé l'aspect juridique de la rétroconception, de montrer que ses buts sont nombreux et qu'ils dépendent très fortement de l'état d'esprit des personnes qui la pratiquent. Nous verrons ensuite qu'il est difficile

de définir une méthodologie de la rétroconception car le spectre de ses objectifs est trop large. Dans le cas le plus simple, typiquement le cracking, nous savons ce que nous cherchons, et dans le cas le plus compliqué, comme la recherche de vulnérabilités, nous n'en n'avons aucune idée. Ceci nous conduira à présenter quelques schémas de protection et les façons de les contourner pour enfin d'illustrer notre présentation par une exemple d'analyse logicielle.

2 Aspects juridiques

Cette partie est basée sur le cours de droit informatique, *l'univers numérique et le droit*, écrit par Maître Gérard HAAS donné à l'École Nationale Supérieure de Techniques Avancées en 2001.

Il convient de ne pas confondre désassemblage et décompilation. Cette dernière transforme le code machine en langage de haut niveau alors que le désassemblage le transforme en langage assembleur.

2.1 Droit d'analyse

L'utilisateur régulier d'un logiciel se voit reconnaître expressément le droit, sans avoir à en demander l'autorisation à l'auteur, d'observer, étudier ou tester le fonctionnement de ce logiciel afin de déterminer les idées et principes qui sont à la base de n'importe quel élément du programme, lorsqu'il effectue des opérations de chargement, d'affichage, de passage, de transmission ou de stockage du programme (Code de la Propriété Intellectuelle art. L.122-6-1-111).

Le droit d'analyse ne peut être exercé que dans le cadre de l'utilisation normale du logiciel. À la différence du droit de décompilation, le droit d'analyse est réservé à l'utilisateur du logiciel, qui ne peut autoriser un tiers à l'exercer à sa place.

2.2 Décompilation

L'utilisateur régulier d'un logiciel peut, sans autorisation de l'auteur, procéder à la reproduction et à la traduction de la forme de son code, c'est à dire sa *décompilation*, lorsque ces opérations sont indispensables pour obtenir les informations nécessaires à l'interopérabilité de ce logiciel avec d'autres logiciels (Code de la Propriété Intellectuelle art L.122-6-1-IV).

Compte tenu des termes de la loi, il semble que, outre la décompilation stricto sensu, le *désassemblage* du logiciel soit permis (Hubert Bitan, 01 Informatique, 17/06/94, p.36). Par ailleurs, la notion de décompilation était inconnue de notre droit, elle a été introduite par la directive européenne du 14 mai 1991.

L'objectif clairement affirmé du législateur est de permettre l'interopérabilité des logiciels, définie par la directive comme la *capacité des programmes d'ordinateurs d'échanger des données et d'utiliser des données qui ont été échangées.* Le droit de décompiler demeure cependant enfermé dans des conditions très strictes :

- il doit s'agir d'un logiciel créé de façon indépendante et donc qui n'est pas déjà conçu pour être compatible avec d'autres logiciels ;
- seuls sont habilités à procéder à des actes de décompilation les licenciés ou utilisateurs réguliers du logiciel ou les tiers agissant pour leur compte ;

- les informations nécessaires à l'interopérabilité ne sont pas déjà facilement et rapidement accessibles ;
- les actes de décompilation sont limités aux parties du logiciel d'origine nécessaires à l'interopérabilité.

Il convient d'être réservé sur la portée pratique de cette dernière limitation. En effet, il est impossible de savoir, la plupart du temps, où se trouvent les interfaces dans un programme, et la preuve d'un abus s'annonce pour le moins délicate, sauf si le concepteur a donné des indications suffisantes... De plus, les dispositions permettant la décompilation n'autorisent pas pour autant que les informations obtenues :

- soient utilisées à des fins autres que l'interopérabilité des logiciels avec d'autres logiciels ;
- soient communiquées à des tiers sauf si cela est nécessaire pour permettre l'interopérabilité avec le logiciel crée de façon indépendante ;
- soient utilisées pour la mise au point, la production ou la commercialisation d'un logiciel dont l'expression est fondamentalement similaire à celle du logiciel d'origine ou pour tout autre acte portant atteinte au droit d'auteur.

3 Les buts de la rétroconception

Pourquoi avoir besoin de désassembler un programme ? Nous avons vu que légalement, et sous certaines conditions, cette opération est autorisée pour assurer une interoperabilité des applications.

Cependant, ce n'est qu'une facette des raisons motivant le désassemblage. La rétroconception permet également de vérifier la présence de portes cachées ou de vulnérabilités dans un logiciel. Elle peut permettre de passer outre un système de protection, voire même permettre l'accès à certaines techniques propriétaires non documentées.

Toutes ces raisons de pratiquer le désassemblage sont très dépendantes des différents acteurs qui vont la réaliser. En effet, leurs motivations ne vont pas être les mêmes. Et cela va avoir une incidence sur les compétences à mettre en œuvre pour la réaliser. En particulier plus le but de la rétroconception est précis, plus les compétences nécessaires sont simples à acquérir : il est plus simple de chercher quelque chose de connu que d'analyser un programme sans savoir ce que l'on cherche.

3.1 Une entreprise

Pour une entreprise, nous identifions trois raisons principales d'avoir recours à la rétronception :

- assurer l'intéropérabilité entre une application et son parc applicatif ;
- vérifier que le produit ne fait que ce qu'il est censé faire ;
- accéder aux technologies mises au point par des sociétés concurrentes.

Les moyens mis en œuvre pour réaliser ces objectifs sont nombreux car le spectre de difficultés rencontrées est large. Pour assurer l'interopérabilité, les problèmes sont relativement aisés à identifier, même si ils ne sont pas forcément simples à résoudre, alors que pour les deux autres objectifs, il est difficile d'identifier ce qu'il faut chercher. Par exemple, la recherche de vulnérabilités est considérée comme le pire cas dans lequel le rétroconcepteur peut se trouver : il n'a aucune idée de ce qu'il cherche, ni de ce qu'il va trouver.

3.2 Un particulier

La problématique est plus simple pour un particulier. Nous identifions deux raisons qui peuvent le mener à analyser un logiciel :
- enlever les mécanismes de protection lui interdisant l'utilisation de ce logiciel sans s'être acquitté des droits de licence ;
- le modifier afin d'incorporer des fonctionnalités particulières.

Le premier cas de figure fait appel à certaines compétences spécifiques aux méthodes de protections utilisées par les éditeurs de logiciels (nous aborderons, par la suite, quelques techniques utilisées par ces concepteurs pour protéger leurs œuvres), tandis que le deuxième cas va nécessiter des compétences qui dependront des modifications à apporter, cela peut aller du très simple au très compliqué.

3.3 Un groupe de pirates

Pour un groupe de pirates nous identifions également deux objectifs :
- supprimer des différents mécanismes de protection présents sur le logiciel afin de permettre de le distribuer massivement de façon gratuite ;
- rechercher des vulnérabilités afin de les exploiter et de les distribuer.

Les compétences nécéssaires pour réaliser le premier objectif sont très spécifiques. En effet, les schémas de protection évoluant finalement assez peu, les méthodes à employer sont relativement simples. Par contre, le deuxième objectif nécessite des connaissances très poussées en programmation de bas niveau, en architecture des systèmes d'exploitation, voire même en architecture rèseau.

3.4 Une organisation criminelle

L'organisation criminelle poursuit les mêmes buts que le groupe de pirates, mais à l'inverse, elle le fait dans un but lucratif.

3.5 Un organisme étatique

L'organisme étatique a deux principales raisons de faire de la rétroconception :
- vérifier que le logiciel ne possède pas de fonctions cachées ou de vulnérabilités pouvant dégrader la sécurité de l'ensemble du système ;
- rechercher des vulnérabilités afin de les exploiter dans un contexte opérationnel.

On considère que ces objectifs sont ceux qui nécessitent le plus de compétences. En terme de difficultés techniques, ce cas est analogue à celui de l'entrerprise.

4 La méthodologie

Il n'existe pas de méthodologie de la rétroconception ! En effet, si cela était le cas, il existerait des logiciels « clé en main » permettant d'analyser un exécutable.

Nous avons vu que les objectifs à atteindre peuvent être extrêmement variés et nécessiter des compétences très différentes. Cela va du plus simple : enlever une boite de dialogue, au plus compliqué : identifier la présence d'une vulnérabilité et la manière de

l'exploiter. Il n'est donc pas possible de décrire une méthodologie de la rétroconception que nous pourrions appliquer à tous les cas de figure.

Cependant, il est quand même possible de dégager un semblant de méthode. Cette esquisse de méthode résulte en fait de l'utilisation de logiciels bien particulier qui vont nous permettre d'accéder à l'information désirée :

- un désassembleur ;
- un débuggeur ;
- des logiciels de surveillance système.

Il est possible de classer ces outils en deux grandes familles : les outils de rétroconception et les outils de surveillance système.

4.1 Les outils de rétroconception

Ils vont nous permettre d'acquérir une vision du fonctionnement interne du programme. Il devient alors possible d'étudier les processus internes du logiciel, de comprendre ses faiblesses et, éventuellement, de les exploiter.

Le désassembleur Il permet d'étudier de façon statique ce que réalise le programme en traduisant l'exécutable en suite d'instructions machine. C'est lui qui va nous permettre de générer le code assembleur correspondant au programme.

Les désassembleurs IDA pro de la société Datarescue [1] ou DASM32 sont couramment utilisés sous Windows.

Le débuggeur Il va nous permettre d'étudier le comportement du programme pendant son exécution. Il va être capable d'en tracer l'exécution pas à pas, de connaître la valeur des registres, de placer des breakpoints (points d'arrêts : lors de la présence d'un breakpoint, le débuggeur fait une pause dans l'exécution de l'application) à certains endroits ou en fonction de certaines conditions.

Il existe deux types de débuggeurs : ceux fonctionnant au niveau applicatif et ceux fonctionnant au niveau système. Pour simplifier, nous pouvons considérer que les débuggeurs de niveau applicatif résident entre l'OS et le programme (ring 3) et que ceux de niveau système résident entre le processeur et l'OS (ces derniers sont plus puissants car ils peuvent débugger au niveau noyau, ring 0). Typiquement, le débuggeur présent dans la suite de développement Visual C++ de Microsoft [2] est de niveau applicatif, et le débuggeur SoftICE proposé par Numega est de niveau système [3].

4.2 Les outils de surveillance

Ce sont des outils qui vont nous permettre de surveiller l'activité du système de fichiers en temps réel, de surveiller l'activité de la base des registres ou l'activité réseau. Cela va permettre d'avoir une information externe au programme et d'identifier les interactions qu'il peut avoir avec son environnement de fonctionnement. Le programme *monitor* de la société Sysinternals [4] permettent de réaliser ces tâches.

5 Quelques protections

Nous venons de voir, qu'hormis l'utilisation d'outils bien spécifiques, il est difficile d'isoler une méthode précise pour faire de la rétroconception d'application. Nous allons détailler les schémas de protection les plus fréquents.

5.1 Les tokens logiciels

C'est l'une des méthodes les plus utilisées de protection logicielle. Ils peuvent prendre différentes formes.

Clé d'enregistrement. Une clé d'enregistrement est présente dans le programme. Lorsque l'utilisateur rentre son numéro de clé, le programme vérifie qu'elle est identique à celle qui est stockée dans le programme.

Serial multiple. Le numéro de série du logiciel est découpé en plusieurs parties ([xx][yy][zz]). Le programme possède un algorithme de vérification du numéro de série qui va vérifier les différentes parties de ce numéro. Cette technique est une amélioration de la précédente car elle permet l'utilisation de différents numéros de série sans avoir à les programmer en dur dans le programme.

Serial/Nom. Le token correspond ici à un couple numéro de série/nom. La vérification utilise le même principe que pour les serials multiples.

Fichier de clé. Le token correspond à un fichier contenant la licence d'utilisation. Elle est stockée dans un fichier sur le disque ou dans la base des registres.

5.2 Les tokens matériels

Étant donné le caractère volatile de l'information numérique et la facilité avec laquelle elle peut être reproduite, il a été envisagé que des solutions à base de tokens matériels pourraient être plus robustes au piratage. Il est en effet plus difficile de dupliquer une clé physique. Elles peuvent prendre plusieurs formes :

Disquette clé. Ce sont des disquettes spéciales dans lesquelles certains secteurs ont été physiquement altérés. Le programme va vérifier la présence des secteurs défectueux.

Dongle. Ce sont des petits périphériques matériels qui s'attachent aux ports d'entrées-/sorties de la machine. Les routines de vérification vont demander certaines valeurs aux périphériques connectés à ces ports. Si le token matériel est là, il va détecter le signal électrique et générer une réponse appropriée.

Carte à puce. L'information et/ou les routines de vérification se trouvent dans le processeur de la carte à puce.

5.3 Les NAG screen

Ce sont typiquement des fenêtres *pop-up* qui annoncent à l'utilisateur qu'il peut utiliser le programme mais qu'il n'en possède pas la licence. Elles s'ouvrent à chaque exécution du programme.

5.4 Les limites

Ce sont des limites dans l'utilisation du logiciel. Elles peuvent porter sur le nombre de fois où le programme peut être utilisé, la durée pendant laquelle il peut fonctionner, etc. Le programme ne pourra plus se lancer lorsque la limite sera atteinte.

5.5 Le chiffrement de code

C'est l'une des façons les plus simples de protéger l'exécutable d'un programme. Ce type de techniques est également utilisé par certains virus :

```
mov EAX, count
mov EBX, address
mov ECX, offset
_LOOP:
xor [ECX], EBX
DEC EAX
ja _LOOP
; ensuite on trouve le code et les données
; chiffrées du programme
```

Ainsi, lorsque le programme va être décompilé, l'utilisateur n'aura pas immédiatement accès aux instructions assembleur, il faudra qu'il identifie la ou les routines de chiffrement et qu'il déchiffre l'exécutable. Dans notre exemple, cette routine est un simple XOR, mais c'est souvent plus compliqué.

5.6 Le packing d'exécutable

Cette méthode est utilisée pour compresser le programme tout en lui permettant d'être exécutable en mémoire. Il existe un certain nombre de logiciels permettant de réaliser cette tâche (UPX, Neolite, PKLITE32, ...). Le fonctionnement de ce procédé est assez analogue au chiffrement de code.

5.7 La transformation de code

Le but de cette méthode est de rendre l'exécutable désassemblé incompréhensible et ainsi de ralentir le travail de l'utilisateur en l'empêchant de récupérer facilement de l'information sur le programme.

Il existe différentes façons de rendre cela possible :
- transformation lexicale ;
- transformation de contrôle : on ajoute des instructions opaques.

Par exemple, la suite d'instructions suivante :

```
instruction_a
instruction_b
```

est remplacée par :

```
instruction_a
if (p == true)
 instruction_b
else
 instruction_b
```

Il est possible de rendre la lecture d'un listing assembleur très difficile en utilisant ce type de méthodes. Par contre, l'efficacité du programme en est diminuée.

5.8 Les fonctions anti-débuggeur

Le but du jeu est de rendre l'utilisation d'un débuggeur beaucoup plus compliquée. Pour réaliser cela, il existe deux méthodes principales :

- actions préventives : ce sont des actions effectuées par le programme pour empêcher l'utilisateur de le tracer durant son exécution (masquage des interruptions, modification de l'IVT, ...) ;
- code auto-modifiant (algorithme de chiffrement/déchiffrement, ...).

Pour plus de précision on se réfera à l'article écrit par Inbar RAZ [5].

6 Les mesures de contournements associées

6.1 Le scénario le plus simple

Les schémas de protection que nous venons de présenter sont très simples à mettre en échec s'ils n'incorporent pas des mécanismes de chiffrement/obscurcissement car ils font souvent appel au modèle de protection suivant :

```
result = security_check(condition1, condition2)
if (result == TRUE)
then
 <autorise et execute le programme>
else
 <essaye encore>
```

La `condition1` peut être le numéro de série entré par l'utilisateur et la `condition2` le numéro de série qui est valide. La fonction `security_check` peut aller de la simple comparaison de bit à une requête I/O. Le pseudo code que nous venons d'écrire peut se traduire, en langage assembleur, de la façon suivante :

```
push condition2
push condition1
call security_check
test EAX, EAX
jnz address1 ; adresse du debut du programme
<essaye encore>
```

Le résultat de l'appel à la fonction est stocké dans le registre `EAX`. L'opération `TEST` réalise un *ET* logique. Si le résultat est 0 (faux), l'opération de *ET* logique va positionner le flag `Z` à 1 dans le registre de flags et le saut ne va pas avoir lieu. Si le contenu du registre `EAX` est différent de 0, alors le saut à l'adresse `address1` va avoir lieu.

Ce type de protection est facilement outrepassable en utilisant un débuggeur ou un désassembleur.

6.2 Utilisation d'un débuggeur

En positionnant certains points d'arrêt (par exemple, sur une fonction de comparaison de chaînes de caractères), il va être possible à l'utilisateur :

- de récupérer le serial (stocké en dur dans le programme : `condition2`) ;
- de modifier l'instruction JNZ en JMP, le saut vers le début du programme aura alors toujours lieu ;
- de modifier l'appel à la fonction `security_check` par des NOP, il ne va rien se passer ;
- d'extraire l'algorithme de génération de clés afin de fabriquer un générateur de clés (keygen) dans le cas où la clé n'est pas stockée en dur dans le programme ;

6.3 Utilisation d'un désassembleur

Il n'est pas systématiquement nécessaire de faire appel à un débuggeur, parfois une analyse statique du programme peut se révéler suffisante. En regardant les références à des chaînes de caractères dans le code désassemblé, l'utilisateur peut remonter jusqu'à la routine `security_check` et la « corriger ».

7 Exemple

Notre but est d'essayer de cacher les connexions réseau ouvertes sur une machine de type Windows 2000. Pour cela nous allons modifier le programme `netstat.exe` de Windows car il permet à un utilisateur de connaître l'état des connexions de sa machine.

Nous allons analyser le fonctionnement de ce programme et son comportement lorsqu'une connexion est ouverte sur la machine, puis nous allons essayer d'identifier la ou les fonctions responsables de l'affichage.

Ensuite nous réfléchirons à une façon de modifier le code du programme qui nous permettra de sélectionner les lignes à afficher.

Cette analyse s'inspire en partie du travail effectué par `threat` sur `nethide`.

7.1 Analyse des fonctionnalités du programme

La façon la plus simple de connaître le fonctionnement du programme est tout simplement de commencer par l'utiliser. Pour cela, il suffit de lancer une fenêtre cmd.exe sous Windows et d'exécuter la commande `netstat -an` qui va nous lister toutes les connexions réseau ouvertes sur la machine (figure 1).

Nous réalisons donc qu'à chaque nouvelle connexion ouverte, une nouvelle ligne est affichée.

Nous recherchons donc la ou les fonctions permettant l'affichage d'une chaîne de caractères. Pour cela, nous désassemblons l'exécutable avec IDA Pro et nous regardons les fonctions « intéressantes » de l'API win32 qui sont importées par le programme :

- `FormatMessageA`
- `CharToOemBuffA`
- `fprintf`
- `sprintf`

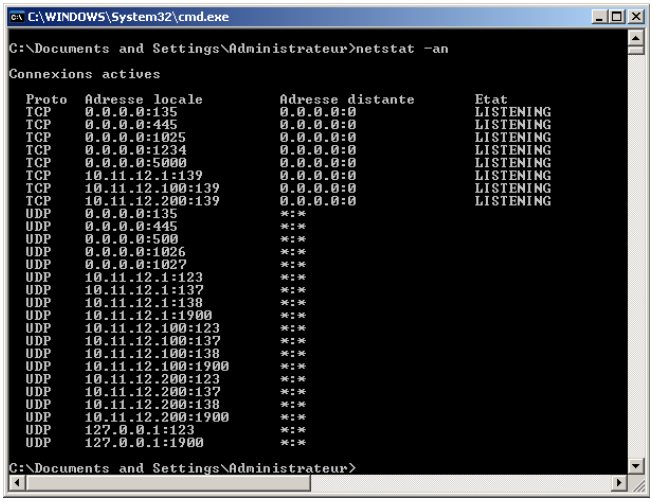


Fig. 1. Liste des connections réseaux ouvertes.

Toutes ces fonctions ne sont pas indispensables à la réalisation de notre objectif. Afin de déterminer quelles vont être les fonctions à modifier, il est nécessaire de se référer à la définition de chacune d'entre elles dans le guide de développement Microsoft MSDN.

Il apparaît que la fonction `FormatMessageA` est utilisée pour formater une chaîne de caractères. C'est exactement la fonction que nous recherchions.

Désormais, nous avons identifié la fonction importée par l'API win32 qui est responsable de l'affichage et du formatage des informations listées par la commande `netstat -an`.

Nous allons maintenant identifier la ou les routines du programme qui font appel à `FormatMessageA`. Pour cela, nous allons utiliser SoftICE et placer un point d'arrêt sur cette fonction. Il est à noter que l'utilisation de SoftICE n'est pas réellement nécessaire car la fonction de formatage de chaînes de caractères n'est apellée que deux fois. Après avoir placé notre point d'arrêt, nous revenons sous Windows et nous relançons la commande `netstat -an`. SoftICE apparaît et nous traçons le programme avec la touche F12 jusqu'à voir les lignes d'affichage dans notre fenêtre `cmd.exe`.

Il apparaît un appel récurrent à la routine se trouvant à l'adresse `001305B`. C'est notre fonction d'affichage !

7.2 Choix d'une fonctionnalité à modifier

Nous savons maintenant que la fonction de formatage des lignes, contenant l'information sur les connexions ouvertes, est appelée par la routine se trouvant à l'adresse `001305B`. C'est cette routine que nous allons analyser afin d'identifier ce qu'il est possible de modifier pour réaliser notre objectif.

Sous IDA Pro, nous allons donc à l'adresse `001305B` afin d'analyser la suite d'instructions assembleur.

```
.text:0100305B ; int __cdecl sub_100305B(int,DWORD dwMessageId,char)
.text:0100305B sub_100305B     proc near        ; CODE XREF: _main+71 p
.text:0100305B                                  ; _main+1C0 p ...
.text:0100305B
.text:0100305B Arguments       = dword ptr -8
.text:0100305B lpszDst         = dword ptr -4
.text:0100305B arg_0           = dword ptr  8
.text:0100305B dwMessageId     = dword ptr  0Ch
.text:0100305B arg_8           = byte ptr  10h
.text:0100305B
.text:0100305B                 push    ebp
.text:0100305C                 mov     ebp, esp
.text:0100305E                 push    ecx
.text:0100305F                 push    ecx
.text:01003060                 lea     eax, [ebp+arg_8]
.text:01003063                 push    ebx
.text:01003064                 mov     [ebp+Arguments], eax
.text:01003067                 lea     eax, [ebp+Arguments]
.text:0100306A                 push    esi
.text:0100306B                 push    eax
.text:0100306C                 xor     esi, esi
.text:0100306E                 lea     eax, [ebp+lpszDst]
.text:01003071                 push    esi
.text:01003072                 push    eax
.text:01003073                 push    esi
.text:01003074                 push    [ebp+dwMessageId]
.text:01003077                 push    esi
.text:01003078                 push    900h
.text:0100307D                 call    ds:FormatMessageA
.text:01003083                 mov     ebx, eax
.text:01003085                 cmp     ebx, esi
.text:01003087                 jnz     short loc_100308D
.text:01003089                 xor     eax, eax
.text:0100308B                 jmp     short loc_10030E7
.text:0100308D ; ---------------------------------------------------------
.text:0100308D
.text:0100308D loc_100308D:                    ; CODE XREF: sub_100305B+2C j
.text:0100308D                 mov     eax, ds:_iob
.text:01003092                 cmp     [ebp+arg_0], 2
.text:01003096                 push    edi
.text:01003097                 lea     esi, [eax+40h]
.text:0100309A                 jz      short loc_100309F
.text:0100309C                 lea     esi, [eax+20h]
.text:0100309F
.text:0100309F loc_100309F:                    ; CODE XREF: sub_100305B+3F j
.text:0100309F                 push    8000h
.text:010030A4                 push    dword ptr [esi+10h]
.text:010030A7                 call    ds:_setmode
.text:010030AD                 mov     edi, [ebp+lpszDst]
.text:010030B0                 pop     ecx
```

```
.text:010030B1                 pop     ecx
.text:010030B2                 xor     eax, eax
.text:010030B4                 or      ecx, 0FFFFFFFFh
.text:010030B7                 repne scasb
.text:010030B9                 not     ecx
.text:010030BB                 dec     ecx
.text:010030BC                 push    ecx
.text:010030BD                 push    [ebp+lpszDst]
.text:010030C0                 push    [ebp+lpszDst]
.text:010030C3                 call    ds:CharToOemBuffA
.text:010030C9                 push    [ebp+lpszDst]
.text:010030CC                 push    offset aS
.text:010030D1                 push    esi
.text:010030D2                 call    ds:fprintf
.text:010030D8                 add     esp, 0Ch
.text:010030DB                 push    [ebp+lpszDst]
.text:010030DE                 call    ds:LocalFree
.text:010030E4                 mov     eax, ebx
.text:010030E6                 pop     edi
.text:010030E7
.text:010030E7 loc_10030E7:                        ; CODE XREF: sub_100305B+30 j
.text:010030E7                 pop     esi
.text:010030E8                 pop     ebx
.text:010030E9                 leave
.text:010030EA                 retn
.text:010030EA sub_100305B     endp
```

Après l'appel à la fonction `FormatMessageA` il y a une suite de cinq instructions très intéressantes :

```
.text:01003083                 mov     ebx, eax
```

Cette instruction copie le nombre de caractères écrits dans le buffer dans `EBX` (`EAX` contient la valeur de retour de l'appel à la fonction `FormatMessageA`).

```
.text:01003085                 cmp     ebx, esi
```

Celle-ci regarde si la chaîne de caractères est vide (`ESI` vaut 0).

```
.text:01003087                 jnz     short loc_100308D
```

Si le nombre de caractères de la chaîne n'est pas nul, alors `ZF` = 0 et le flot d'instructions se déplace à l'adresse `100308D` (affichage de la chaîne de caractères).

```
.text:01003089                 xor     eax, eax
```

Sinon le registre `EAX` est mis à 0.

```
.text:0100308B                 jmp     short loc_10030E7
```

Cette dernière instrction déplace le flot d'instructions à la fin du programme (rien à afficher).

7.3 Réalisation de la modification

Nous venons de voir que si il n'y a rien à afficher, nous allons directement à l'adresse de fin du programme (`10030E7`). Si ce n'est pas le cas, nous nous déplaçons à l'adresse `100308D`, soit 4 octets plus loin dans le flot d'instructions.

Afin de modifier les instructions pour que plus rien ne soit affiché par le programme, et cela, même si il y a quelque chose dans la chaîne de caractères, il suffit de modifier l'adresse du saut conditionnel afin qu'il désigne le saut qui amène à la fin du programme. Typiquement, il suffit de modifier :

```
.text:01003087                  jnz     short loc_100308D
```

par

```
.text:01003087                  jnz     short loc_100308B
```

Cela revient à changer la valeur d'un seul octet et le programme n'affichera plus rien (figure 2). Il suffit ensuite d'éditer l'exécutable `netstat.exe` avec un éditeur hexadécimal et de changer l'octet désiré.

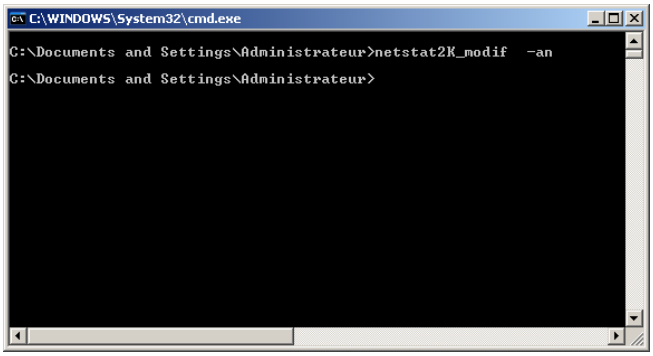


Fig. 2. Affichage du programme netstat modifié

8 Conclusion

Nous venons de montrer que la rétroconception est un moyen efficace d'obtenir de l'information sur un programme exécutable dont nous ne possédons pas les sources.

En modifiant un seul octet du programme, nous avons été capables de modifier son fonctionnement au point de le rendre inefficace.

Pour éviter ce type de manipulations, il existe des mécanismes de protection. Cependant il est illusoire d'espérer mettre au point des techniques réellement incontournables car il y aura toujours un moment où le code machine apparaîtra en clair en mémoire.

9 Remerciements

Je tiens à remercier D.E. pour ses réponses à mes questions, S. Griffe et P. Biondi pour toutes les relectures.

Références

1. IDA Pro, http ://www.datarescue.com
2. Microsoft Visual C++, http ://www.microsoft.com
3. SoftICE, http ://www.numega.com
4. SysInternals, http ://www.sysinternals.com
5. Anti Debugging Tricks, Inbar RAZ, http ://www.bsdg.org/swag/FAQ/0054.PAS.html

Manipulation ELF et sécurité sous UNIX

Julien Vanègue[1] and Sébastien Roy[2]

[1] *may@epita.fr*
[2] *seb@nbs-system.com*

1 Introduction

Cet article aborde les problématiques de la sécurité informatique, notamment dans les domaines d'interception d'information au sein d'un binaire ELF et d'approche statique des protections automatiques contre les failles de sécurité de type buffer overflow. Le shell ELFsh [1] est utilisé comme support logiciel pour l'exposé. Cette boîte a outil de manipulation d'objets ELF se compose de plusieurs composants :

- Le shell lui-même
- La librairie de manipulation ELF : *libelfsh*
- La librairie de manipulation de code : *libasm*

Le shell est un interpreteur de commandes, il permet de faire plusieurs types d'opérations sur les objets. Au niveau de ce composant, chaque opération de manipulation est representée sous forme d'une commande. Chaque opération peut être appelée depuis la ligne de commande du shell UNIX, depuis une session ELFsh interactive, ou depuis un script ELFsh. Ceci est un script ELFsh :

```
#!/usr/bin/elfsh

# Call regular shell
exec cat func2.c
exec gcc -c func2.c
exec cat exec.c
exec gcc exec.c
exec a.out
# Load ELF objects
load a.out
load func2.o
# Insert ET\_REL into ET\_EXEC
reladd 1 2
# Switch on a.out and list SHT
switch 1
s
# Grep in the symbol table for inserted symbols
sym reloc|func2
# Disassemble injected section
D func2.o.text%134
# Dump in hexa injected data objects
X testreloc
# Hijack puts() fonction using GOT infection
got puts
```

```
set 1.got[puts] puts_troj
# Save and execute the output object
save a.out.injected
exec a.out.injected
quit
```

Les mécanismes sous-jacents seront bien sûr detaillés. La bibliothèque de manipulation ELF (*libelfsh*) met a disposition plus de 260 opérations sous forme de fonctions C. La bibliothèque de manipulation d'opcodes IA32 permet la lecture et modification des instructions, elle met à disposition une trentaine de fonctions. Chacune de ces librairies permet la lecture et l'écriture des objets manipulés (a savoir les objets ELF et les instructions assembleur IA32).

2 Présentation des besoins

Il fût nécessaire de coder les composants bas niveau de notre modèle, en délaissant les composants existants, comme *libopcodes* et *libbfd* [2].

Libopcodes n'est malheureusement qu'une bibliothèque de désassemblage qui ne dispose pas vraiment d'une interface d'analyse. *Libasm* permet une représentation interne détaillée de chaque instruction, ce qui permet de simplifier et modulariser les algorithmes, notamment pour les besoins de réencodage (modification) du code, et de traçage de flux.

Libbfd est une bonne bibliothèque de translation binaire, mais dont le backend ELF reste assez pauvre dès que l'on veut disposer de certaines capacités exotiques, comme le détournement de fonctions (ou tout autre capacité dont l'implémentation est dépendante de l'architecture), la relocation d'object non relogeable, ou encore le remapping.

Globalement, les outils GNU ont été codés pour garantir une interface unique sur un grand nombre de formats et d'assembleurs, ce qui est parfait du point de vue de l'éditeur de liens ou d'un outil comme *objdump* [2], mais insuffisant pour un outil d'analyse inverse. Ces dernières années, le besoin de translation binaire se fait de moins en moins ressentir, puisque les principaux UNIX tendent à implementer ELF comme format principal.

2.1 Développement sans source

Dans un cadre de production, ou dans un environnement hostile comme celui d'un serveur compromis ou vulnérable, il est fréquent de ne pas disposer intégralement des source du systeme d'exploitation (et de ses packages) sur la machine locale. Il est aussi fréquent de ne pas pouvoir transférer ces sources intégralement dans un laps de temps raisonnable — elles ne sont parfois tout simplement pas disponible — parfois même la recompilation n'est pas désirée. Qu'il s'agisse de logiciels propriétaires, ou d'exploits binaires, le besoin de disposer d'une interface claire de manipulation se fait ressentir. Nous nous attarderons sur deux concepts :

- la résidence : étude de l'injection ;
- l'interception : étude du détournement.

Une telle interface permet à la fois l'ajout de fonctionnalités et la modification de fonctionnalités existantes, le tout sans recompilation.

2.2 Protection système : ASLR

L'ASLR (*Address Space Layout Randomization*) permet de bloquer l'exploitation des attaques basées sur la prédiction d'adresses, comme les buffer overflow par *return into libc* ou *return into PLT* [3] .

Le projet PaX [4] propose une implémentation dynamique de l'ASLR, qui consiste à modifier directement le processus au moment de l'exécution. Cette implémentation au niveau noyau fait souffrir la machine protegée INTEL d'une chute de performance de 200% (sur une compilation de noyau Linux avec l'option RANDEXEC activée sur tous les binaires), due aux techniques utilisées (notamment par le détournement du gestionnaire de fautes de page depuis le noyau).

ELFsh propose une implementation expérimentale de l'ASLR en userland sur les machines INTEL, qui permet de conserver la performance d'origine, puisque le remapping est effectué statiquement directement dans le binaire, en modifiant l'adresse de chaque segment mappé, et en relogeant l'exécutable après avoir reconstruit ses tables de relocation (qui sont strippées par defaut dans les objets ELF `ET_EXEC`).

Il existe une implementation statique fiable de l'ASLR (et_dyn.zip) , qui consiste à relinker les binaires d'une distribution en bibliothèqyes dynamiques, il faut pour cela disposer de tous les objets relogeables qui composent l'executable (tous les `.o`), ainsi que les `Makefile` d'origine du package, ce qui n'est pas applicable aisément dans un environnement de production, ou dans un environnement sans source.

Nous verrons que le problème reste ouvert, puisque notre implementation ASLR admet toujours des faux positifs de relocation, qui ne sont pas toujours évidents à détecter de manière automatique.

3 Le flux de contrôle

Le flux de contrôle est le chemin d'exécution du programme. Il est intéressant de l'étudier pour plusieurs raisons. Dans le cadre de la manipulation binaire, nous sommes particulièrement intéressés par la mise en place de *hooks*, c'est-à-dire de points de détournement. Les utilisations sont multiples : audit, logging, déclenchement de fonctionalités supplémentaires, etc.

Certains modèles de hooks [6] sont intéressants et peuvent être appliqués en partie sur des binaires ELF. L'inconvénient de [6] est que ce modèle utilise un système dynamique de modification/restauration d'octets au début de la fonction, et qu'il est impossible de restaurer des octets originaux au moment de l'exécution, puisque les fonctions d'un objet ELF sont contenues dans les segments exécutables en lecture seule (c'est-à-dire un `PT_LOAD r-x`). Bien qu'il soit possible de rendre une zone accessible en écriture pour un laps de temps fini par l'appel système `mprotect()`, nous éviterons de l'utiliser puisqu'il est en général filtré par les protections non-exécutable comme PaX.

De plus, le contexte utilisateur veut que les interfaces de résolution d'adresses des fonctions soient différentes pour les fonctions propres au binaire, et les fonctions des librairies dont dépend le binaire. Cette dernière utilise un mécanisme de patch au besoin, au moment de l'exécution, en utilisant les sections `.got` et `.plt` [1,7] : nous l'expliciteront dans la section 3.2.

3.1 Interception statique

Dans un premier temps, voyons les points d'entrée statiques du binaire. Ils existent quelque soit le contenu de l'exécutable, et peuvent être modifiés avec les commandes

write et *set* dans ELFsh [1]. On distingue le point d'entrée du programme (champ `e_entry` dans le header ELF, qui pointe sur la fonction `_start`, généralement située au debut de la section `.text`), la fonction `_init` (dans la section `.init`), la fonction `_fini` (dans la section `.fini`), les constructeurs (section `.ctors`), et les destructeurs (section `.dtors`), et enfin la fonction `main`.

```
$ elfsh -f ./a.out -D _start

08048310 [foff: 784] _start + 0    xor   %ebp,%ebp
08048312 [foff: 786] _start + 2    pop   %esi
08048313 [foff: 787] _start + 3    mov   %esp,%ecx
08048315 [foff: 789] _start + 5    and   $FFFFFFF0,%esp
08048318 [foff: 792] _start + 8    push  %eax
08048319 [foff: 793] _start + 9    push  %esp
0804831A [foff: 794] _start + 10   push  %edx
0804831B [foff: 795] _start + 11   push  $<.fini>
08048320 [foff: 800] _start + 16   push  $<.init>
08048325 [foff: 805] _start + 21   push  %ecx
08048326 [foff: 806] _start + 22   push  %esi
08048327 [foff: 807] _start + 23   push  $<main>
0804832C [foff: 812] _start + 28   call  <__libc_start_main@@GLIBC_2.0>
08048331 [foff: 817] _start + 33   hlt
08048332 [foff: 818] _start + 34   nop
08048333 [foff: 819] _start + 35   nop

$ elfsh -f ./a.out -D '^\.init:0%37' -D '^\.fini:0%28'

08048298 [foff: 664] .init + 0     push  bp
08048299 [foff: 665] .init + 1     mov   sp,%ebp
0804829B [foff: 667] .init + 3     sub   4,%esp
0804829E [foff: 670] .init + 6     push  bx
0804829F [foff: 671] .init + 7     call  init + 12>
080482A4 [foff: 676] .init + 12    pop   bx
080482A5 [foff: 677] .init + 13    add   29C,%ebx
080482AB [foff: 683] .init + 19    call  etext>
080482B0 [foff: 688] .init + 24    call  rame_dummy>
080482B5 [foff: 693] .init + 29    call  _do_global_ctors_aux>
080482BA [foff: 698] .init + 34    pop   bx
080482BB [foff: 699] .init + 35    leave
080482BC [foff: 700] .init + 36    ret

0804842C [foff: 1068] .fini + 0    push  %ebp
0804842D [foff: 1069] .fini + 1    mov   %esp,%ebp
0804842F [foff: 1071] .fini + 3    sub   $14,%esp
08048432 [foff: 1074] .fini + 6    push  %ebx
08048433 [foff: 1075] .fini + 7    call  <_fini + 12>
08048438 [foff: 1080] .fini + 12   pop   %ebx
08048439 [foff: 1081] .fini + 13   add   $1108,%ebx
0804843F [foff: 1087] .fini + 19   nop
08048440 [foff: 1088] .fini + 20   call  <__do_global_dtors_aux>
```

```
08048445 [foff: 1093] .fini + 25   pop   %ebx
08048446 [foff: 1094] .fini + 26   leave
08048447 [foff: 1095] .fini + 27   ret

$ elfsh -f ./a.out -D __do_global_ctors_aux%36

08048400 [foff: 1024] + 0    push  %ebp
08048401 [foff: 1025] + 1    mov   %esp,%ebp
08048403 [foff: 1027] + 3    sub   $14,%esp
08048406 [foff: 1030] + 6    push  %ebx
08048407 [foff: 1031] + 7    mov   $<.ctors>,%ebx
0804840C [foff: 1036] + 12   cmp   $FFFFFFFF,<.ctors>
08048413 [foff: 1043] + 19   je    <__do_global_ctors_aux + 33>
08048415 [foff: 1045] + 21   mov   (%ebx),%eax
08048417 [foff: 1047] + 23   call  *%eax
08048419 [foff: 1049] + 25   add   $FFFFFFFC,%ebx
0804841C [foff: 1052] + 28   cmp   $FFFFFFFF,(%ebx)
0804841F [foff: 1055] + 31   jne   <__do_global_ctors_aux + 21>
08048421 [foff: 1057] + 33   pop   %ebx
08048422 [foff: 1058] + 34   leave
08048423 [foff: 1059] + 35   ret

$ elfsh -D __do_global_dtors_aux%79

08048360 [foff: 864] + 0     push  %ebp
08048361 [foff: 865] + 1     mov   %esp,%ebp
08048363 [foff: 867] + 3     sub   $8,%esp
08048366 [foff: 870] + 6     cmp   $0,<completed.4>
0804836D [foff: 877] + 13    jne   <__do_global_dtors_aux + 77>
0804836F [foff: 879] + 15    jmp   <__do_global_dtors_aux + 35>
08048371 [foff: 881] + 17    mov   <p.3>,%eax
08048376 [foff: 886] + 22    lea   4(%eax),%edx
08048379 [foff: 889] + 25    mov   %edx,<p.3>
0804837F [foff: 895] + 31    mov   (%eax),%eax
08048381 [foff: 897] + 33    call  *%eax
08048383 [foff: 899] + 35    mov   <p.3>,%eax
08048388 [foff: 904] + 40    cmp   $0,(%eax)
0804838B [foff: 907] + 43    jne   <__do_global_dtors_aux + 17>
0804838D [foff: 909] + 45
             mov  $<__deregister_frame_info@@GLIBC_2.0>, %eax
08048392 [foff: 914] + 50    test  %eax,%eax
08048394 [foff: 916] + 52    je    <__do_global_dtors_aux + 67>
08048396 [foff: 918] + 54    add   $FFFFFFF4,%esp
08048399 [foff: 921] + 57    push  $<.eh_frame>
0804839E [foff: 926] + 62    call  <__deregister_frame_info@@GLIBC_2.0>
080483A3 [foff: 931] + 67    mov   $1,<completed.4>
080483AD [foff: 941] + 77    leave
080483AE [foff: 942] + 78    ret

$ elfsh -f /bin/telnet -ctors -dtors
```

```
[Constructors array .::. CTORS]
[Object /bin/telnet]

[00]  0xffffffff        <?>
[01]   0x804e0b0        <.text + 15088>
[02]   0x804e970        <.text + 17328>
[03]   0x805291c        <.text + 33628>
[04]   0x80547d0        <.text + 41488>
[05]   0x8054bec        <.text + 42540>
[06]       (nil)        <?>

[Destructors array .::. DTORS]
[Object /bin/telnet]

[00]  0xffffffff        <?>
[01]   0x804e0c8        <.text + 15112>
[02]   0x804e988        <.text + 17352>
[03]   0x8052934        <.text + 33652>
[04]   0x80547e8        <.text + 41512>
[05]   0x8054c04        <.text + 42564>
[06]       (nil)        <?>

$
```

Grâce à ces dumps, nous déduisons le chemin d'exécution du programme :

```
DEBUT DU PROCESS

_start() =>
          __libc_start_main() =>
               [...] *
               _init() =>
                __do_global_ctors_aux() =>
                                        ctors[01]()
                                        ctors[02]()
                                        ctors[..]()
                                        ctors[NN]()
               main()  =>
                                        [...]
               exit()  =>
               Tables __atexit/__on_exit =>
                         __do_global_dtors_aux() =>
                                        dtors[01]()
                                        dtors[02]()
                                        dtors[..]()
                                        dtors[NN]()
FIN DU PROCESS
```

La fonction `__libc_start_main()` se trouve dans la libc, ou elle est chargée d'initialiser l'espace mémoire du processus. Cette étape, marquée d'une étoile, est hautement

dépendante du système d'exploitation ; elle est en général très peu documentée [8] : l'éditeur de liens dynamique y prend la main pour la première fois, il mappe alors les dépendences et effectue leurs relocations, puis appelle les constructeurs par la fonction `_init()`, enfin appelle la fonction `main()`. Au retour du `main()`, `exit()` est directement appelée avec la valeur de retour, qui à son tour se charge de lancer tous les destructeurs, le processus est alors terminé.

3.2 Interception dynamique

Le rôle de l'éditeur de liens dynamique ne se résume pas à la construction du processus avant l'appel du `main()`. Il est aussi chargé d'assurer la résolution d'adresses entre les objets, ce qui leur permet d'assurer un transfert de contrôle au besoin, au moment de l'exécution. En d'autres termes, il permet de résoudre les adresses des fonctions et variables externes à l'objet courant, de manière à pouvoir s'en servir.

Voyons de plus près les caractèristiques d'une bibliothèque dynamique (les entêtes de sections non mappées ont été retirées) :

```
 $ elfsh -f /lib/ld-linux.so.2 -s | grep -v nil

 [01] 0x94       a------ .hash               foff:000148 sz:000836
 [02] 0x3d8      a------ .dynsym             foff:000984 sz:001760
 [03] 0xab8      a------ .dynstr             foff:002744 sz:003468
 [04] 0x1844     a------ .gnu.version        foff:006212 sz:000220
 [05] 0x1920     a------ .gnu.version_d      foff:006432 sz:000200
 [06] 0x19e8     a------ .rel.data           foff:006632 sz:000096
 [07] 0x1a48     a------ .rel.got            foff:006728 sz:000536
 [08] 0x1c60     a------ .rel.plt            foff:007264 sz:000224
 [09] 0x1d40     a-x---- .plt                foff:007488 sz:000464
 [10] 0x1f10     a-x---- .text               foff:007952 sz:065161
 [11] 0x11da0    a------ .rodata             foff:073120 sz:010648
 [12] 0x15740    aw----- .data               foff:083776 sz:000180
 [13] 0x157f4    aw----- .got                foff:083956 sz:000392
 [14] 0x1597c    aw----- .dynamic            foff:084348 sz:000136
 [15] 0x15a04    -w----- .sbss               foff:084512 sz:000000
 [16] 0x15a20    aw----- .bss                foff:084512 sz:001004

 $ elfsh -f /lib/ld-linux.so.2 -p

4 [Program header table .:::. PHT]
 [Object /lib/ld-linux.so.2]

 [00] 0x00000000 -> 0x00014738 r-x memsz:83768
                                  foff:00000 => Loadable segment
 [01] 0x00015740 -> 0x00015E0C rw- memsz:01740
                                  foff:83776 => Loadable segment
 [02] 0x0001597C -> 0x00015A04 rw- memsz:00136
                                  foff:84348 => Dynamic info

 [Program header table .:::. SHT correlation perspective]
 [Object /lib/ld-linux.so.2]
```

```
[*] SHT is not stripped

[00] {PT_LOAD}          .hash .dynsym .dynstr .gnu.version .gnu.version_d
.rel.data .rel.got .rel.plt .plt .text .rodata
[01] {PT_LOAD}         .data .got .dynamic
[02] {PT_DYNAMIC}      .dynamic
```

On distingue clairement que les adresses dans la SHT et la PHT sont relatives par rapport à l'adresse base de la bibliothèque, c'est à dire par rapport au début du fichier (offset de fichier 0), ce qui explique que l'étape de relocation soit obligatoire pour les librairies `ET_DYN`, et donc un peu plus complexe que la relocation des binaires `ET_EXEC`. On distinguera notamment des sections de relocation supplémentaire comme `.rel.text` ou `.rel.data` dans ces bibliothèques.

Quant à la relocation à la volée, elle s'effectue aussi bien sur les objets de type `ET_EXEC` que sur les objets `ET_DYN`, en utilisant les sections `.got`, `.plt`, `.rel.got`, `.rel.plt`, `.dynsym` et `.dynstr`. Cette étape de relocation va permettre de modifier l'image mémoire de l'objet au moment ou le programme en a besoin, et non au moment du mapping initial. En fait, elle se resume à la mise à jour de pointeurs dans un tableau (la section `.got`) de manière indépendante pour chacune des entrée, lors de leur première utilisation.

Chaque entrée de `.got` correspond à l'adresse d'une fonction externe à l'objet, mais dont il dépend. Par exemple, un programme qui dépend de la libc, devra reloger les entrées de `.got` pour toutes les fonctions de la libc qu'il utilise. Il n'est pas possible de prédire ces adresses à la compilation pour la même raison que celles cités précédemment, c'est à dire qu'une bibliothèque peut être mappée n'importe ou dans le processus. Voici à quoi ressemble une section `.got` dans le binaire :

```
$ elfsh -f /bin/ls -got '0]'

[Global Offset Table .::. GOT]
[Object /bin/ls]

[000] 0x080535B8        .dynamic
[010] 0x08049036        _obstack_begin + 6
[020] 0x080490D6        __errno_location + 6
[030] 0x08049176        fputs + 6
[040] 0x08049216        printf + 6
[050] 0x080492B6        __lxstat64 + 6
[060] 0x08049356        isatty + 6
[070] 0x080493F6        fwrite + 6

$ elfsh -f /bin/ls -got | wc -l
    85
$
```

Les trois premières entrées de `.got` sont des entrées spéciales, dont nous allons éclaircir le fonctionnement. En fait, chaque entrée de `.got` pointe par défaut (c'est-à-dire quand la `.got` n'a pas encore été relogée dans le binaire) sur l'entrée de la section `.plt` pour cette fonction, exceptés les trois premières, qui sont respectivement utilisées

pour contenir l'adresse de la section `.dynamic`, l'adresse de `dl-resolve`, et l'adresse de la structure `linkmap` de l'objet courant [8].

Voyons à quoi ressemble une section *.plt* sur architecture `INTEL` :

```
 08048FB0 .plt + 0                      push          .got + 4
 08048FB6 .plt + 6                      jmp           *.got + 8
 08048FBC .plt + 12                     add           %al,(%eax)
 08048FBE .plt + 14                     add           %al,(%eax)

 08048FC0 .plt + 16                     jmp           *.got + 12
 08048FC6 .plt + 22                     push          $0
 08048FCB .plt + 27                     jmp           .plt

 08048FD0 .plt + 32                     jmp           *.got + 16
 08048FD6 .plt + 38                     push          $8
 08048FDB .plt + 43                     jmp           .plt
[...]
 08049030 .plt + 128                    jmp           *.got + 40
 08049036 .plt + 134                    push          $38
 0804903B .plt + 139                    jmp           .plt
[...]
```

Voici la meme chose sous `SPARC` :

```
[...]
 00031484 <.plt>:
   31484:       00 00 00 00     unimp  0
   31488:       00 00 00 00     unimp  0
   3148c:       00 00 00 00     unimp  0
[...]
   314a8:       00 00 00 00     unimp  0
   314ac:       00 00 00 00     unimp  0
   314b0:       00 00 00 00     unimp  0

   314b4:       03 00 00 30     sethi  %hi(0xc000), %g1
   314b8:       30 bf ff f3     b,a    0x31484
   314bc:       01 00 00 00     nop

   314c0:       03 00 00 3c     sethi  %hi(0xf000), %g1
   314c4:       30 bf ff f0     b,a    0x31484
   314c8:       01 00 00 00     nop
[...]
```

Bien qu'elle contienne du code, on peut dire que la `.plt` est une table, puisque chacune de ses entrées fait 16 octets (12 octets sous `SPARC`) et contient 3 instructions [`jmp`, `push`, `jmp`] ou [`sethi`, `ba`, `nop`] sous `SPARC`. On notera que la première entrée est spéciale sur les 2 architectures (les 4 premieres pour `SPARC`), nous allons expliquer en quoi tout de suite en prenant pour témoin la machine `INTEL`.

Les mechanismes sont similaires, excepté le fait que la `.got` n'est pas utilisée pour le transfert de contrôle sous `SPARC`, mais que l'entrée de la `.plt` est patchée directement.

Lors de l'appel d'une fonction externe à l'objet (par exemple, l'appel à `printf` de la libc), le programme doit passer par un `call` sur l'entrée de la `.plt` correpondante à la fonction. Par exemple, la derniere entrée affichée (adresse `08049030`) correspond à la dixième entrée de la `.got` ; le premier `jmp` utilise la valeur de `.got` [10]. On appelle ce type de saut *indirect* car il n'utilise pas une valeur immédiate codée dans l'instruction elle même, mais une valeur lue à l'adresse indiquée par son opérande.

Ainsi, la valeur par défaut de l'entrée de la `.got` va etre utilisée par l'entrée courante de la *.plt*. On constate que l'adresse qu'elle contient pointe sur la deuxiéme instruction de l'entrée correspondante dans la `.plt`. Cette deuxième instruction `push` permet de sauver sur la pile l'offset de l'entrée de relocation correspondant à l'entrée de la `.got` que nous devons patcher.

L'offset $38 correspond à un décalage de 0x38 octets par rapport à l'adresse basse de la table de relocation (qui est disponible dans le segment `PT_DYNAMIC` pour le linker dynamique). Chaque entrée de la table (`.rel.plt`) est codée sur `sizeof(Elf32_Rel)`, ou `sizeof(Elf32_Rela)`, l'entrée à l'offset 0x38 sera utilisée par `dl-resolve()` pour connaître les informations de relocation relatives à cette entrée.

Enfin, la première entrée (entrée spéciale) de la `.plt` est appelée par le deuxième `jmp`. Elle-même utilise la troisième entrée de la *.got* pour invoquer la fonction de relocation à la volée, qui se trouve dans le linker dynamique. Cette troisième entrée `.got` [2] est également réservée : elle est renseignée avant le lancement de la fonction `main()`. La fonction chargée de la mise à jour de la `.got` s'appelle en général `dl-resolve()`, même si ce n'est pas standardisé.

Quand l'entrée de la `.got` est remplie, la fonction de résolution à la volée appelle la fonction externe du programme, dans notre cas `printf()`. Ainsi les prochains appels à `printf()` utiliseront directement la valeur renseignée de la `.got`, sans repasser par le linker dynamique et sa fonction `dl-resolve`, puisque le premier `jmp` de l'entrée de la `.plt` utilisera maintenant la valeur renseignée de l'entrée de la `.got` pour `printf()`.

Il est donc aisé de détourner une fonction en modifiant son entrée dans la `.got`, ainsi `dl-resolve()` ne sera jamais appelé pour cette entrée, et une fonction tierce pourra être appelée à la place.

Détournons malloc :

```
[ELFsh-0.5b5]$ load /bin/ls

[*] New object /bin/ls loaded on Thu May  1 00:18:15 2003

[ELFsh-0.5b5]$ got malloc

[Global Offset Table .::. GOT]
[Object /bin/ls]

[023] 0x08049106        <malloc + 6>

[ELFsh-0.5b5]$ set 1.got[23] 0x42424242

[*] Field set succesfully

[ELFsh-0.5b5]$ save /tmp/ls.bad

[*] Object /tmp/ls.bad save successfully
```

```
[ELFsh-0.5b5]$ quit

[*] Unloading object 1 (/bin/ls) *

        Good bye ! .::. The ELF shell 0.5b5

$ elfsh -f /tmp/ls.bad -got 42424242

[*] Object /tmp/ls.bad has been loaded (O_RDONLY)

[Global Offset Table .::. GOT]
[Object /tmp/ls.bad]

[023] 0x42424242        <?>

[*] Object /tmp/ls.bad unloaded

 $ /tmp/ls.bad
Segmentation fault (core dumped)
$ gdb /tmp/ls.bad --core=/tmp/core

[...]

#0  0x42424242 in ?? ()
(gdb) bt
#0  0x42424242 in ?? ()
#1  0x0804e1d6 in strcpy () at ../sysdeps/generic/strcpy.c:31
#2  0x0804a29c in strcpy () at ../sysdeps/generic/strcpy.c:31
#3  0x08049634 in strcpy () at ../sysdeps/generic/strcpy.c:31
#4  0x400502eb in __libc_start_main
                  (main=0x80495a8 <strcpy+408>, argc=1, ...)
(gdb)
```

À noter qu'il est possible de forcer le renseignement complet de la `.got` avant que le `main()` ne s'exécute. Pour cela, il suffit de créer une variable d'environnement `LD_BIND_NOW` et de la mettre à 1, elle sera directement interpretée par le linker dynamique au lancement du programme. Dans ce cas, toutes les entrées de la `.got` sont mises à jour au démarrage du programme, et le détournement par `.got` n'est pas possible. Il faut alors patcher la section `.plt` directement pour détourner une fonction externe [9] , et ainsi utiliser un pointeur autre que celui renseigné par le linker dynamique dans la section `.got`. Sur architecture SPARC, l'infection `.plt` est toujours nécessaire, puisque cette architecture n'utilise pas la `.got` pour la résolution d'adresse de fonction externe, le lecteur est invité a se réferrer à la documentation de la fonction `elfsh_hijack_function` de `ELFsh` .

4 Résidence

Voyons comment nous pouvons insérer du code et des donnees excédentaires dans un binaire ELF.

4.1 Première approche : le code PIC

Nous pouvons choisir la solution simple du code indépendant de sa position (PIC), pour qu'il puisse s'exécuter n'importe ou dans le processus. Dans ce cas, nous n'avons pas à nous soucier de reloger le code avant de l'exécuter.

Cette méthode est également utilisée pour les shellcodes, c'est-à-dire les codes injectés lors d'exploitations de type buffer overflow. Ainsi le code peut s'exécuter aussi bien sur la pile que dans le tas, sans se soucier de son adresse de base.

Toutefois, il devient plus délicat d'utiliser les fonctions de l'objet hôte, ou les fonctions exportées de bibliothèques dans ce type de code. Il est alors nécéssaire de disposer d'un propre moteur de relocation statique, et ainsi reloger certaines parties du code injecté, pour le renseigner des éventuelles adresses absolues dont il a besoin.

Dans ce chapitre, nous expliciterons en quoi cette méthode est naïve, et devient trop complexe pour des injections de moyenne ou grosse envergure. Nous proposerons plusieurs méthodes alternatives.

4.2 Zones d'alignement

Chaque segment de type `PT_LOAD` doit être aligné sur une page sur architecture INTEL (4096 bytes) et sur architecture SPARC (8192 bytes) . Cela laisse de la place pour un parasite qui viendrait se loger dans le padding d'un exécutable `PT_LOAD` . Comme le padding n'est pas physiquement inséré dans le fichier, une telle résidence aura pour consequence de décaler tous les offsets de fichier pour chaque section se trouvant après le parasite, mais pas les adresses virtuelles des segments. Voyons les `PT_LOAD` decrit par la PHT sous SPARC :

```
[02] PT_LOAD 0x10000 r-x memsz:0070100 foff:00000000 filesz:0070100
[03] PT_LOAD 0x311d8 rwx memsz:0002624 foff:00070104 filesz:0001816
```

et sous INTEL :

```
[02] PT_LOAD 0x08048000 r-x memsz:041916 foff:000000 filesz:041916
[03] PT_LOAD 0x080533BC rw- memsz:001356 foff:041916 filesz:000676
```

L'éditeur de liens s'assure que l'espace d'adressage du padding est respecté.

Par contre, il n'est pas vraiment envisageable d'utiliser le padding de chacune des sections, la taille maximale du padding étant 3 bytes — car l'addresse d'une section doit etre alignée sur 4 bytes — nous ne pourrions meme pas coder une instruction de branchement tel que le `jmp` indirect, qui permettrait de passer le contrôle à une partie du parasite placée dans le padding des sections suivantes.

4.3 Ajout de section

Synchroniser la perspective par sections de l'éditeur de liens est intéressant dès lors que l'on souhaite relinker des objets entre eux, ou debugger le binaire hôte de manière décente. Nous pouvons injecter les sections de plusieurs façons :

- section mappée de code ;
- section mappée de données ;
- section non mappée.

Les sections non mappées sont les plus simples à injecter, elles sont souvent utiles pour stocker des informations, comme des tables de références, des tables de symboles excédentaires, ou toutes autres tables qui peuvent faciliter l'analyse par la suite.

```
$ elfsh -f fake_aout -q -s
[SECTION HEADER TABLE .::. SHT is not stripped]
[Object fake_aout]

[00] (nil)      ---                 foffset(00000000) size(00000000)
[01] 0x80480f4  a-- .interp         foffset(00000244) size(00000019)
[...]
[21] 0x804963c  aw- .bss            foffset(00001596) size(00000024)
[22] (nil)      --- .stab           foffset(00001596) size(00002388)
[23] (nil)      --- .stabstr        foffset(00003984) size(00006498)
[24] (nil)      --- .comment        foffset(00010482) size(00000228)
[25] (nil)      --- .note           foffset(00010710) size(00000120)
[26] (nil)      --- .shstrtab       foffset(00010830) size(00000232)
[27] (nil)      --- .symtab         foffset(00012252) size(00001264)
[28] (nil)      --- .strtab         foffset(00013516) size(00000569)
[29] (nil)      --- .newsect        foffset(00014095) size(00000022) *

$
```

Les sections de données sont en général ajoutées après la section `.bss`, qui est la dernière section du segment de données. La `.bss` n'est pas physiquement dans le fichier. Il possède seulement une entrée dans la *SHT* entre la dernière section mappée et la premiere section non mappée, ce qui permet à l'éditeur de liens de l'identifier comme la zone des données non initialisées.

```
$ elfsh -q -f fake_aout -s
[SECTION HEADER TABLE .::. SHT is not stripped]
[Object fake_aout]

[00] (nil)      ---                 foffset(00000000) size(00000000)
[01] 0x80480f4  a-- .interp         foffset(00000244) size(00000019)
[...]
[20] 0x8049614  aw- .got            foffset(00001556) size(00000040)
[21] 0x804963c  aw- .bss            foffset(00001596) size(00000024)
[22] 0x8049654  a-x .newsect        foffset(00001620) size(00000030) *
[23] (nil)      --- .stab           foffset(00001650) size(00002388)
[...]
[28] (nil)      --- .symtab         foffset(00012306) size(00001264)
[29] (nil)      --- .strtab         foffset(00013570) size(00000569)
```

Les sections de code peuvent être mappée à divers endroits. Une première approche consiste à les mapper dans le padding du segment exécutable, mais cela limite la taille de notre code injecté. Nous pouvons également les injecter en premier, entre la première section (section `NULL`) et la deuxième (généralement `.interp`). Cette deuxième approche est intéressante du fait qu'elle permet d'injecter autant de données que l'on souhaite, en étendant l'espace d'adressage par le haut, sans décaler les adresses virtuelles des sections suivantes.

```
$ elfsh -f fake_aout -q -s
[SECTION HEADER TABLE .:::. SHT is not stripped]
[Object fake_aout]

[00] (nil)      ---                 foffset(00000000) size(00000000)
[01] 0x80470f4  a-x .newsect        foffset(00000244) size(00004096) *
[02] 0x80480f4  a-- .interp         foffset(00004340) size(00000019)
[03] 0x8048108  a-- .note.ABI-tag   foffset(00004360) size(00000032)
[04] 0x8048128  a-- .hash           foffset(00004392) size(00000056)
[05] 0x8048160  a-- .dynsym         foffset(00004448) size(00000144)
[...]
[27] (nil)      --- .shstrtab       foffset(00014910) size(00000231)
[28] (nil)      --- .symtab         foffset(00016332) size(00001264)
[29] (nil)      --- .strtab         foffset(00017596) size(00000569)
```

Bien sûr, nous devons synchroniser la perspective du système, en modifiant la PHT (particulièrement les champs de l'entête des segments mappés) afin que les données des sections injectées soient inclues dans le segment approprié. L'algorithme d'insertion de section se resume à :

1. La creation de l'en-tête
2. L'insertion de l'en-tête dans la SHT (présente dans une zone nonmappée)
3. L'insertion des données initiales dans la nouvelle section

À chaque insertion de section, les tables de symboles doivent être synchronisées, plus precisément le champ d'index de section (`st_sctndx`) doit être mis à jour pour chaque entrée de la table des symboles. Ce champ est notamment utilisé pour indiquer que le symbole est non-defini (`SHN_UNDEF`) ou emphcommon (`SHN_COMMON`), autrement il indique la section parente de l'objet référencé par le symbole. De même, l'index de la section des noms de sections doit être mis à jour dans l'entête ELF (`e_shstrndx`) si la section injectée a été inserée apres `.shstrtab`.

4.4 Extension de segments

Dans le cas d'une injection de section post-bss, nous devons fixer le segment `PT_LOAD` autorisé en écriture (`rw-`) pour qu'il prenne en compte l'insertion physique du bss (en modifiant la taille physique `p_filesz` par la valeur de la taille mémoire `p_memsz` de l'entrée de la PHT) et l'insertion de la nouvelle section.

- Pour chaque entrée de la PHT
  - Si le segment est de type `PT_LOAD` et accessible en écriture (`aw-`)
    - Changer `p_filesz` pour la meme valeur que `p_memsz`
    - Étendre `p_filesz` et `p_memsz` de la taille de la section injectée

Dans le cas d'une injection de section pre-`interp`, nous devons décrémenter l'adresse de base du segment `PT_LOAD` exécutable (`r-x`) pour qu'il prenne en compte l'insertion de la nouvelle section de code au dessus de la section `.interp`. Cette methode permet d'éviter de reloger l'exécutable lui-même à chaque insertion, et n'admet pas de limite de taille contrairement à l'injection par padding de segment.

- Pour chaque entrée de la PHT
  - Si le segment est de type `PT_LOAD` executable (`a-x`)
    - Ajouter la longueur de la section injectée à `p_filesz` et `p_memsz`

  - Soustraire la taille de la section injectée à `p_vaddr` et `p_paddr`
- Sinon si le segment est de type `PT_PHDR`
  - Soustraire la taille de la section injectée à `p_vaddr` et `p_paddr`
- Sinon
  - Ajouter la taille de la section injectée à `p_offset`

L'avantage ultime de cette méthode, est qu'elle est compatible avec les environnements de type non-executable, qui refusent toute exécution de code dans le processus, en dehors des segments exécutables `PT_LOAD` (dans le but principal de protéger des buffer overflow dont le shellcode s'exécute dans la pile ou le tas). Il n'existe pas d'autre méthode pour étendre plus d'une page de code dans ce segment dans un environnement non exécutable. Si l'environnement n'est pas non-exécutable, il est possible d'utiliser l'injection post-bss pour insérer du code.

4.5 La section `.dynamic`

Cette section permet d'indexer toutes les tables nécessaires au linkage dynamique. Voici sa representation :

```
$ elfsh -f /bin/ls -d

[*] Object /bin/ls has been loaded (O_RDONLY)

[SHT_DYNAMIC]
[Object /bin/ls]

[00] Name of needed library            =>    librt.so.1 {DT_NEEDED}
[01] Name of needed library            =>     libc.so.6 {DT_NEEDED}
[02] Address of init function          =>    0x08048F88 {DT_INIT}
[03] Address of fini function          =>    0x0804F45C {DT_FINI}
[04] Address of symbol hash table      =>    0x08048128 {DT_HASH}
[05] Address of dynamic string table   =>    0x08048890 {DT_STRTAB}
[06] Address of dynamic symbol table   =>    0x08048380 {DT_SYMTAB}
[07] Size of string table              =>     821 bytes {DT_STRSZ}
[08] Size of symbol table entry        =>      16 bytes {DT_SYMENT}
[09] Debugging entry (unknown)         =>    0x00000000 {DT_DEBUG}
[10] Processor defined value           =>    0x0805348C {DT_PLTGOT}
[11] Size in bytes for .rel.plt        =>     560 bytes {DT_PLTRELSZ}
[12] Type of reloc in PLT              =>            17 {DT_PLTREL}
[13] Address of .rel.plt               =>    0x08048D58 {DT_JMPREL}
[14] Address of .rel.got section       =>    0x08048D20 {DT_REL}
[15] Total size of .rel section        =>      56 bytes {DT_RELSZ}
[16] Size of a REL entry               =>       8 bytes {DT_RELENT}
[17] UNK 6FFFFFFE                      =>           [?] {UNK 6FFFFFFE}
[18] UNK 6FFFFFFF                      =>           [?] {UNK 6FFFFFFF}
[19] UNK 6FFFFFF0                      =>           [?] {UNK 6FFFFFF0}

[*] Object /bin/ls unloaded
```

Nous trouvons depuis cette table les addresses virtuelles des tables de relocation, tables des symboles, tables de hash des symboles. Nous trouvons également des entrées

de type DT_NEEDED qui sont intéressantes pour la résidence, puisqu'elles indexent les dépendances (bibliothèqyes : type ELF ET_DYN) du binaire. Nous pouvons donc :

- ajouter des entrées de type DT_NEEDED ;
- modifier une entrée existante pour la transformer en DT_NEEDED.

L'ajout d'une entrée peut être problématique. En effet, la section .dynamic se trouve (en général) juste avant le .bss, de manière générale, entre les sections de données initialisées (.data, .rodata, etc) et la section des données non initialisées (.bss). Ainsi, il faudra décaler la section .bss, et mettre à jour toutes ses références dans les autres sections (notamment dans .text), ce qui n'est pas toujours trivial, comme explicité dans la partie sur la problématique de l'ASLR.

La modification d'entrée est plus subtile. Chaque entrée est structurée comme telle :

```
typedef struct
{
 Elf32_Sword   d_tag;                  /* Dynamic entry type */
 union
   {
     Elf32_Word d_val;                 /* Integer value */
     Elf32_Addr d_ptr;                 /* Address value */
   } d_un;
} Elf32_Dyn;
```

Le champ d_tag contient le type d'entrée (dans notre cas : DT_NEEDED), le champ val (ou ptr) contient la valeur du pointeur, ou d'index associé à l'entrée. Dans le cas d'une entrée DT_NEEDED, le champ d_val contient un offset en octets depuis le début de la section .dynstr, où se trouve la chaîne de caracteres du nom de la bibliothèque.

Nous pouvons remarquer que certaines entrées ne sont pas obligatoires pour que l'exécutable puisse etre lancé, comme par exemple les entrées de type DT_DEBUG (index 9). Si nous changeons d_tag pour y mettre DT_NEEDED, et que nous changeons d_val pour y mettre l'offset d'une chaîne de caractères valide en tant que nom de bibliothèque, nous pouvons donc rajouter une dépendance au binaire ELF ET_EXEC sans injection de données supplementaires, sans changer sa taille et sans relocation.

Il faut pour cela trouver un nom de bibliothèque valide pour le système, terminé par NUL. Par exemple, dans ls :

```
$ elfsh -f /bin/ls -X dynstr | grep so
.dynstr + 16  6C 69 62 72 74 2E 73 6F 2E 31 00 63 6C 6F 63 6B
                                                librt.so.1.clock
.dynstr + 48  69 6E 5F 75 73 65 64 00 6C 69 62 63 2E 73 6F 2E
                                                in_used.libc.so.
.dynstr + 176 72 6E 61 6C 00 71 73 6F 72 74 00 6D 65 6D 63 70
                                                rnal.qsort.memcp
.dynstr + 784 65 65 00 6D 62 73 69 6E 69 74 00 5F 5F 64 73 6F
                                                ee.mbsinit.__dso
$

$ cat > /tmp/newlib.c
function()
{
printf("my own fonction \n");
}
```

```
$ gcc -shared /tmp/newlib.c -o /lib/rt.so.1
$ elfsh

        Welcome to The ELF shell 0.5b5 .::.

        .::. This software is under the General Public License
        .::. Please visit http://www.gnu.org to know about Free Software

[ELFsh-0.5b5]$ load /bin/ls

  [*] New object /bin/ls loaded on Mon Apr 28 23:09:55 2003

[ELFsh-0.5b5]$ d DT_NEEDED|DT_DEBUG

[SHT_DYNAMIC]
[Object /bin/ls]

[00] Name of needed library            =>          librt.so.1 {DT_NEEDED}
[01] Name of needed library            =>           libc.so.6 {DT_NEEDED}
[09] Debugging entry (unknown)         =>          0x00000000 {DT_DEBUG}

[ELFsh-0.5b5]$ set 1.dynamic[9].tag DT_NEEDED

  [*] Field set succesfully

[ELFsh-0.5b5]$ set 1.dynamic[9].val 19

  [*] Field set succesfully

[ELFsh-0.5b5]$ save /tmp/ls.new

  [*] Object /tmp/ls.new saved successfully

[ELFsh-0.5b5]$ quit

  [*] Unloading object 1 (/bin/ls) *

        Good bye ! .::. The ELF shell 0.5b5
```

Nous avons crée un nouveau `ls`, en changeant une entrée facultative `DT_DEBUG` de la section `.dynamic`, en une entrée `DT_NEEDED`. L'exécutable reste valide, seule une dépendance a été ajoutée.

```
$ ldd /tmp/ls.new
librt.so.1 => /lib/librt.so.1 (0x40021000)
libc.so.6 => /lib/libc.so.6 (0x40033000)
rt.so.1 => /lib/rt.so.1 (0x40144000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40146000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
$ ./ls.new
```

```
AcroOlAAFj  ELFSH_DEBUG  ls.new  newlib.c
$ elfsh -f ls.new -d DT_NEEDED

[*] Object ls.new has been loaded (O_RDONLY)

[SHT_DYNAMIC]
[Object ls.new]

[00] Name of needed library            =>          librt.so.1 {DT_NEEDED}
[01] Name of needed library            =>           libc.so.6 {DT_NEEDED}
[09] Name of needed library            =>             rt.so.1 {DT_NEEDED}

[*] Object ls.new unloaded
```

Certains binaires ne possèdent pas d'entrées facultatives dans `.dynamic` (`DT_DEBUG` ou autres), il n'est donc pas toujours possible d'utiliser la section `.dynamic` et d'encapsuler le code excédentaire dans une bibliothèque.

De plus, le nombre limité d'entrées facultatives dans la section `.dynamic` ne permet pas une granularisation des zones résidentes pour le code et les données excédentaires. Est-il aisé de relinker des modules relogeables dans un objet `ET_EXEC` non-relogeable ? La réponse est oui.

4.6 Injection `ET_REL` dans `ET_EXEC`

Cette méthode est très efficace et portable. En effet, seule la fonction de relocation est dépendante de l'architecture, mais toutes les primitives de manipulation ELF sont communes à toutes les machines.

Notre algorithme est simple, il s'effectue en 2 passes. La première passe, qui est la plus complexe, et consiste à insérer des copies des sections mappées du module dans l'exécutable, est la suivante :

- Pour chaque section du module
  - Si la section est de type BSS (`SHT_NOBITS` dans `sh_type`)
    - Initialiser l'interface interne du BSS si besoin
    - Insérer une nouvelle zone dans la section `.bss` pour ce module
  - Sinon, si elle est allocatable (`SHF_ALLOC` dans `sh_flags`)
    - Si elle est writable (`SHF_WRITE` dans `sh_flags`) alors :
      - Insérer la section sous le `.bss`
      - Résoudre et insérer les symboles pointant sur la section
    - Sinon
      - Insérer la section au dessus de `.interp`
      - Résoudre et insérer les symboles pointant sur la section
  - Sinon passer à la section suivante.

La deuxième passe permet de reloger chaque section injectée en utilisant sa table de relocation dans l'objet relogeable `ET_REL`. Étant donné que l'exécutable lui-même n'a pas besoin d'etre relogé (puisque ses sections mappées ne sont pas deplacée dans l'espace d'adressage lors de l'insertion d'un ou plusieurs modules), cette implementation à l'avantage de ne pas générer de faux-positifs de relocation.

Voici l'algorithme de cette deuxieme passe :

- Pour chaque section du module `ET_REL`
  - Si la section est de type BSS (`SHT_NOBITS` dans `sh_type`)

    - Ne pas reloger
  - Sinon, si elle est allocatable (`SHF_ALLOC` dans `sh_flags`)
    - Trouver la section injectée correspondante dans l'objet `ET_EXEC`
    - Trouver la section de relocation correspondante dans l'objet `ET_REL`
    - Si toutes ces sections sont disponibles :
      - Reloger la section injectée à l'aide de :
        `La table de relocation (.rel.*) dans l'objet ET_REL`
        `La table des symboles (.symtab) dans l'objet ET_EXEC`
        `La table des symboles dynamiques (.dynsym) dans l'objet ET_EXEC`
    - Sinon passer à la section suivante.
  - Sinon passer à la section suivante.

Le fait d'utiliser toutes les tables des symboles permet de relinker à la fois en utilisant les objets du module et les objets du binaire. Ainsi, le code injecté dans le module peut faire appel aux fonctions et utiliser les variables du binaire original.

Voici la sortie d'un script ELFsh où la commande `reladd` (insertion d'un module relogeable) est utilisée :

```
~exec cat func.c

int glvar_testreloc = 42;

int func(char *str)
{
  if (!str)
    glvar_testreloc++;
  printf("ET_REL takes control now [%s:%u]\n", str, glvar_testreloc);
  return (0);
}

 [*] Command executed successfully

~exec gcc -c func.c

 [*] Command executed successfully

~exec cat exec.c

int main()
{
  printf("First_printf\n");
  puts("Second_puts");
  return (0);
}

[*] Command executed successfully

~exec gcc exec.c

[*] Command executed successfully
```

```
~exec /tmp/a.out
First_printf
Second_puts

[*] Command executed successfully

~load a.out

[*] New object a.out loaded on Sun Apr  6 00:51:11 2003

s
 [SECTION HEADER TABLE .::. SHT is not stripped]
 [Object a.out]

[000] (nil)      -------                    foffset:00000000 size:00000244
[001] 0x80480f4  a------ .interp            foffset:00000244 size:00000019
[002] 0x8048108  a------ .note.ABI-tag      foffset:00000264 size:00000032
[003] 0x8048128  a------ .hash              foffset:00000296 size:00000052
[004] 0x804815c  a------ .dynsym            foffset:00000348 size:00000128
[005] 0x80481dc  a------ .dynstr            foffset:00000476 size:00000127
[006] 0x804825c  a------ .gnu.version       foffset:00000604 size:00000016
[007] 0x804826c  a------ .gnu.version_r     foffset:00000620 size:00000032
[008] 0x804828c  a------ .rel.dyn           foffset:00000652 size:00000008
[009] 0x8048294  a------ .rel.plt           foffset:00000660 size:00000040
[010] 0x80482bc  a-x---- .init              foffset:00000700 size:00000037
[011] 0x80482e4  a-x---- .plt               foffset:00000740 size:00000096
[012] 0x8048350  a-x---- .text              foffset:00000848 size:00000332
[013] 0x804849c  a-x---- .fini              foffset:00001180 size:00000028
[014] 0x80484b8  a------ .rodata            foffset:00001208 size:00000034
[015] 0x80494dc  aw----- .data              foffset:00001244 size:00000012
[016] 0x80494e8  aw----- .eh_frame          foffset:00001256 size:00000004
[017] 0x80494ec  aw----- .dynamic           foffset:00001260 size:00000200
[018] 0x80495b4  aw----- .ctors             foffset:00001460 size:00000008
[019] 0x80495bc  aw----- .dtors             foffset:00001468 size:00000008
[020] 0x80495c4  aw----- .got               foffset:00001476 size:00000036
[021] 0x80495e8  aw----- .bss               foffset:00001512 size:00000024
[022] (nil)      ------- .stab              foffset:00001512 size:00001932
[023] (nil)      ------- .stabstr           foffset:00003444 size:00006377
[024] (nil)      ------- .comment           foffset:00009821 size:00000228
[025] (nil)      ------- .note              foffset:00010049 size:00000120
[026] (nil)      ------- .shstrtab          foffset:00010169 size:00000222
[027] (nil)      ------- .symtab            foffset:00011552 size:00001248
[028] (nil)      ------- .strtab            foffset:00012800 size:00000551

~load func.o
 [*] New object func.o loaded on Sun Apr  6 00:51:11 2003

~s
 [SECTION HEADER TABLE .::. SHT is not stripped]
```

```
 [Object func.o]

[000] (nil)      -------                    foffset:00000000 size:00000064
[001] (nil)      a-x---- .text              foffset:00000064 size:00000050
[002] (nil)      aw----- .data              foffset:00000116 size:00000004
[003] (nil)      aw----- .bss               foffset:00000120 size:00000000
[004] (nil)      ------- .note              foffset:00000120 size:00000020
[005] (nil)      a------ .rodata            foffset:00000160 size:00000050
[006] (nil)      ------- .comment           foffset:00000210 size:00000038
[007] (nil)      ------- .shstrtab          foffset:00000248 size:00000071
[008] (nil)      ------- .symtab            foffset:00000760 size:00000192
[009] (nil)      ------- .strtab            foffset:00000952 size:00000051
[010] (nil)      ------- .rel.text          foffset:00001004 size:00000032

~relinject 1 2
 [*] ET_REL func.o injected succesfully in ET\_EXEC a.out

~switch 1
 [*] Switched on object 1 (a.out)

~s
 [SECTION HEADER TABLE .::. SHT is not stripped]
 [Object a.out]

[000] (nil)      -------                    foffset:00000000 size:00000244
[001] 0x80460f4  a------ func.o.rodata      foffset:00000244 size:00004096
[002] 0x80470f4  a-x---- func.o.text        foffset:00004340 size:00004096
[003] 0x80480f4  a------ .interp            foffset:00008436 size:00000019
[004] 0x8048108  a------ .note.ABI-tag      foffset:00008456 size:00000032
[005] 0x8048128  a------ .hash              foffset:00008488 size:00000052
[006] 0x804815c  a------ .dynsym            foffset:00008540 size:00000128
[007] 0x80481dc  a------ .dynstr            foffset:00008668 size:00000127
[008] 0x804825c  a------ .gnu.version       foffset:00008796 size:00000016
[009] 0x804826c  a------ .gnu.version_r     foffset:00008812 size:00000032
[010] 0x804828c  a------ .rel.dyn           foffset:00008844 size:00000008
[011] 0x8048294  a------ .rel.plt           foffset:00008852 size:00000040
[012] 0x80482bc  a-x---- .init              foffset:00008892 size:00000037
[013] 0x80482e4  a-x---- .plt               foffset:00008932 size:00000096
[014] 0x8048350  a-x---- .text              foffset:00009040 size:00000332
[015] 0x804849c  a-x---- .fini              foffset:00009372 size:00000028
[016] 0x80484b8  a------ .rodata            foffset:00009400 size:00000034
[017] 0x80494dc  aw----- .data              foffset:00009436 size:00000012
[018] 0x80494e8  aw----- .eh_frame          foffset:00009448 size:00000004
[019] 0x80494ec  aw----- .dynamic           foffset:00009452 size:00000200
[020] 0x80495b4  aw----- .ctors             foffset:00009652 size:00000008
[021] 0x80495bc  aw----- .dtors             foffset:00009660 size:00000008
[022] 0x80495c4  aw----- .got               foffset:00009668 size:00000036
[023] 0x80495e8  aw----- .bss               foffset:00009704 size:00000024
[024] 0x8049600  aw----- func.o.data        foffset:00009728 size:00000004
[025] (nil)      ------- .stab              foffset:00009732 size:00001932
```

```
[026] (nil)      ------- .stabstr           foffset:00011664 size:00006377
[027] (nil)      ------- .comment           foffset:00018041 size:00000228
[028] (nil)      ------- .note              foffset:00018269 size:00000120
[029] (nil)      ------- .shstrtab          foffset:00018389 size:00000260
[030] (nil)      ------- .symtab            foffset:00019892 size:00001248
[031] (nil)      ------- .strtab            foffset:00021140 size:00000551

~got puts
[Global Offset Table .::. GOT]
[Object a.out]

[004] 0x0804830A  <puts@@GLIBC_2.0 + 6>

~set 1.got[4] 1.sht[2].addr
 [*] Field set succesfully

~save /tmp/a.out.injected
 [*] Object /tmp/a.out.injected save successfully

~exec /tmp/a.out.injected
First_printf
ET_REL takes control now [Second_puts:42]

 [*] Command executed successfully
```

Comme vous pouvez le voir, la fonction `puts()` est detournée vers le module inséré, ce qui permet de conclure que la résidence par injection `ET_REL` a réussi. Puisque la SHT est synchronisée avec l'image mémoire décrite par la PHT, il est aisé de retirer les modules ainsi inserés, puisque la SHT donne la provenance de chaque section par son nom. De même, cette implémentation fonctionne sur les systèmes protegés par PAX, puisque toutes les injections de code étendent directement la zone exécutable du binaire, décrite par l'entrée de la PHT dont les droits sont `r-x`.

5 ASLR

5.1 Problématique

Pour contrer les protections non-exécutables, de nouvelles techniques comme le *return into libc* ont été inventées [10]. Cette attaque consiste à modifier le pointeur de pile, et le déplacer sur une séquence d'adresses de procedures exécutables, de manière à simuler des frames de fonctions dont le code est déjà présent dans le processus. Pour cela, les adresses des fonctions et de certains de leurs paramètres sont necessaires, et c'est au moment du `ret` de la fonction vulnérable que la séquence de fonctions est executée. Ainsi il n'est plus nécessaire d'injecter du code dans le processus, et les protections non-executable sont contournées.

La défense contre cette attaque est simple et efficace, puisqu'elle consiste à générer un aléa sur toutes les adresses de base des bibliothèques en modifiant le comportement de `mmap`. Cependant, certaines zones du processus restent mappées à des adresses fixes, comme les segments `PT_LOAD` du binaire, et certaines sections de ces segments se sont

avérées particulièrement intéressantes [3] à la génération de séquences de frames. Ainsi les fonctions, plus généralement les groupes d'instructions, dans les sections `.plt` ou `.text` peuvent être utilisés.

Il est complexe de stopper cette attaque, car un grand nombre d'adresses absolues sont codées en dur dans un binaire, par exemple dans son entête, ses sections de code (`.text`, `.plt`), ses sections de donnees (`.got`, `.data`, `.rodata`), sa section dynamique (`.dynamic`), sa section de symboles dynamiques (`.dynsym`) et autres sections de constructeurs et destructeurs. De plus, les sections de relocation pour un objet binaire exécutable de type `ET_EXEC` ont été supprimées par l'éditeur de liens, il est toutefois possible de les conserver avec `ld` par l'option `--emit-relocs`, mais ce n'est pas le cas pour les binaires installés par la grande majorité des distributions.

Grace au projet PAX [4], il existe une protection contre ces attaques. Elle consiste à mapper 2 fois le binaire dans le processus, l'un aux adresses du binaire, dont l'accès va être filtré par le mecanisme de page fault, et l'autre à une adresse aléatoire. Quand l'instance remappée du code accède aux segments d'origine par ses instructions d'adressage absolue, la référence est patchee dynamiquement depuis le handler de faute de page, ainsi l'exécution peut se dérouler malgré une chute de la performance. Cette protection n'est pas parfaite [13] mais semble très difficile à contourner en pratique par buffer overflow.

Une protection ASLR statique a pour but de patcher l'exécutable binaire de manière à le mapper à un autre endroit qu'aux adresses pour lesquelles il a été compilé. Pour cela, il est nécessaire de reconstruire une partie des informations de relocations du binaire, afin de savoir ou les références absolues doivent être mises à jour lors du déplacement des segments. Une premiere approche naïve consiste à lire toutes les sections mappées, et vérifier pour chaque mot de la taille du pointeur, si sa valeur correspond à une adresse mappée du binaire. Dans ce cas, nous devons créer une entrée de relocation pour ce pointeur. Meme si cette methode n'est pas fiable, et nous allons voir pourquoi dans les prochains paragraphes, elle a le mérite de fonctionner sur `/bin/ls` :

```
~quiet
~load /bin/ls
~modload ../../modules/modremap.so
~e

[ELF HEADER]
[Object /bin/ls, MAGIC 0x464C457F]

Architecture        :       Intel 80386  ELF Version       :           1
Object type         : Executable object  SHT strtab index  :          26
Data encoding       :     Little endian  SHT foffset       :       44336
PHT foffset         :                52  SHT entries number:          28
PHT entries number  :                 6  SHT entry size    :          40
PHT entry size      :                32  ELF header size   :          52
Entry point         :         0x8049420  (.text)
{PAX FLAGS = 0x0}
PAX_PAGEEXEC        :          Disabled  PAX_EMULTRAMP     :Not emulated
PAX_MPROTECT        :        Restricted  PAX_RANDMMAP      :  Randomized
PAX_RANDEXEC        :    Not randomized  PAX_SEGMEXEC      :     Enabled

~s
```

```
[SECTION HEADER TABLE .:::. SHT is not stripped]
[Object /bin/ls]

[00] (nil)      ---                 foffset(00000000) size(00000244)
[01] 0x80480f4  a-- .interp         foffset(00000244) size(00000019)
[02] 0x8048108  a-- .note.ABI-tag   foffset(00000264) size(00000032)
[03] 0x8048128  a-- .hash           foffset(00000296) size(00000600)
[04] 0x8048380  a-- .dynsym         foffset(00000896) size(00001296)
[05] 0x8048890  a-- .dynstr         foffset(00002192) size(00000875)
[06] 0x8048bfc  a-- .gnu.version    foffset(00003068) size(00000162)
[07] 0x8048ca0  a-- .gnu.version_r  foffset(00003232) size(00000128)
[08] 0x8048d20  a-- .rel.got        foffset(00003360) size(00000016)
[09] 0x8048d30  a-- .rel.bss        foffset(00003376) size(00000040)
[10] 0x8048d58  a-- .rel.plt        foffset(00003416) size(00000560)
[11] 0x8048f88  a-x .init           foffset(00003976) size(00000037)
[12] 0x8048fb0  a-x .plt            foffset(00004016) size(00001136)
[13] 0x8049420  a-x .text           foffset(00005152) size(00024636)
[14] 0x804f45c  a-x .fini           foffset(00029788) size(00000028)
[15] 0x804f480  a-- .rodata         foffset(00029824) size(00012092)
[16] 0x80533bc  aw- .data           foffset(00041916) size(00000188)
[17] 0x8053478  aw- .eh_frame       foffset(00042104) size(00000004)
[18] 0x805347c  aw- .ctors          foffset(00042108) size(00000008)
[19] 0x8053484  aw- .dtors          foffset(00042116) size(00000008)
[20] 0x805348c  aw- .got            foffset(00042124) size(00000300)
[21] 0x80535b8  aw- .dynamic        foffset(00042424) size(00000168)
[22] 0x8053660  -w- .sbss           foffset(00042592) size(00000000)
[23] 0x8053660  aw- .bss            foffset(00042592) size(00000680)
[24] (nil)      --- .comment        foffset(00042592) size(00000988)
[25] (nil)      --- .note           foffset(00043580) size(00000520)
[26] (nil)      --- .shstrtab       foffset(00044100) size(00000235)
[27] (nil)      --- .symtab         foffset(00044335) size(00000000)

~p
[Program header table .:::. PHT]
[Object /bin/ls]

[00] 0x08048034 -> 0x080480F4 r-x memsz(000192) foff(000000052)
                                                filesz(000192)
[01] 0x080480F4 -> 0x08048107 r-- memsz(000019) foff(000000244)
                                                filesz(000019)
[02] 0x08048000 -> 0x080523BC r-x memsz(041916) foff(000000000)
                                                filesz(041916)
[03] 0x080533BC -> 0x08053908 rw- memsz(001356) foff(000041916)
                                                filesz(000676)
[04] 0x080535B8 -> 0x08053660 rw- memsz(000168) foff(000042424)
                                                filesz(000168)
[05] 0x08048108 -> 0x08048128 r-- memsz(000032) foff(000000264)
                                                filesz(000032)

~findrel
```

```
[*] EXTRA relocs for /bin/ls
[*] Total number of absolute references : 1606
[*] Section                       srcref[0] dstref[0]
[*] Section .interp               srcref[0] dstref[0]
[*] Section .note.ABI-tag         srcref[0] dstref[0]
[*] Section .hash                 srcref[0] dstref[0]
[*] Section .dynsym               srcref[0] dstref[0]
[*] Section .dynstr               srcref[0] dstref[0]
[*] Section .gnu.version          srcref[0] dstref[0]
[*] Section .gnu.version_r        srcref[0] dstref[0]
[*] Section .rel.got              srcref[0] dstref[0]
[*] Section .rel.bss              srcref[0] dstref[0]
[*] Section .rel.plt              srcref[0] dstref[0]
[*] Section .init                 srcref[0] dstref[1]
[*] Section .plt                  srcref[72] dstref[6]
[*] Section .text                 srcref[705] dstref[741]
[*] Section .fini                 srcref[0] dstref[0]
[*] Section .rodata               srcref[812] dstref[101]
[*] Section .data                 srcref[17] dstref[1]
[*] Section .eh_frame             srcref[0] dstref[0]
[*] Section .ctors                srcref[0] dstref[2]
[*] Section .dtors                srcref[0] dstref[1]
[*] Section .got                  srcref[0] dstref[72]
[*] Section .dynamic              srcref[0] dstref[0]
[*] Section .sbss                 srcref[0] dstref[0]
[*] Section .bss                  srcref[0] dstref[444]
[*] Section .comment              srcref[0] dstref[0]
[*] Section .note                 srcref[0] dstref[0]
[*] Section .shstrtab             srcref[0] dstref[0]
[*] Section .symtab               srcref[0] dstref[0]

~remap 0x11223344
[*] Base address adapted to be congruent pagesize
[*] Delta is 091DB000
[*] MODREMAP : Section .sbss wont be relocated
[*] MODREMAP : Section .bss wont be relocated
[*] MODREMAP : Section .symtab wont be relocated
[*] Remapping at base 11223000 -OK-
[*] Total number of modified references : 1878
        PHT relocation : 12
        SHT relocation : 23
        ENT relocation : 1
        RAW relocation : 1842

~e

[ELF HEADER]
[Object /bin/ls, MAGIC 0x464C457F]

Architecture      :      Intel 80386 ELF Version       :           1
```

```
Object type       :Executable object SHT strtab index  :          26
Data encoding     :    Little endian SHT foffset       :       44336
PHT foffset       :               52 SHT entries number:          28
PHT entries number:                6 SHT entry size    :          40
PHT entry size    :               32 ELF header size   :          52
Entry point       :       0x11224420 (.text)
{PAX FLAGS = 0x0}
PAX_PAGEEXEC      :         Disabled PAX_EMULTRAMP     :Not emulated
PAX_MPROTECT      :       Restricted PAX_RANDMMAP      :  Randomized
PAX_RANDEXEC      :   Not randomized PAX_SEGMEXEC      :     Enabled

~s
[SECTION HEADER TABLE .::. SHT is not stripped]
[Object /bin/ls]

[00] (nil)      ---                 foffset(00000000) size(00000244)
[01] 0x112230f4 a-- .interp         foffset(00000244) size(00000019)
[02] 0x11223108 a-- .note.ABI-tag   foffset(00000264) size(00000032)
[03] 0x11223128 a-- .hash           foffset(00000296) size(00000600)
[04] 0x11223380 a-- .dynsym         foffset(00000896) size(00001296)
[05] 0x11223890 a-- .dynstr         foffset(00002192) size(00000875)
[06] 0x11223bfc a-- .gnu.version    foffset(00003068) size(00000162)
[07] 0x11223ca0 a-- .gnu.version_r  foffset(00003232) size(00000128)
[08] 0x11223d20 a-- .rel.got        foffset(00003360) size(00000016)
[09] 0x11223d30 a-- .rel.bss        foffset(00003376) size(00000040)
[10] 0x11223d58 a-- .rel.plt        foffset(00003416) size(00000560)
[11] 0x11223f88 a-x .init           foffset(00003976) size(00000037)
[12] 0x11223fb0 a-x .plt            foffset(00004016) size(00001136)
[13] 0x11224420 a-x .text           foffset(00005152) size(00024636)
[14] 0x1122a45c a-x .fini           foffset(00029788) size(00000028)
[15] 0x1122a480 a-- .rodata         foffset(00029824) size(00012092)
[16] 0x1122e3bc aw- .data           foffset(00041916) size(00000188)
[17] 0x1122e478 aw- .eh_frame       foffset(00042104) size(00000004)
[18] 0x1122e47c aw- .ctors          foffset(00042108) size(00000008)
[19] 0x1122e484 aw- .dtors          foffset(00042116) size(00000008)
[20] 0x1122e48c aw- .got            foffset(00042124) size(00000300)
[21] 0x1122e5b8 aw- .dynamic        foffset(00042424) size(00000168)
[22] 0x1122e660 -w- .sbss           foffset(00042592) size(00000000)
[23] 0x1122e660 aw- .bss            foffset(00042592) size(00000680)
[24] (nil)      --- .comment        foffset(00042592) size(00000988)
[25] (nil)      --- .note           foffset(00043580) size(00000520)
[26] (nil)      --- .shstrtab       foffset(00044100) size(00000235)
[27] (nil)      --- .symtab         foffset(00044335) size(00000000)

~p
[Program header table .::. PHT]
[Object /bin/ls]

[00] 0x11223034 r-x memsz(0000000192) foff(0000000052) filesz(0000000192)
[01] 0x112230F4 r-- memsz(0000000019) foff(0000000244) filesz(0000000019)
```

```
[02] 0x11223000 r-x memsz(0000041916) foff(0000000000) filesz(0000041916)
[03] 0x1122E3BC rw- memsz(0000001356) foff(0000041916) filesz(0000000676)
[04] 0x1122E5B8 rw- memsz(0000000168) foff(0000042424) filesz(0000000168)
[05] 0x11223108 r-- memsz(0000000032) foff(0000000264) filesz(0000000032)

~save /tmp/ls.remapped
[*] Object /tmp/ls.remapped save successfully

~exec /tmp/ls.remapped

CVS
graph-tests.esh
regression-tests.esh
regression-tests.out
remap_script.esh
remap_script.esh~
remap_script.out
strings_tests.esh
strings_tests.out

[*] Command executed successfully
```

Ce n'est malheureusement pas si facile pour tous les binaires, et les gros exécutables sont en général mal relogés par cette methode. Nous allons nous pencher sur les problèmes de mauvaises entrées de relocation, aussi appelés faux positifs.

5.2 Faux positifs et heuristiques de détection

Voyons le résultat de la méthode naïve sur un binaire tel que `ssh` :

```
$ /tmp/ssh.remapped
/tmp/ssh.remapped: error while loading shared librairies :
                                              unexpected PLT reloc
type 0x24
$
```

Le code de reconstruction de table de relocation a mal interpreté certains mots, et les a relogés alors qu'ils n'étaient pas des références absolues. Il est en effet probable que certaines séquences d'octets puissent, par coïncidence, représenter un pointeur valide sans en être un, comme ce fut le cas cette fois pour 4 octets dans une des entrées de relocation de la table *.rel.plt*.

Pour prévenir une telle confusion, nous devons associer à chaque type de section sa propre fonction de reconstruction. Ainsi, selon le type (`sh_type`) de la section, nous utiliserons une fonction différente, avec un comportement approprié :

1. Analyser entrée par entrée dans les sections `.rel*` et `.dynamic`
2. Analyser mot par mot dans `.got`, `.ctors`, `.dtors` et `.hash`
3. Analyser symbole après symbole dans `.symtab` et `.dynsym`
4. Analyser instruction par instruction dans les sections exécutables

5. Analyser de manière naïve pour les autres sections (`.data`, `.rodata`)

Pour chaque section de relocation, nous nous contenterons d'examiner le champ `r_offset`, pour chaque symbole nous utiliserons le champ `st_value` et pour les entrées de `dynamic` le champ `d_ptr`.

Voyons maintenant si `ssh` peut etre remappé :

```
$ /tmp/ssh.remapped mayhem@segfault.net
Enter passphrase for key '/home/mayhem/.ssh/id_dsa': Segmentation fault
$
```

L'erreur intervient avant la première frappe au clavier, voyons pourquoi :

```
$ gdb -q /tmp/ssh.remapped --core=core
(no debugging symbols found)...Core was generated by `/tmp/ssh.remapped
[...]
#0  0x11237e0b in EVP_get_digestbyname ()
(gdb) bt
#0  0x11237e0b in EVP_get_digestbyname ()
#1  0x00000000 in ?? ()
(gdb) x/1i $eip
0x11237e0b <EVP_get_digestbyname+71943>:        testb  $0x11,0x25b7df0c
(gdb)
```

L'opérande de cette instruction semble etre un faux positif, voyons les logs de notre reconstructeur de relocation pour cette adresse virtuelle :

```
$ elfsh -f /tmp/ssh.remapped -D text | grep 11237E0B

11237E0B [foff: 85515] .text + 71915
                          test $11,25B7DF0C F6 05 0C DF B7 25 11

$ elfsh -f /tmp/ssh.remapped -findrel | grep text | grep 11237E0

[1887] From .text + 71918 TO .bss  + 25503 [ 11237E0E -> 1125B7DF ]
```

Nous avons creé une entrée de relocation en croyant que les 4 derniers octets de cette instruction formaient un pointeur. On confirme cette erreur manuellement puisqu'on peut voir que le fameux pointeur appartient à deux opérandes distinctes de l'instruction. Nous devons nous assurer que chaque entrée de relocation reconstruite pointe sur une — et une seule — opérande lors de l'analyse des sections exécutables.

Malgre cette vérification supplémentaire, il existe toujours certains cas particuliers qui génèrent des fausses entrées de relocation :

```
movl $immed, %reg  ; put $immed in %reg
```

Dans cette construction, et dans le cas ou `$immed` représente une adresse valide du binaire, le comportement par défaut est de reconstruire une entrée de relocation qui passe le test d'alignement d'opérande. Toutefois, `$immed` n'est pas toujours une adresse. La valeur peut être tout simplement une valeur immediate qui, par coincidence, pointe dans l'espace d'adressage.

Afin de detecter ces construction particulières, il nous faut tracer le flux de données, c'est-à-dire l'utilisation du registre `%reg`. Nous devons pour cela typer chaque instruction, dans le but de determiner si la valeur immediate detectee est utilisee comme un

pointeur (un registre de base par exemple) ou une valeur immediate (comme un mot dans du code de hashage par exemple, pour reprendre le cas ssh).

Les faux négatifs sont également un problème, par exemple lors de la construction d'adresse au moment de l'exécution. Cette technique est utilisée par les virus, dans le but de cacher leurs références, et ainsi rendre la tâche difficile aux programmes d'analyse automatique. Elle consiste à construire les adresses par des opérations arithmétiques, dans le but de ne pas la coder en dur, et ainsi échapper à la détection. ELFsh ne dispose pas d'un moteur fiable d'analyse de flux, la protection statique ASLR reste donc un problème ouvert.

5.3 Remappage ET_EXEC

L'algorithme de remapping d'objets `ET_EXEC` est indépendant de l'algorithme de relocation, meme s'il en dépend. Après avoir mis à jour le point d'entrée dans l'entête ELF, nous devons parcourir la PHT et la SHT afin de déplacer chaque segment et chaque section mappée dans l'espace d'adressage virtuel.

- Calculer la différence entre les anciennes et nouvelles adresses de bases
- Ajouter la différence à `e_entry` dans le header ELF
- Pour chaque segment mappé :
  - Ajouter la différence à `p_vaddr`
  - Ajouter la différence à `p_paddr`
- Pour chaque section mappée :
  - Ajouter la différence à `sh_addr`

Toutefois, cette méthode est limitée par le choix de la nouvelle adresse de base, qui sera maintenant toujours utilisée, à moins que l'on remappe plusieurs fois le meme objet. Pour éviter le multiple remapping, il est intéressant de transformer un objet exécutable `ET_EXEC` en une bibliothèque partagée de type `ET_DYN`. Ainsi nous utiliserons le mécanisme de randomisation dynamique de `mmap()` pour générer notre aléa au cours de chaque exécution.

5.4 Relinkage ET_EXEC en ET_DYN

Les différences majeures entre un objet `ET_EXEC` et un objet `ET_DYN` sont :

L'adressage relatif : Afin de transformer un adressage absolu en adressage relatif, il suffit de soustraire l'adresse basse du binaire à chaque référence. Il faut alors synchroniser la table des symboles et la tables des symboles dynamiques avec le nouvel espace d'adressage.

La relocation au moment du mapping : Contrairement aux exécutables, les bibliothèques doivent être rélogées une première fois avant d'être exécutée, notamment en relogeant toutes les références de type relatif :

```
#define R_386_RELATIVE  8               /* Adjust by program base */
#define R_SPARC_RELATIVE 22             /* Adjust by program base */
```

Cela veut dire que notre bibliothèque va comporter une grand nombre d'entrées de relocation supplémentaires (une pour chaque référence que nous avons passée en relative dans le binaire), et donc qu'il faut étendre les sections de relocation. L'extension d'une section `.rel` ne pose pas de probleme. Cela dit, on distingue un cas particulier : la `.plt`.

Le format de la `.plt` est différent pour les objets de type `ET_DYN` car elle admet une forme PIC[3], dont l'adressage est entierement basé sur la valeur du registre `EBX` (sous INTEL) :

```
0001B160 [foff: 110944] .plt + 0                push          4(%ebx)
0001B166 [foff: 110950] .plt + 6                jmp           *8(%ebx)
0001B16C [foff: 110956] .plt + 12               add           %al,(%eax)
0001B16E [foff: 110958] .plt + 14               add           %al,(%eax)

0001B170 [foff: 110960] .plt + 16               jmp           *C(%ebx)
0001B176 [foff: 110966] .plt + 22               push          $0
0001B17B [foff: 110971] .plt + 27               jmp           <.plt>

0001B180 [foff: 110976] .plt + 32               jmp           *10(%ebx)
0001B186 [foff: 110982] .plt + 38               push          $8
0001B18B [foff: 110987] .plt + 43               jmp           <.plt>

0001B190 [foff: 110992] .plt + 48               jmp           *14(%ebx)
0001B196 [foff: 110998] .plt + 54               push          $10
0001B19B [foff: 111003] .plt + 59               jmp           <.plt>
```

Cette spécificité est importante pour la translation `ET_EXEC` en `ET_DYN`, car il n'est pas envisageable de conserver le registre `%ebx` à la bonne valeur (l'adresse du début de la `.got`) durant toute l'exécution du processus. Ainsi le format de la `.plt` d'un objet binaire n'est pas exécutable tel quel si l'executable est relinké en bibliothèque.

La solution consiste à ajouter des entrées de relocation qui pointent dans la `.plt` du binaire, qui fait plusieurs référence absolues, comme décrit dans la section d'interception de flux de contrôle dynamique (section 3.2. Comme la `.plt` est une section de code qui n'est en principe pas modifiée au cours de l'exécution, il faut préciser un paramètre de comportement à l'éditeur de liens dynamique, afin qu'il passe les pages de code en écriture lors de la relocation de la section. Ce comportement est spécifié par une entrée dans la section dynamique, l'entrée `DT_TEXTREL` :

```
#define DT_TEXTREL      22              /* Reloc might modify .text */
```

L'éditeur de lien dynamique vérifie la valeur de ce paramètre dans l'instance mappée de la section `.dynamic` (dans la structure linkmap de cet objet, dont l'adresse se trouve dans la deuxième entrée de la `.got`) Si ce flag est présent et q'il est amené à reloger une section en lecture seule, il effectue un appel à `mprotect(2)` dans le but de changer les droits des pages de cette section, à ce moment il effectue la relocation, et enfin restaure les anciens droits en lecture seule.

Cette fonctionnalité n'est pas implementée car elle est bridée par le moteur limité de traçage de flux dans libasm. Tout comme le remapping `ET_EXEC`, l'algorithme de relinkage `ET_EXEC` en `ET_DYN` est indépendant de l'algorithme de reconstruction des entrées de relocation, mais il en depend.

6 Le modèle ELFsh

Il est orienté sur deux axes :

[3] *Position Independant Code*

– le croisement des espaces de noms ;
– la virtualisation des composants logiciels et des objets ELF.

Les fonctionnalités d'ELFsh peuvent etre appelées depuis le shell, dans une session interactive, par un script shell, ou par un script ELFsh. L'idée d'une interface GUILE [12] ou PERL a été discutée, mais ne reste qu'au stade de projet pour le moment.

D'un autre côté, chaque composant du shell — notamment les commandes — et chaque structure ELF sont virtualisés en objets, qui possèdent leurs propres routines (méthodes) et attributs.

6.1 Orienté Objet

Il existe 3 types d'objets ELFsh.

Objet L1 Ce sont les objets de niveau 1 (Level 1), c'est-à-dire les objets parents.

```
/* ELFsh Level 1 object (= parent object) structure */
typedef struct s_L1handler
{
 hash_t *l2list;    /* A ptr on the child L2 hashtable */
 u_int elem_size;  /* Size of one element of this object */
   /* Handlers */
 void *(*get_obj)(void *file, void *arg); /* Read object */
   /* Read multiple instanced L1 obj */
 void *(*get_obj_idx)(void *file, u_int i, void *a);
 void *(*get_obj_nam)(void *file, char *name); /* Read by name */
 void *(*get_entptr)(void *file, u_int idx);  /* Get address */
 u_int (*get_entval)(void *ptr);  /* Get value */
 u_int (*set_entval)(void *ptr, u_int vaddr); /* Set value */
} elfshL1_t;
```

Un objet L1 est utilisé pour représenter :
– L'en-tete ELF (label : `hdr`)
– La PHT (label : `pht`)
– La table des symboles (label : `symtab`)
– La table des symboles dynamiques (label : `dynsym`)
– La section dynamique (label : `dynamic`)
– La section `.got` (label : `got`)
– La section `.ctors` (label : `ctors`)
– La section `.dtors` label : `dtors`)
– Les tables de relocation (label : `rel`)
– La liste des sections (label : `section`)

Objet L2 Ce sont les objets de niveau 2 (Level 2), c'est-à-dire les objets enfants des objets L1.

L'objet L2 est utilisé pour répresenter la liste des champs de structure de chaque entrée des objets ELF (sht, pht, relocations, symboles, entête ELF, etc). L'objet L1 section est différent, en ce sens que ses objets L2 sont abstraits et ne coïncident pas avec les champs de leur entête (pour cela, l'objet *sht* a été mis à disposition) mais permet d'indexer les données de la section. De même, les objets *GOT*, *CTORS* et *DTORS* n'admettent pas d'objets fils L2.

```
/* ELFsh Level 2 object (= L1 child) structure */
typedef struct s_L2handler
{
   /* For fields */
 int           (*get_obj)(void *obj); /* Read object */
 int           (*set_obj)(void *par, u_int arg); /* Write object */
   /* For names */
 char          *(*get_name)(elfshobj_t *, void *obj); /* Get name */
 int           (*set_name)(elfshobj_t *, void *, char *);/* Set name */
   /* For sections data */
 char *(*get_data)(elfshsect_t *, u_int off, u_int sizelem); /* Read data */
 int  (*set_data)(elfshsect_t *, u_int, char *, u_int, u_int);/* Write data */
 char type;     /* Object type */
} elfshL2_t;
```

Voici la liste des objets L2 pour chaque objet L1 :

hdr : magic class type machine version entry phoff shoff flags ehsize phentsize shentsize phnum shnum shstrndx pax_pageexec pax_emultramp pax_mprotect pax_randmmap pax_randexec pax_segmexec

sht : type offset addr size link info align entsize a w x s m l o

pht : type offset paddr vaddr filesz memsz flags align

symtab/dynsym : name value size bind type other

dynamic : val tag

section : name raw

rel : type sym offset

Objet abstrait PATH Cet objet permet d'avoir une represéntation unique d'un objet, quelque soit son type et sa profondeur. C'est un objet abstrait, aussi appelé méta-objet.

```
int  vm_lookup_param(char *path, elfshpath_t *pobj, u_int mode);
```

Cette fonction de l'API interne permet de résoudre un objet abstrait selon son chemin dans l'arborescence à deux niveaux des objets L1 et L2. Il permet également de représenter des valeurs immédiates.

```
/* Meta object */
typedef struct s_elfshpath
{
   /* Handlers */
 u_long             (*get_obj)(void *parent);
 u_long             (*set_obj)(void *parent, long value);
 char               *(*get_name)(elfshobj_t *, void *obj);
 int                (*set_name)(elfshobj_t *, void *, char *);
 char              *(*get_data)(elfshsect_t *, u_int off, u_int);
 int                (*set_data)(elfshsect_t *, u_int, char *, u_int, u_int);
 elfshobj_t         *root;             /* Root parent */
 void               *parent;         /* Direct parent */
```

```
 u_int              off;              /* Optional byte offset */
 u_int              size;             /* Size of the immediate string */
 u_int              sizelem;          /* Size of element for OBJRAW */
 char               immed;            /* Immediate binary flag */
   /* Immediate value of immed flag is set */
 union                immval {
   char               *str;
   long               ent;
 }                    immed_val;

   /* Here is the object type list */
 #define              ELFSH_OBJINT 0       /* Dword */
 #define              ELFSH_OBJSTR 1       /* String */
 #define              ELFSH_OBJRAW 2       /* Raw */
 #define              ELFSH_OBJUNK 3       /* Unknown */
 char type;
} elfshpath_t;
```

Toutes les commandes ELFsh utilisent les objets *PATH*, et n'interagissent aucunement avec les objets L1 et L2. Ceux-là sont uniquement connus du code de résolution de chemin interne, c'est-à-dire par la fonction `vm_lookup_param()`.

6.2 Modèle Ouvert

ELFsh dispose d'une interface de module simple, ainsi qu'une API interne de gestion de tables de hash et de gestion de commandes.

Modules Voici un module de base :

```
#include "elfsh.h"

#define CMD_TEST "cmdtest"

int my_cmd_print()
{
  puts("\n [*] I do block the print command, oh yeah . \n");
  return (0);
}

int my_new_cmd()
{
  puts("\n [*] This is a new command, oh no ! \n");
  return (0);
}

void elfsh_init()
{
  puts(" [*] ELFsh modtest init -OK- \n");
  vm_setcmd(CMD_PRINT, my_cmd_print, ELFSH_ORIG, (u_int) ELFSH_ORIG);
```

```
  vm_addcmd(CMD_TEST, my_new_cmd, NULL, 0);
}

void elfsh_fini()
{
  puts(" [*] ELFsh modtest fini -OK- \n");
  vm_setcmd(CMD_PRINT, cmd_print, ELFSH_ORIG, (u_int) ELFSH_ORIG);
  vm_delcmd(CMD_TEST);
}
```

Interface des commandes Il est possible de rajouter des commandes à la volée, et même de détourner des commandes existantes par cette voie. L'API interne se résume à des fonctions de manipulation d'objets abstraits comme ceux presentés précédemment, et des commandes du shell. Voici le log de la session ELFsh générée par l'utilisation de ce module :

```
[ELFsh-0.5b4]$ print 1 2 3 4

1 2 3 4

[ELFsh-0.5b4]$ cmdtest

 [!] Unknown command cmdtest .::. type 'help' for command list

[ELFsh-0.5b4]$ modload modules/modtest.so

 [*] ELFsh modtest init -OK-

[ELFsh-0.5b4]$ cmdtest

 [*] This is a new command, oh no !

[ELFsh-0.5b4]$ print toto 42

 [*] I do block the print command, oh yeah .

[ELFsh-0.5b4]$ modunload modules/modtest.so

 [*] ELFsh modtest fini -OK-

 [*] Module modules/modtest.so unloaded on Tue Feb 25 01:56:13 2003

[ELFsh-0.5b4]$ print 1 2 3 4

1 2 3 4

[ELFsh-0.5b4]$ cmdnew

 [!] Unknown command cmdnew .::. type 'help' for command list
```

```
[ELFsh-0.5b4]$
```

Tables de hash En interne, le shell indexe tous ces objets par des tables de hash, dont l'API est également disponible depuis les modules. Dans la version 0.5b6, il est nécessaire d'utiliser directement ces tables de hash pour rajouter des objets L1 et L2 à l'arborescence. Le lecteur est invité à consulter la référence de l'API interne sur la page du projet [1].

6.3 Portabilité

Toutes les fonctionnalités présentées dans ce exposé (injection de sections, détournement de fonctions, etc) sont compatibles INTEL et SPARC, excepté l'injection `ET_REL` sous SPARC qui est en cours de développement au moment de l'écriture de cet article (Avril 2003).

L'ASLR a été etudié sur machine INTEL uniquement, du fait des heuristiques de détection de faux positifs de relocation qui sont dépendantes de l'architecture et donc de *libasm*. Toutefois, les algorithmes de remapping présentés sont indépendants de l'architecture.

Le code est developpé principalement sous Linux, régulierement porté sous les machines NetBSD, FreeBSD et Solaris. Une version portable de la serie 0.5 devrait être publiée prochainement. Nous travaillons également au portage MIPS/IRIX et collaborons avec d'autres programmeurs afin de disposer d'une interface d'analyse à la *libasm* pour l'architecture SPARC.

Références

1. *The ELF shell.* `http://devhell.org/projects/elfsh/`
2. *GNU binutils.* `ftp://ftp.gnu.org/gnu/binutils/`
3. *Advanced ret-into-libc exploits.* `http://phrack.org/phrack/58/p58-0x04`
4. *The PAX project.* `http://pageexec.virtualave.net`
5. *ELF TIS reference.* `http://x86.ddj.com/ftp/manuals/tools/elf.pdf`
6. *IA32 advanced function hooking.* `http://phrack.org/phrack/58/p58-0x08`
7. *ELF runtime fixup.* `http://devhell.org/~mayhem/papers/elf-runtime-fixup.txt`
8. *ELF rtld internals.* `http://devhell.org/~mayhem/papers/elf-rtld.txt`
9. *ELF PLT Infection.* `http://phrack.org/phrack/56/p56-0x07`
10. *Getting around non-exec stack.* `http://www.securityfocus.com/archive/1/7480`
11. *x86 processor informations.* `http://www.sandpile.org/ia32/index.htm`
12. *The GNU extension langage.* `http://www.gnu.org/software/guile/guile.html`
13. *Bypassing PaX ASLR protection.* `http://www.phrack.org/phrack/59/p59-0x09.txt`

Atouts et limites du modèle de sécurité du pare-feu personnel

Cédric Blancher

Cartel Sécurité
blancher@cartel-securite.fr
http://www.cartel-securite.fr/

Résumé Le développement des accès haut-débit pour les particuliers a ouvert un nouveau marché en matière de sécurité. Les spécificités des accès "grand public" ont conduit à l'émergence d'un nouveau type de produit : le firewall personnel.

Si le concept est intéressant et si ce type outil apparaît aujourd'hui comme un composant incontournable de protection, il est essentiel d'en apprécier les atouts comme les limites. L'article vise donc à présenter ce qu'est un firewall personnel par ses concepts fondamentaux, puis en analyser les faiblesses dans son contexte de fonctionnement. Nous pourrons ainsi en définir les limites pour parvenir à une utilisation éclairée et efficace.

1 Introduction

Le firewall personnel, ou pare-feu personnel[1], vise à offrir aux connexions Internet dites personnelles, à savoir un poste directement relié à Internet, segment de marché jusqu'alors oublié des éditeurs, un logiciel de sécurité réseau simple et efficace.

En effet, jusqu'à l'apparition des premiers produits de ce type, les logiciels et autres équipements de sécurité réseau (i.e. les firewalls) visaient à protéger des réseaux entiers situés en amont. L'utilisation d'un tel dispositif pour protéger un poste isolé supposait la constitution d'un réseau local de manière à placer ce dernier en amont du firewall. La mise en place efficace de ce type de configuration demande d'une part des moyens matériels, donc financiers, et d'autre part des compétences certaines pour configurer convenablement le tout. Ce type d'infrastructure de sécurité n'est donc pas adaptée aux besoins de sécurité des particuliers, c'est-à-dire l'utilisateur moyen d'Internet à domicile. Il convenait donc de trouver autre chose.

2 Les bases du firewall personnel

2.1 Description du contexte

La spécificité des connexions monopostes est leur caractère essentiellement "client". Une telle connexion n'est en effet pas adaptée à une utilisation de type "serveur", ne serait-ce que parce que les adresses IP affectées par les fournisseurs d'accès sont souvent

[1] par soucis de simplicité, nous utiliserons le terme plus courant de "firewall"

dynamiques. Malgré tout, cette dimension doit être prise en compte, puisque cependant possible.

Ce qui caractérise une utilisation cliente, c'est le manque, voire l'absence, de services accessibles depuis Internet, donc de points d'entrées susceptibles d'être exploités par un pirate pour compromettre le poste. Il en résulte qu'en l'absence de points d'accès directs, la principale menace qui pèse sur notre utilisateur est la réception par des moyens divers (courrier électronique, téléchargements, etc.) de programmes type "chevaux de Troie" conduisant à la mise en place de logiciels intrus, commes des espions (spyware) ou des portes dérobées (backdoor) par exemple.

Le rôle d'un firewall dans ce contexte sera double. Il devra d'abord fermer tous les accès au poste protégé, évitant ainsi l'utilisation distante de services ouverts dans les installations par défaut. Ensuite, il devra empêcher l'envoi ou la réception de données via le réseau par des programmes non autorisés pouvant entraîner des fuites d'information ou la compromission du poste par un intrus qui pourra l'utiliser à des fins détournées. C'est cette dernière tâche qui sera la plus importante et la plus problématique.

2.2 Utilisation d'un système de filtrage de paquets classique

Cette tâche peut être réalisée avec un système de filtrage que l'on qualifiera de classique, c'est à dire s'appuyant uniquement sur les caractéristiques des flux réseau locaux. On va donc filtrer les paquets entrant et sortant sur la base des éléments protocolaires qu'ils contiennent. Il doit être noté que pour une station terminale, ce type de filtrage n'apporte rien par rapport à un filtrage effectué par un équipement tiers, puisqu'il ne s'appuie que sur les caractéristiques des flux observés.

Supposons un poste A relié à Internet. Si on veut restreindre son utilisation du réseau à une utilisation du Web, nous devons mettre en place ce type de règles :

```
Entrée :
   source :           inconnue
   destination :      IPa
   protocole :        TCP
   port source :      80
   port destination : > 1023
   drapeau TCP :      ! SYN

Sortie :
   source :           IPa
   destination :      inconnue
   protocole :        TCP
   port source :      > 1023
   port destination : 80
   drapeau TCP :      -
```

Pour peu que nous utilisions un firewall à états, nous pouvons simplifier notre jeu de règles à :

```
Sortie :
   source :           IPa
   destination :      inconnue
```

```
    protocole :        TCP
    port source :      > 1023
    port destination : 80
```

Le problème principal réside dans le filtrage de la plage des ports non privilégiés (1024-65535). En effet, ce sont les ports de cette plage qui sont utilisés par les applications clientes comme source, sans qu'on puisse dans ce contexte les distinguer entre eux. En effet, n'importe quel port de cette plage peut servir à une application donnée. À l'inverse, n'importe quelle application est susceptible d'utiliser un port donné de cette plage. Les choses se compliquent encore lorsqu'on doit prendre en compte des protocoles applicatifs comme FTP ou H323 qui incluent une négociation de ports pour l'établissement de connexions annexes. Si on considère l'exemple de FTP, dont le mode de transfert de données actif implique la connexion du serveur FTP sur le client, vers un port de la plage non privilégiée, on s'aperçoit que l'utilisation d'une telle application entraine l'autorisation de connexion vers n'importe quel port de cette plage, ce qui réduit considérablement l'intérêt de notre filtrage. En effet, si nous considérons l'utilisation du FTP par notre poste A, sur un firewall à état ne prenant pas en compte les spécificités de FTP, nous aurons un jeu de règles du type :

```
Entrée :
   source :           inconnue
   destination :      IPa
   protocole :        TCP
   port source :      20
   port destination : > 1023

Sortie :
   source :           IPa
   destination :      inconnue
   protocole :        TCP
   port source :      > 1023
   port destination : 21

   source :           IPa
   destination :      inconnue
   protocole :        TCP
   port source :      > 1023
   port destination : > 1023
```

Un tel jeu de règle autorise n'importe quelle machine distante à se connecter à un port non privilégié du poste A pourvu qu'elle utilise le port 20 comme source, et à n'importe quelle application locale de se connecter à un port non privilégié distant. Un filtrage à état prenant capable de gérer les spécificités du protocole FTP apporterait une solution efficace à ce problème. Malheureusement, les firewalls personnels du marché n'incluent pas ce type de fonctionnalité.

2.3 Filtrage par application

La conclusion de ces constatations est que le filtrage de paquets répond difficilement à notre problématique. À moins d'utiliser un système de filtrage à état (stateful) complet, il nous faudra trouver d'autres critères.

L'élément que nous pouvons examiner lorsqu'on filtre des flux locaux est l'application qui les génère ou les reçoit. En effet, si on part du principe qu'on est capable d'associer un comportement donné à une application donnée (i.e. un client FTP fait du FTP, un MUA fait du SMTP, du POP, de l'IMAP, etc.), on est non seulement capable de rendre plus efficace notre système de filtrage, mais aussi d'en simplifier grandement la configuration. Il est important pour la suite de bien noter que ce principe constitue la base du fonctionnement du firewall personnel.

En insérant des points de lecture au sein de la pile TCP/IP du système d'exploitation, nous pouvons savoir par quelle application est émis un paquet, ou à quelle application est destiné un paquet reçu. En établissant une liste des applications autorisées (ou non) soit à initier un flux, soit à se mettre en écoute, nous sommes en mesure d'apporter un élément de réponse probant à notre problème.

3 Le firewall personnel dans la pratique

Dans la pratique, le firewall personnel est un système dont le rôle principal sera de filtrer les demandes d'ouverture de socket par les applications du système. La plupart des produits entrant dans cette catégorie sont à destination des systèmes d'exploitation grand public, à savoir Microsoft Windows 95, 98, Me, NT, 2000 et XP ou encore MacOS d'Apple. Des produits ciblés pour les entreprises sont aussi proposés, sur les mêmes plateformes.

De tels outils ne sont pas proposés pour les Unix du marché. Certains de ceux-ci disposent de solution de filtrage de paquets performantes, comme Netfilter, IP Filter ou encore Packet Filter. Bien que n'entrant pas dans cette catégorie d'outils, nous considèrerons brièvement le cas de Netfilter, le filtre de paquets Linux, dans la mesure où il inclue des fonctionnalités de filtrage similaires. Mais l'essentiel de cet article portera sur les architectures Windows.

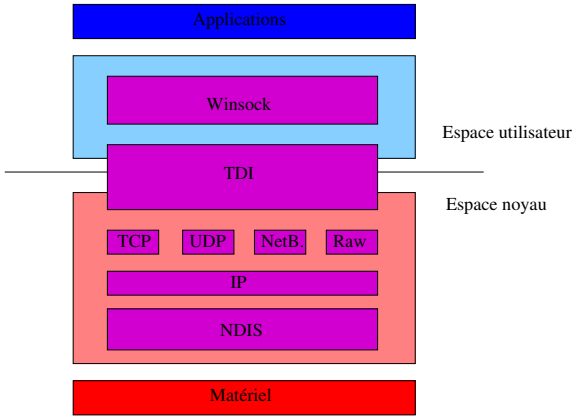
3.1 Caractéristiques principales

Un firewall personnel nécessite l'introduction dans la pile TCP/IP (cf. Tab. 1) de deux points d'entrées (hook).

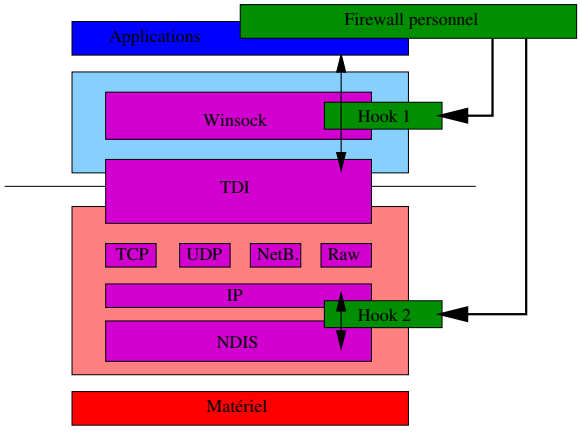
Le premier point se situe au niveau de l'API Winsock (cf. Tab. 2). Cette API sert d'interface entre l'espace utilisateur, donc les applications, et la couche TDI (Transport Driver Interface) qui sert d'interace d'accès aux protocoles de niveau 4 (typiquement TCP et UDP) en espace noyau. Ce point filtre les demandes d'ouverture de sockets de communications par les applications, quel que soit leur type (TCP, UDP ou RawIP sous Windows XP) et leur mode (client ou écoute).

Le second (Cf. Tab. 2) vient se positionner entre la couche IP et la couche NIDS (Network Driver Interface Specification) qui réalise l'interface unifiée pour les pilotes de périphériques de communication et qui intègre les protocoles de niveau 2 comme PPP, PPPoE ou encore PPTP. C'est le point de filtrage des paquets qui passent de la couche liaison à la couche IP, quel qu'en soit le sens.

La majorité des produits du marché gèrent les états, au moins pour TCP. Ils sont donc capables de savoir si un paquet donné appartient ou non à une connexion déjà établie. Certains ne savent pas gérer les états pour UDP. Dans la mesure ou aucun paramètre ne nous permet de discriminer le sens d'un flux UDP, cette absence est un manque lourd qui entraine des ouvertures importantes dans le jeu de règles. Aucun



Tab. 1. Pile réseau générique Windows



Tab. 2. Positionnements des hooks

ne possède de gestion efficace des erreurs ICMP. Si cela ne pose pas vraiment de problème de sécurité, cette mauvaise gestion conduit à des alertes et des refus de messages d'erreur valides que l'outil n'est pas capable d'associer au flux qui les a généré. Il s'en suit parfois des problèmes de connectivité, en particulier pour les découvertes de PMTU. Enfin, aucun ne gère de protocole applicatif faisant intervenir une négociation de flux annexe, comme FTP ou H323.

La méthode de configuration est sensiblement la même pour tous ces outils : il s'agit d'une méthode d'apprentissage. Chaque fois qu'une application demande à accéder au réseau, le firewall vérifie sa présence dans une liste de contrôle recensant les applications faisant l'objet d'une décision. S'il la connait, il applique la décision associée. S'il ne la connait pas, il demande à l'utilisateur ce qu'il doit faire. La nouvelle application est alors introduite dans la liste de contrôle. Cette approche présente deux avantages. D'une part, elle permet de simplifier grandement la tâche de configuration, rendant l'outil prêt à l'utilisation, sans aucune spécification initiale. D'autre part, ce mécanisme sert de système de remontée d'alerte : tout accès inconnu est notifié à l'utilisateur. Il en va souvent de même pour les paquets reçus. Chacun fait l'objet d'une notification jusqu'à ce qu'une règle, plus ou moins précise, soit établie par l'utilisateur pour le traiter.

Les applications sont reconnues par le chemin absolu de l'exécutable correspondant. On évite ainsi qu'un binaire portant le nom d'une application autorisée puisse initier des flux réseau. En outre, ce chemin est assorti d'une somme de contrôle, basée sur MD5, pour éviter le remplacement d'un exécutable valide par une version vérolée contenant du code malicieux.

On peut classer ces produits en deux catégories, selon les fonctionnalités de filtrage qu'ils offrent.

3.2 Produits basiques

Les produits basiques ne permettent que l'autorisation ou l'interdiction d'applications. Une fois une application autorisée, elle peut initier n'importe quel type de flux. Ce type d'outil est le plus simple à utiliser puisqu'il ne demande pratiquement aucune connaissance de la part de l'utilisateur. C'est cette facilité d'utilisation qui place souvent ce type d'outil en tête des comparatifs.

D'un point de vue purement technique, cette approche suppose que l'utilisateur connaisse, au moment où il autorise une application donnée, tous les types de flux qu'elle peut être amenée à générer ou recevoir. On retrouve là le principe de base évoqué plus haut. Comme nous le verrons plus loin, cette condition est loin d'être remplie, posant dès lors un potentiel problème de sécurité. En effet, il est des applications auxquelles nous ne pouvons pas faire aveuglément confiance. Nous verrons plus loin que Internet Explorer, par exemple, peut être utilisé, de manière légitime ou non, par des applications tierses pour initier des flux réseau. C'est une de ses nombreuses fonctionnalités. Un trojan pourrait tout à fait exploiter ce type d'interface pour passer outre les restrictions imposées par le firewall personnel en profitant de celles dont jouit ce navigateur.

3.3 Produits avancés

Les produits avancés permettent d'associer à une application le type de flux qu'elle est autorisée à initier ou recevoir. Cette approche est certes plus fine et efficace en ce qu'elle permet de contraindre le type d'utilisation d'une application donnée, mais rend

la configuration plus difficile en ce qu'elle réclame des connaissance techniques de la part de l'utilisateur.

Cependant, les applications versatiles ou mettant en œuvre des protocoles incluant des flux négociés demandent des autorisations extrêmement larges qui rendent ce type de restrictions inutiles en l'absence d'un filtrage à états complet comme vu plus haut. Considérons un navigateur. Un tel outil propose au minimum de support de HTTP et de FTP en mode passif. La seule autorisation de ce dernier protocole lui donne le droit de se connecter, en TCP, à tout port non privilégié de n'importe quel adresse (on ne connaît pas à priori les adresses de serveurs qu'on consulte) depuis n'importe quel port non privilégié local. Autant dire que ce navigateur peut faire ce qu'il veut sur le réseau. C'est un problème épineux si on considère qu'Internet Explorer tombe dans cette catégorie et qu'il peut être amené à servir de relai pour d'autres applications...

3.4 Linux/Netfilter

Netfilter [1], le système de filtrage des noyaux Linux, n'est pas un firewall personnel. Il inclue cependant une concordance, "owner", lui permettant d'intégrer des éléments de l'espace applicatifs dans ses règles :

- UID et/ou GID de l'utilisateur à qui appartient le processus générant le flux ;
- PID et/ou SID du processus générant le flux ;
- nom de la commande dont est issu le processus qui génére le flux.

Si la fonctionnalité permettant de spécifier le nom d'une commande parait intéressant, elle est malheureusement assez limitée. Le nom qui est utilisé par cette concordance est celui qui apparait dans la liste des processus. Or ce dernier peut être aisément modifié, par exemple en établissant tout simplement un lien symbolique portant le nom d'une commande autorisée vers un exécutable interdit. Il conviendrait donc de durcir cette fonctionnalité pour véritablement reconnaître un exécutable et non une commande.

Cependant, cette concordance offre des possibilités intéressantes en ce qu'elle permet de mettre en place des politiques de filtrage différentes selon les utilisateurs de la machine. En particulier, nous sommes capables de filtrer de manière très précise des applications s'exécutant sous un UID spécifique, ce qui est le cas de nombreux serveurs, et ainsi compliquer la tâche d'un intrus qui serait parvenu à en exploiter une vulnérabilité.

4 Limites du concept et contournement

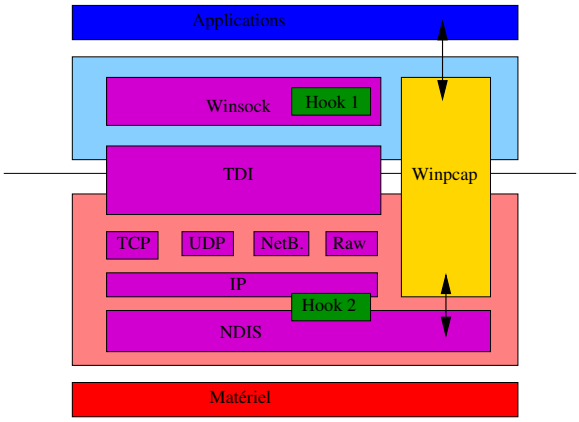
Le concept du firewall personnel est intéressant par le type de filtrage qu'il propose. Il permet ainsi de simplifier fortement la sécurisation d'un poste isolé connecté à Internet. Cependant, il est possible de contourner les restrictions mises en place.

Ces méthodes d'utilisation d'applications tierses autorisées reposent peu sur de véritables failles des firewalls, mais plus sur des faiblesses du système sous-jacent. Cependant, celles-ci suffisent à remettre en cause le fondement même de leur fonctionnement basé sur une confiance dans le comportement des applications autorisées.

4.1 Accès aux couches basses

La méthode la plus simple pour contourner un filtre de paquet local est d'injecter son flux à un niveau inférieur à celui auquel il opère. Des bibliothèques permettent

en effet d'injecter directement au niveau de la couche liaison des trames complètes. Parmis ces bibliothèques, on pourra citer les libnet et libdnet en environnement Unix et la winpcap [2] en environnement Windows. Des bibliothèques de capture de trames permettrons de lire le trafic reçu pour en extraire des informations. On pourra citer la libpcap sous Unix et la winpcap sous Windows.



Tab. 3. Positionnement de la couche winpcap

Cette dernière vient en effet s'intégrer directement au dessus de la couche NIDS, en parallèle des couches TCP/IP et TDI, et dispose de ses propres API (cf. Tab. 3) pour proposer aussi bien la capture que l'injection de paquets. De part sa position, elle évite tous les points de contrôle mis en place par un éventuel firewall personnel. Il faut bien différencier cette bibliothèque des sockets RawIP de Windows XP, qui restent contrôlables par un firewall personnel, puisque placées entre l'API Winsock et la couche IP.

Si l'appel à ces bibliothèques sous Unix suppose les droits superutilisateur, la situation est extrêmement différente sous Windows. D'une part parce que les Windows 95, 98 et Me ne possèdent pas de notion d'utilisateur et de droits. De fait, n'importe quelle application est susceptible d'accéder à de telles fonctionnalités. D'autre part parce que sous les version de Windows qui gèrent des droits (NT, 2000, XP), les droits d'administration ne sont nécessaires que pour l'installation de la bibliothèque. Une fois celle-ci installée, n'importe quel utilisateur à la possibilité d'y faire appel. Or, certaines machines possède cette bibliothèque, de part de vieilles installations ou pour des outils de monitoring réseau.

Fort de la possibilité d'accéder à ces deux types de bibliothèques, nous sommes en mesure de développer des applications capables de communiquer via le réseau sans jamais passer par les mécanismes de vérification du firewall, voire de rediriger les flux d'applications classiques dans un canal caché de ce type. L'outil WhiteCane [3,4], par exemple, permet de rediriger un flux UDP banal vers n'importe quelle application écoutant sur un port local.

Sous certains Unix, il est également possible d'insérer des modules permettant à des flux spécifiques d'échapper au couches de filtrage de paquets. Le module nfbypass [5] permet ainsi de faire échapper au contrôle de Netfilter tout paquet ayant pour source ou destination une adresse IP arbitraire.

4.2 Accès aux applications autorisées

Une technique simple consiste à exploiter des services mis à disposition par certaines applications via des liens OLE ou des méthodes ActiveX. On peut ainsi se servir de Microsoft Internet Explorer pour mandater des flux. Cette approche peut se révéler particulièrement intéressante. D'abord parce que Internet Explorer est autorisé dans l'immense majorité, sinon la totalité, des configurations de firewall personnel. Ensuite, parce que cette application ne peut pas être restreinte comme nous avons pu le constater plus tôt. Enfin, lorsque le passage par un mandataire (proxy) est nécessaire, Internet Explorer, configuré en conséquence, l'utilisera automatiquement, authentification comprise si celle-ci a déjà été effectuée une fois durant la session. On se reportera à la présentation de JAB [6].

La technique la plus efficace consiste certainement à exploiter des "fonctionnalités" des environnements Windows NT/2000/XP qui permettent à un utilisateur d'introduire du code, via une DLL, dans l'espace mémoire d'un processus existant, comme pour un plugin, via la directive CreateRemoteThread() [7,8,9]. Cette insertion nécessite un privilège particulier, SeDebug, que tous les utilisateurs du système possèdent par défaut. Cette technique permet donc à n'importe quelle application d'exécuter des commandes dans le contexte d'une autre, et ainsi d'en usurper les privilèges au niveau du firewall personnel pour initier ou recevoir des flux réseau. Un exploit générique a été publié sur Bugtraq pour illustrer cette méthode [10].

4.3 Attaque du firewall

Si nous disposons de droits suffisants, ce qui est le cas des environnements Windows 95/98/Me, nous pouvons tenter d'attaquer le firewall personnel directement. Parmi les technique possibles, nous pourrons citer la désactivation pure et simple du processus de protection (cf. virus BugBear [11]) ou l'altération des fichiers de configuration [12]. Il faut bien se garder à l'esprit qu'une application autorisée n'est finallement caractérisée que par trois éléments dans un fichier de configuration non signé :

- le nom de l'exécutable ;
- son chemin absolu sur le système de fichier ;
- une somme de contrôle MD5.

Du fait qu'une somme MD5 ne constitue pas une signature, elle peut-être recalculée par n'importe qui. Dès lors, nous sommes capable de modifier une entrée de plusieurs manières :

- modification du binaire et calcul d'une nouvelle somme MD5 ;
- modification du chemin et calcul d'une nouvelle somme MD5 ;
- modification du nom et calcul d'une nouvelle somme MD5 ;
- etc.

Nous pourrons aussi utiliser la technique d'injection de DLL pour en modifier le comportement si nous possédons les privilèges adéquats. Ceci nous garantit en plus la furtivité, puisqu'aucun signe ne trahira le canal que nous venons d'ouvrir. Enfin, il ne faut pas oublier que le firewall est un service comme les autres qui peut lui aussi faire l'objet de vulnérabilités pouvant conduire à la compromission du poste [13].

4.4 Problèmes de configuration

Les firewalls personnels souffrent en outre de problèmes de configuration qui altèrent leur efficacité. D'une part, tous les produits n'offrent pas, comme nous avons pu le voir précédemment, les mêmes fonctionnalités en terme de configuration. D'autre part, leurs soucis de simplicité et d'ergonomie entrainent parfois des problèmes de filtrage.

- L'impossibilité de limiter une application autorisée permet à un programme intrus d'obtenir un accès complet au réseau en exploitant une technique d'injection de flux par l'intermédiaire d'une application autorisée. C'est une faiblesse majeure des outils basiques qui ne peuvent donc pas répondre à une problématique de sécurisation avancée.
- L'impossibilité de gérer d'autre protocoles que TCP, UDP et ICMP permet à un programme intrus de créer des canaux en utilisant des protocoles que le firewall pourrait ne pas prendre en charge.
- La gestion des états souvent incomplète implique la mise en œuvre de jeux de règles compliqués et faibles, en particulier dans le cas de protocoles applicatifs impliquant des négociations de flux et celui de la gestion des erreurs ICMP.
- Le jeu de règles d'origine (configuration par défaut) laisse des ouvertures permettant le passage de flux vers l'extérieur, en particulier pour les règles de gestion de DHCP et DNS, comme l'ont montré des alertes, dont une récente, sur des produits commerciaux [14,**?**]. En outre, les règles de base ouvrent toujours l'interface de loopback. Cette dernière peut être exploitée par des outils pour servir de relai de communication entre un flux autorisé sur le réseau et une application à priori interdite écoutant sur l'interface de loopback [3,4], implémentant ainsi un canal caché.
- La méthode de configuration peut elle-même entrainer la mise en place de règles trop laxistes par manque d'information des messages ou par simple lassitude de l'utilisateur devant l'avalanche de popups à laquelle il doit faire face. En effet, il doit valider toute nouvelle application, revalider toute application patchée (cf. somme de contrôle) et prendre une décision pour tout paquet reçu non attendu.

4.5 Le discours marketting

Comme tout produit commercial, les firewalls personnels sont proposés à grands renforts de slogans percutants qui promettent la "protection optimale contre les pirates, les vers et le trojans". Comme nous l'avons vu précédemment, il faut grandement relativiser tout cela. En effet, ce discours ne tient pas compte des faiblesses des systèmes que ces produits sont sensés protéger. Il est par exemple difficile de promettre la protection contre les trojans pour un système sur lequel le premier processus venu peut en tuer un autre, le firewall en tête, comme l'ont appris à leurs dépends les utilisateurs infectés par BugBear [**?**],...

De plus, le discours semble volontairement alarmiste. Les messages d'alertes et les journaux d'activité utilisent souvent des termes du domaines de l'intrusion. Le but est vraisemblablement double, permettant d'une part à l'outil d'auto-justifier une certaine utilité et invitant d'autre part l'utilisateur à investir dans la version payante bien mieux founie en fonctionnalités.

5 Les bases d'une mise en œuvre efficace

5.1 Une utilisation éclairée

Lorsqu'on met en place une solution de sécurité, il est vital d'en maîtriser tous les aspects. L'utilisateur doit donc, pour en tirer le meilleur parti, savoir ce qu'est un firewall personnel, ce qu'il fait et ne fait pas, de manière à évaluer au mieux le niveau de protection fourni.

En outre, l'utilisateur doit être conscient que les concepts manipulés par ce type d'outil ne sont pas simples au point de ne pas avoir à les connaître. Il doit donc posséder un minimum de connaissances pour effectuer une configuration efficace.

5.2 Gestion efficace des droits

Pour assurer à l'outil un socle système adéquat, une gestion efficace des droits utilisateur est indispensable. La première mesure à prendre est de proscrire tout système n'implémentant pas les notions d'utilisateur et de droits d'accès. Comme nous l'avons vu précédemment, un tel système ne peut pas fournir les mécanismes pouvant empêcher le contournement trivial d'un firewall.

La gestion des droits doit être configurée conformément à la règle du moindre privilège. Il y a fort à parier que les utilisateurs d'un système bureautique n'aient pas besoin des privilèges d'administration ou SeDebug par exemple... Sur un système comme Windows XP en version Family, tous les utilisateurs ont les droits d'administration par défaut, c'est dommage.

Les applications autorisées par le firewall doivent être restreintes au maximum en terme de protocoles et de jeu de ports. Ces restrictions devront être reprises au niveau d'éventuels équipements de sécurité situés en aval dans l'infrastructure réseau.

5.3 Une bonne hygiène de vie

Enfin, de bonnes habitudes s'imposent et constituent la base d'une utilisation sûre :
- travailler avec compte non privilégié ;
- ne pas ouvrir n'importe quel document ;
- mettre en place un bon antivirus et le tenir à jour ;
- tenir son système et ses applications à jour ;
- être attentif aux messages d'alertes des applications.

Les voies les plus efficaces pour introduire un trojan sont d'une part le génie social, basé sur l'exploitation de la crédulité des utilisateurs, et d'autre part les failles des applications courantes comme Internet Explorer, Outlook et Outlook Express, la combinaison des deux se révélant souvent catastrophique.

6 Conclusion

Comme nous avons pu le voir, le concept de firewall personnel est intéressant, mais souffre de limitations parfois fortes. Certaines sont dûes à l'implémentation des outils, d'autres au système sur lesquels ils s'exécutent. Quoi qu'il en soit, ce type d'outil n'en perd pas pour autant tout intérêt et se révèle un composant indispensable de protection des postes directement connectés à Internet.

Cependant, il ne peut assurer à lui seul la protection intégrale d'un poste informatique quel qu'il soit, loin de là. Il doit impérativement être complété d'autres mécanismes de sécurité. Mais plus que tout dispositif technique, ce sont surtout une prise de conscience et une éducation de l'utilisateur qui constitueront les mesures les plus efficaces

7 Références

Références

1. Linux Netfilter, `http://www.netfilter.org/`
2. Winpcap, `http://winpcap.polito.it/`
3. M. Blanc, É. Detoisien, A. Guignard & L. Oudot, Pénétration de réseaux et backdoors furtives, Linux Magazine Hors Série 12, novembre 2002.
4. Éric Detoisien, WhiteCane,`http://valgasu.rstack.org/tools/whitecane.zip`
5. Truff, nfbypass, `http://projet7.tuxfamily.org/factory/releases/nfbypass.c`
6. Nicolas Grégoire, Prise de contrôle via Internet Explorer d'une machine compromise située en réseau inconnu, SSTIC 2003
7. Ivo Ivanov, API hoking revealed, `http://codeguru.earthweb.com/system/apihook.html`
8. Zoltan Csizmadia, Injecting a DLL into another process's address space, `http://codeguru.earthweb.com/dll/LoadDll.shtml`
9. John Peloquin, Remote library loading, `http://codeguru.earthweb.com/dll/Loader.html`
10. Xenophi1e, Thermite, Bugtraq `http://cert.uni-stuttgart.de/archive/bugtraq/2003/02/msg00268.html`
11. Virus BugBear, `http://www.virusbtn.com/resources/viruses/bugbear.xml`
12. Kerio Personal Firewall Remote Authentication Packet Buffer Overflow Vulnerability, `http://www.securityfocus.com/bid/7180`
13. ZoneAlarm Personal Firewall Port 67 Vulnerability, `http://www.securityfocus.com/bid/1137`
14. Kerio Personal Firewall Firewall Filter Bypass Vulnerability, `http://www.securityfocus.com/bid/7436`

Profils Propres pour la Détection d'Intrusion

Yacine Bouzida and Sylvain Gombault

Département RSM GET/ENST Bretagne
02, rue de la Châtaigneraie, CS 17607
35576 Cesson Sévigné CEDEX, FRANCE,
{Yacine.Bouzida,Sylvain.Gombault}@enst-bretagne.fr

Résumé Nous présentons, dans cet article, une nouvelle méthode de détection d'intrusion appartenant à la famille comportementale qui est basée sur l'analyse en composantes principales (ACP). Cette approche fonctionne en projetant les profils des utilisateurs sur un espace de traits qui peut décrire de signifiantes variations entre les profils. Ces traits sont connus sous le nom de *"profils propres"* car ils sont les vecteurs propres de l'ensemble des profils. L'opération de projection caractérise un profil utilisateur par une somme pondérée de l'ensemble des profils. Pour détecter si un profil est anormal, il suffit de comparer ses poids à ceux des profils utilisateurs connus. Les principaux avantages de cette méthode sont : (i) elle permet, au premier lieu, d'apprendre les profils utilisateurs ensuite déterminer si un nouveau profil correspond ou non à ceux des utilisateurs connus, (ii) son implémentation est très simple sur tous les systèmes ayant des mécanismes d'audit de sécurité et (iii) elle est robuste et permet de produire de très hauts taux de détection. Nous introduirons quelques formules nécessaires pour calculer les profils propres, ensuite nous décrirons l'algorithme de détection basé sur ces profils propres. Au départ, nous avons appliqué cette méthode sur un ensemble de profils des utilisateurs simulés, ensuite, nous avons utilisé un ensemble de profils des utilisateurs dans un réseau réel en analysant leur activité de navigation Web et les résultats expérimentaux sont très satisfaisants.

1 Introduction

Toute violation de la politique de sécurité d'un système informatique est vue comme un objectif potentiel d'intrusion [1].

La mise en place d'un IDS (Intrusion Detection System) demande la surveillance continue de ce qui se passe sur le système ou sur le réseau que l'on veut protéger. Cette surveillance est réalisée via certains mécanismes d'audit de sécurité.

Depuis le début des années 90, plusieurs outils de détection d'intrusion ont été développés et ont vu le jour. Ces outils aident les Officiers de sécurité *(SSOs : Site Security Officers)* à identifier d'éventuelles violations de la politique de sécurité. En général, il existe deux grandes familles de détection d'intrusion; l'approche comportementale et l'approche par scénario.

Approche comportementale Cette approche est basée sur le comportement d'utilisateur et/ou application, on parle alors de profil utilisateur ou comportement d'une application. Elle a été proposée par Anderson en 1980 et reprise par Denning [2]

en 1987. Anderson a proposé de décrire le profil utilisateur par un ensemble de mesures pertinentes modélisant au mieux son comportement afin de détecter par la suite toute déviation de son comportement habituel ainsi appris. Cette approche cherche alors à répondre à la question : "le comportement actuel de l'utilisateur et/ou application est il cohérent avec son comportement passé?". (pour plus de détail des différents outils comportementaux, se référer à [5,6,7,8,12,13]).

Approche par scénario Cette approche cherche à retrouver des attaques connues dans le fichier d'audit. Elle nécessite donc une connaissance, a priori, des attaques bien définies. Cette approche cherche alors à répondre à la question : "le comportement actuel de l'utilisateur et/ou application contient-il une attaque connue?". Dans ce cas, une construction d'une base de données d'attaques ou de scénarios d'attaques est nécessaire. (voir [4,9,10,11,14,16] pour d'éventuels outils de ce type d'approche).

Une détection d'intrusion hybride qui combine ces deux techniques en même temps peut être ajoutée comme une troisième approche des IDS.

L'analyse en composantes principales (ACP) [15] est une méthode populaire utilisée d'une manière excessive dans la réduction des dimensions d'espace et elle est utilisée dans plusieurs textes d'analyse multivariée. Elle est appliquée dans plusieurs domaines tels que la compression de données, le traitement d'images et la reconnaissance des formes.

L'ACP consiste à réduire un système complexe de corrélations en un plus petit nombre de dimensions [17]. Ayant un ensemble de vecteurs observés $\{v_i\},\ i \in \{1,\ldots,N\}$, les q principaux axes $\{w_j\},\ j \in \{1,\ldots,q\}$ sont les axes orthonormés sur lesquels la variation sous une projection est importante. Il est prouvé que les vecteurs w_j sont données par les q vecteurs propres dominants (i.e. ceux associés aux plus grandes valeurs propres) de la matrice de covariance $C = \sum_i \frac{(v_i-\overline{v})(v_i-\overline{v})^T}{N}$ tel que $Cw_j = \lambda_i w_j$, où $\overline{v}$ est la moyenne arithmétique simple. $u_i = C^T(v_i - \overline{v})$, de dimension q, est ainsi une représentation réduite du vecteur observé v_i.

Dans les sections qui suivent, nous présenterons une nouvelle méthode de détection d'intrusion comportementale basée sur l'analyse en composantes principales. Le paragraphe 2 présente l'approche des profils propres, le paragraphe suivant illustre les différentes étapes de la méthode. Le paragraphe 4 présente les premiers résultats expérimentaux sur un certain nombre de profils utilisateurs sous unix et le paragraphe 5 donne des résultats expérimentaux réels sur un fichier web log proxy. Enfin, le paragraphe 6 conclut l'article.

2 L'approche des profils propres

La plupart des travaux effectués sur l'approche comportementale ne se sont intéressés que sur les mesures d'un profil utilisateur qui sont importantes pour les utiliser dans l'algorithme de détection. Ceci nous a suggéré qu'une théorie d'information codant et décodant les comportements des utilisateurs peut fournir de nouvelles informations sur ces comportements contenant de signifiantes caractéristiques.

Dans le langage de la théorie de l'information, nous voulons extraire les informations essentielles dans un profil utilisateur, le coder d'une manière très efficace, ensuite comparer un comportement ainsi codé avec une base de données des comportements utilisateurs codés de la même façon. Une méthode simple pour extraire les informations contenues dans un profil consiste à capturer la variation dans une collection de comportements des utilisateurs et utiliser ces informations pour coder et comparer les profils des comportements utilisateurs.

Dans le langage mathématique, nous souhaitons trouver les composantes principales de la distribution des comportements, ou bien trouver les vecteurs propres de la matrice de corrélation de l'ensemble des profils utilisateurs, en considérant un comportement comme un point (ou vecteur) dans un espace de dimension égale au nombre des différentes métriques utilisées. Ces métriques sont des mesures qui peuvent être le temps CPU utilisé, le nombre de commandes introduites par l'utilisateur durant une session, le nombre de chaque type d'événement audité durant un intervalle de temps,...

Ces vecteurs propres peuvent être considérés comme un ensemble de traits qui caractérisent la variation entre les comportements des utilisateurs. Chaque position d'un comportement contribue plus ou moins pour chaque vecteur propre que nous appelons profil propre.

Chaque profil peut être représenté par une combinaison linéaire des profils propres. Chaque profil peut être aussi *approximé* en utilisant seulement les meilleurs profils —ceux correspondant au plus grandes valeurs propres— et qui comptent pour la plus grande variation dans l'ensemble des profils des utilisateurs.

Un profil normal d'un utilisateur, dans notre système, consiste en un ensemble de mesures statistiques telles que celles proposées par Denning [2] :
- le temps CPU utilisé,
- le nombre de mots de passe erronés introduits durant un intervalle de temps,
- le nombre de chaque type de commande introduite par l'utilisateur pendant un intervalle de temps,
- le nombre de fichiers ouverts pendant une période,
- le nombre d'applications exécutées durant une session,
- le nombre de page web visitées pendant une journé, ...

L'ensemble de ces mesures est collecté dans un vecteur de dimension n appelé vecteur du profil utilisateur représentant le profil de l'utilisateur durant une session ou un intervalle de temps choisi, où n est le nombre de mesures statistiques mentionné ci-dessus.

Si Γ est le profil qui correspond à un comportement d'un certain utilisateur, alors nous pouvons écrire

$$\Gamma = \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{pmatrix} \tag{1}$$

où $m_i, i = 1, \ldots, n$ sont les mesures caractérisant le profil utilisateur.

3 Les différentes étapes de la méthode

Le schéma général de notre système est décrit dans la figure (1).

Nous pouvons résumer les différentes étapes de cette méthode comme suit :

1. Un processus d'initialisation est nécessaire, il consiste à :
   (a) auditer l'ensemble des profils des utilisateurs du réseau,
   (b) calculer les vecteurs propres de cet ensemble,
   (c) extraire les profils traits des profils connus,
   (d) pour chaque classe (utilisateur), déterminer le vecteur trait de référence,
   (e) déterminer le seuil de chaque classe.

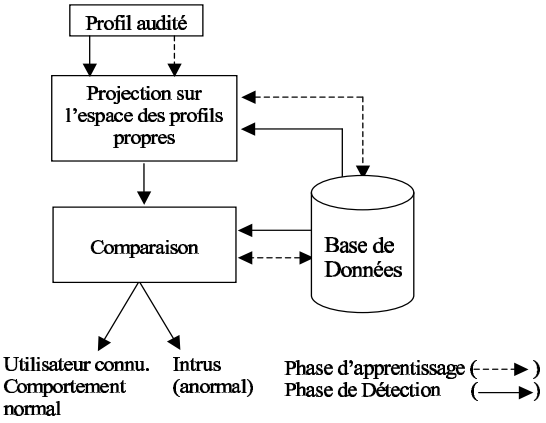


Fig. 1. Structure générale de notre système

2. Un processus de classification est ensuite utilisé pour détecter si un nouveau comportement ainsi audité est normal ou non. Il comporte les étapes suivantes :
   (a) projection du profil audité sur l'espace des profils propres,
   (b) comparaison de son vecteur trait avec ceux des profils connus pour l'identification et la détection.

3.1 Le processus d'initialisation

Considérons un système comportant N utilisateurs. Chaque utilisateur est audité NU fois pendant des périodes différentes. Alors le nombre de profils est de $M = N \times NU$.

Profil moyen des utilisateurs

Soit l'ensemble des profils audités $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$. Le profil moyen Ψ de cet ensemble est défini comme étant le centre de gravité des mesures des profils utilisés pour l'apprentissage.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i \tag{2}$$

Profil caricature d'un utilisateur

Le profil caricature Φ_i est défini comme la différence entre le profil connu Γ_i et le profil moyen Ψ :

$$\Phi_i = \Gamma_i - \Psi \tag{3}$$

Calcul des profils propres

Les profils propres sont les vecteurs propres de la matrice de covariance C où

$$C_{(n \times n)} = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T = AA^T \tag{4}$$

et

$$A_{(n \times M)} = \frac{1}{\sqrt{M}} [\Phi_1 \Phi_2 \ldots \Phi_M] \tag{5}$$

Soit U_k le $k^{\text{ième}}$ vecteur propre de C associé à la valeur propre λ_k et
$\mathbf{U} = [\mathrm{U_1 U_2 \ldots U_M}]$ est la matrice de ces vecteurs propres (profils propres). Alors

$$CU_k = \lambda_k U_k \tag{6}$$

tel que

$$U_k^T U_n = \begin{cases} 1 \text{ si } k = n \\ 0 \text{ si } k \neq n \end{cases} \tag{7}$$

le vecteur trait du profil est :

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix} \tag{8}$$

Les poids $\omega_i, i = 1, \ldots, M$ décrivent la contribution de chaque profil propre dans la représentation du profil introduit, traitant les vecteurs propres comme l'ensemble de base des comportements des utilisateurs.

Ce vecteur trait peut être, par la suite, utilisé dans un algorithme de classification standard pour trouver la classe dans les comportements établis dans la phase d'apprentissage qui décrit au mieux ce nouveau comportement. La première idée qui vient à l'esprit pour déterminer quelle classe produit la meilleure description d'un vecteur en entrée est de trouver une classe dans l'espace des profils propres qui minimise la distance euclidienne avec le profil en entrée décrite dans le paragraphe suivant.

Si la taille d'un vecteur de profil est n (nombre de mesures considérées), la matrice C est de taille $n \times n$ et le calcul de n valeurs propres et vecteurs propres est une tâche très difficile à accomplir si on considère des centaines de mesures. Ainsi, une méthode faisable, en utilisant les équations (4) et (6), on peut obtenir

$$A\,A^T\,U_k = \lambda_k\,U_k \tag{9}$$

$$A^T\,A\,(A^T\,U_k) = \lambda_k\,(A^T\,U_k) \tag{10}$$

soit

$$Y_k = A^T\,U_k \tag{11}$$

alors

$$A^T\,A\,Y_k = \lambda_k\,Y_k \tag{12}$$

d'après l'équation (12), Y_k est un vecteur propre de la matrice $A^T A$ associé à la valeur propre λ_k.
Soit $X_k = \alpha_k Y_k$, donc X_k est aussi un vecteur propre de la matrice $A^T A$
d'après l'équation (11)

$$X_k^T\,X_k = (\alpha_k\,A^T\,U_k)^T\,(\alpha_k\,A^T\,U_k) = \alpha_k^2 \lambda_k\,U_k^T\,U_k \tag{13}$$

Comme $U_k^T\,U = 1$
Alors

$$X_k^T\,X_k = \alpha_k^2 \lambda_k \tag{14}$$

Pour obtenir un vecteur X_k normé (*i.e.* $X_k^T\,X_k = 1$) il faut avoir

$$\alpha_k^2 \lambda_k = 1 \Rightarrow \alpha_k = \frac{1}{\sqrt{\lambda_k}} \tag{15}$$

D'après les équations (11) et (9), nous avons :

$$A\,Y_k = A\,A^T\,U_k \tag{16}$$

$$A\,Y_k = \lambda_k\,U_k \tag{17}$$

$$U_k = \frac{1}{\lambda_k}\,A\,Y_k = \frac{1}{\lambda_k\,\alpha_k}A\,X_k = \frac{1}{\sqrt{\lambda_k}}A\,X_k \tag{18}$$

D'où

$$U_k = \frac{1}{\sqrt{\lambda_k}}A\,X_k \tag{19}$$

Avec cette analyse, les calculs sont réduits d'une manière considérable, de l'ordre du nombre de mesures utilisées jusqu'à l'ordre du nombre de profils utilisés dans la phase d'apprentissage, car X_k est un vecteur propre de $A^T A$ qui est de dimension très réduite M.

Calcul des vecteurs traits Le vecteur trait Ω_i d'un profil Γ_i est obtenu en projetant son vecteur caricature Φ_i sur l'espace des profils propres

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix}$$

où $\mathbf{U} = [\mathrm{U_1 U_2 \ldots U_M}]$
Ainsi, chaque profil est représenté par un ensemble de M' points (le vecteur trait).

Organisation en classes A chaque utilisateur lui correspond une classe composée des vecteurs traits obtenus en projetant les NU profils audités dans l'espace des profils propres. Chaque classe k est représentée par un vecteur trait de référence Ω^k :

$$\Omega^k = \frac{1}{NU}\sum_{i=1}^{NU}\Omega_i^k \tag{20}$$

Où Ω_i^k est le $i^{\text{ième}}$ vecteur trait du $k^{\text{ième}}$ utilisateur (classe).
La méthode la plus simple qui détermine la classe la plus proche du profil introduit Γ_i consiste à déterminer la classe k qui minimise la distance Euclidienne

$$\epsilon_k = \|\Omega_i - \Omega^k\|_2 \tag{21}$$

En plus, ce profil qui vient d'être audité sera affecté à la classe la plus proche si la condition suivante est vérifiée : $\epsilon_k < seuil\theta_k$ (θ_k : seuil choisi par expérience.) dans le cas contraire, le profil est classé comme intrus!

3.2 Le processus d'identification et de détection

Le processus de détection comporte les étapes suivantes

1. auditer un nouveau profil Γ_i à vérifier,
2. calculer son profil caricature $\Phi_i = \Gamma_i - \Psi$,

3. projeter le profil caricature Φ_i sur l'espace des profils. On obtient alors le vecteur trait :

$$\Omega_i = U^T \times \Phi_i = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix}$$

où **U** est la matrice des profils propres $U_i,\ i = 1, \ldots, M$

4. Déterminer la classe k qui minimise la distance $\epsilon_k = \|\Omega_i - \Omega^k\|_2$
Si $\epsilon_k < \theta_k$ **alors**
le nouveau profil observé est celui du $k^{\text{ième}}$ utilisateur
Sinon
le nouveau profil observé est anormal.
Fsi

4 Résultats expériementaux préliminaires

Afin de montrer la rapidité, la robustesse et la simplicité de notre approche, nous avons considéré un simple exemple (qui peut être considéré comme une petite partie de notre système car on ne prend en compte que les occurrences des événements et quelques commandes) qui consiste en quatre types d'utilisateurs qui ont été utilisés dans [14,16] : l'utilisateur de mail, le novice, le développeur et l'utilisateur inexpérimenté. Les enchaînements qui suivent correspondent à des activités possibles pendant une courte période, de l'ordre de 30 minutes. Les mesures que nous considérons dans notre première expérimentation sont les différentes commandes[1] introduites par l'utilisateur durant un intervalle de temps.

Définition de ces types de comportements Les différents profils utilisateurs sont décrits dans le tableau (1) après avoir transformé les commandes UNIX en événements d'audit AIX correspondant (pour plus de détail, voir [16]) pendant la session d'audit :

Application de la méthode des profils propres Appliquons les différentes étapes citées ci-dessus à ces différents utilisateurs :

Le processus d'initialisation

1. les profils générés par ces comportements d'utilisateurs (la base d'apprentissage) sont décrits dans le tableau (1),

2. le profil moyen de ces comportements est le suivant (en utilisant l'équation (2)),

$$\Psi^T = \tfrac{1}{4}\sum_{i=1}^{4} \Gamma_i^T =$$
$$(0\,0\,0\,0\,0\,2.25\,1.75\,1.75\,0\,0\,0\,0\,1.75\,0\,0\,0.75\,26.5\,0\,2.5\,0\,0\,0\,0\,0\,0.5\,1\,0.25),$$

[1] Ces commandes sont traduites en évévenements d'audit AIX

	L'utilsateur de mail Γ_1	Le novice Γ_2	Le développeur Γ_3	L'utilisateur expérimenté Γ_4
user_login fail				
user log (23h à 6h)				
short_session				
use_su Ok				
use_su fail				
who, w, finger...		2	3	4
more, pg, cat,...		1	3	3
ls OK	2	2	3	
ls fail				
df, hostname, uname				
arp, netstat, ping				
ypcat				
lpr	4	1	1	1
rm, mv	4			
ln				
whoami,id				
rexec, rlogin, rsh			1	2
proc_Execute	16	18	55	17
Proc_SetPetri				
file_open fail		4	2	4
file_open fail cp				
file_open .netrc				
file_read lpr				
file_read passwd ...				
file_write				
passwd...fail				
file_write cp ok		1	1	
file_unlink rm	4			
file_mode				1

Tab. 1. les profils générés des différents utilisateurs

3. calculer le vecteur trait (équation (3) pour chaque utilisateur $(\Phi_1, \Phi_2, \Phi_3, \Phi_4)$),
4. calculer la matrice A à partir de l'équation (5),
5. calculer les vecteurs propre de la matrice de covariance $C = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^T = AA^T$: Ceci peut être fait en utilisant le résultat défini dans l'équation(19).

$$A^T A = \begin{bmatrix} 37.15 & 20.03 & -77.40 & 20.21 \\ 20.03 & 19.65 & -60.28 & 20.59 \\ -77.40 & -60.28 & 204.78 & -67.09 \\ 20.21 & 20.59 & -67.09 & 26.28 \end{bmatrix}$$

Les valeurs propres de cette matrice sont

$$\begin{bmatrix} 273.792 \\ 12.249 \\ 1.833 \\ 0 \end{bmatrix}$$

Leurs vescteurs propres correspondants sont

$$X_1(273.792) = \begin{bmatrix} -0.329 \\ -0.254 \\ 0.865 \\ -0.282 \end{bmatrix}, X_2(12.249) = \begin{bmatrix} -0.786 \\ 0.268 \\ -0.038 \\ 0.556 \end{bmatrix}, X_3(1.830) = \begin{bmatrix} 0.157 \\ -0.783 \\ 0.026 \\ 0.601 \end{bmatrix}$$

Les vecteurs propres $U_k, k = 1, \ldots, 3$ de la matrice de covariance $C = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T = A\,A^T$ sont obtenus à partir de l'équation (19). Dans cet exemple, nous avons considéré un seul profil pour chaque utilisateur. Ainsi, chaque profil correspond exactement à la classe qu'il forme $\Omega_k, k = 1, \ldots, 4$
$(carN = 4)$.
Le tableau (2) suivant présente la distance euclidienne entre les différents utilisateurs après projection (à partir de l'équation(21)).

	Utilisteur1	Utilisteur2	Utilisteur3	Utilisteur4
Utilisteur1	0	4.095	19.927	4.797
Utilisteur2		0	18.577	2.179
Utilisteur3			0	19.111
Utilisteur4				0

Tab. 2. la distance euclidienne entre les diffŕentes classes

Si nous essayons de représenter les quatre profils dans le nouvel espace engendré par les profils propres, leur projection est représentée dans la figure (2) suivante.

5 Application de cette méthode sur un fichier web log

Ce paragraphe présente l'ensemble des tests pour mettre en œuvre le modèle de l'ACP et son application sur un fichier web log réel.

Plusieurs systèmes (voir pour l'instant [19]) sont implémentés pour des connaissances implicites à partir des fichiers web log mais aucun d'entre eux, à notre connaissance, ne s'est intéressé à répondre aux questions suivantes : le comportement de l'utilisateur change -t- il avec le temps, est il possible de caractériser les comportements utilisateurs en utilisant les fichiers web log et comment ?

Nous avons utilisé un fichier log (access.log) généré par SQUID qui est un logiciel de cache (web proxy cache) gratuit et open source [18]. Ce software est installé sur un pare-feu (firewall) reliant un réseau local contenant 6 machines d'utilisateurs qui sont des stagiaires à l'ENST-Bretagne.

Nous avons audité ces utilisateurs pendant un mois et effectué des tests sur ce fichier access.log ainsi généré qui est d'une taille d'environ 43 Méga Octets. Chaque entrée dans ce fichier a dix champs ayant le format suivant : **[Timestamp] [Elapsed-Time]**

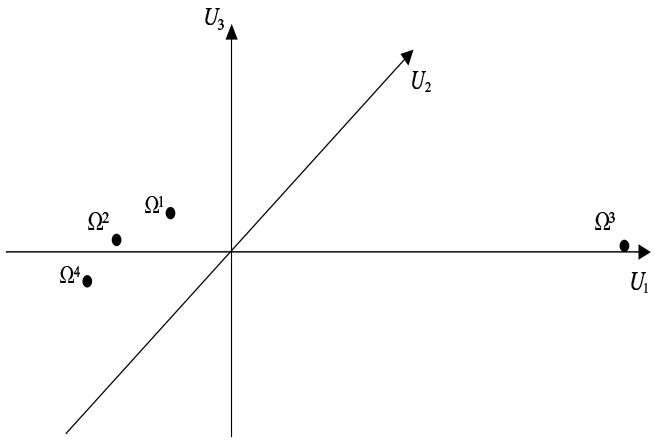


Fig. 2. La projection des profils des utilisateurs sur le nouvel espace des profils propres

[Client-IP] [**Action**]/[**Code**] [**Size**] [**Method**] [**URL**] [**Hierarchy**]/[**Content Type**]
Avec :

1. Timestamp : L'horodatage indiquant l'heure de la requête, exprimé en seconde à partir du 1 janvier 1970 et avec une résolution en millisecondes,
2. Elapsed Time : Ce champ indique le temps écoulé de la requête en millisecondes,
3. Client-IP : L'adresse IP du client,
4. Action : Ce champ décrit comment la requête a été traitée localement dans le serveur proxy,
5. Code : Le code d'état envoyé au client comme réponse à sa requête,
6. Size : Ce champ enregistre le nombre d'octets transférés du proxy vers le client pour une requête donnée,
7. Method : La méthode http demandé par le client,
8. URL : L'URL du document demandé,
9. Hierarchy : Ce champ décrit où et comment le document est cherché,
10. Content : Le type du document spécifié.

Exemple :
985796546.734 1 192.168.xx.xx TCP_MISS/200 207 GET http ://good.niagara .sightpath.com/images/homepage/smb-job-npm.gif - NONE/-image/gif

Avant de passer à la phase d'identification et de détection, le système doit acquérir un ensemble d'informations pour accomplir cette tâche. C'est ce qu'on appelle *la phase d'apprentissage*. Son rôle est de fixer les paramètres du système pour fournir les meilleurs résultats. C'est la tâche la plus importante et la plus délicate.

Elle est importante car les performances du système dépendent directement du choix de ces paramètres. Elle est délicate car il n'existe ni de formules ni de méthodes directes permettant la détermination de leurs valeurs, il faut les calculer par expérience.

Afin d'utiliser l'ACP, la matrice de corrélation doit être calculée. Pour ce faire, chaque élément du vecteur de profil Γ_i détermine le nombre de requêtes pour une URL

donnée pendant une session d'audit (dans notre cas, une session consiste en une journée pendant la période d'activité des utilisateurs (de 08h00 à 19h00)). Ainsi, la taille du vecteur profil Γ_i dépend du nombre d'URL visitées par l'ensemble des utilisateurs choisis pour l'apprentissage durant une certaine période.

Dans notre expérimentation, nous avons utilisé six clients pour tester notre méthode. Le nombre de profils audités choisis pendant un mois était de 96 profils, soit 16 profils pour chaque utilisateur. Le reste des profils n'est pas sélectionné car ils correspondent à des journées de non activité (tel que les week-end et jours d'absences de ces utilisateurs).

Au départ, nous avons divisé cette base de profils en deux classes. La première, utilisée pour la phase d'apprentissage, est composée de 24 profils des trois premiers utilisateurs représentant leurs profils durant les deux premières semaines, soit 8 profils par utilisateur. Les 8 autres profils des 3 premiers clients sont utilisés pour tester la capacité de généralisation du système. Les 48 profils restant des 3 derniers clients, que l'on appelle ensemble non familier, sont utilisés pour évaluer la capacité du système pour rejeter les utilisateurs inconnus. Nous avons considéré les taux suivants pour présenter nos résultats :
Taux d'identification avec succès Le pourcentage de profils identifiés avec succès,
Taux de confusion ou faux négatifs Le pourcentage de profils qui ne sont ni identifiés avec succès ni rejetés,
Taux de rejet avec succès Définit le taux de profils non familiers qui sont rejetés,
Taux de faux rejet ou faux positifs Le taux de l'ensemble familier qui est rejeté.
Le tableau (3) suivant récapitule les résultats obtenus

	Ensemble Familier	
	Ensemble Appris	Ensemble non Appris
Taux d'identification avec succès	17/24 (70,83%)	10/24 (41,67%)
Taux de faux rejet ou faux positifs	7/24 (29,17%)	14/24 (58,33%)

Tab. 3. Les performances du sytème avec la première expérimentation

Ce tableau montre que les résultats obtenus ne sont pas intéressants, d'ailleurs 29,17 % des profils de l'ensemble familier ne sont pas reconnus. La cause de ces faux positifs est due en réalité aux changements des comportements des utilisateurs avec le temps.

Ceci nous a poussé à utiliser une autre stratégie qui consiste à auditer les utilisateurs pendant une période (nous avons choisi cette période égale à 4 jours successifs), ensuite nous introduisons les profils des utilisateurs de la journée qui suit pour la détection. Nous avons gardé trois utilisateurs pour la phase d'apprentissage. Au fur et à mesure que nous avançons d'une journée, la base d'apprentissage est mise à jour en considérant les profils des 3 utilisateurs pendant les quatre derniers jours d'audit choisis précédant la détection sauf dans le cas où le profil réel de l'utilisateur est détectée comme anormal. Ceci nous rappelle l'approche statistique de DENNING [2] qui détermine, à partir de n observations $x_1, x_2, \ldots, x_n$ d'une variable aléatoire x si une nouvelle observation x_{n+1} est anormale par rapport aux n observations précédentes.

Dans notre cas, chaque observation représente le profil d'un utilisateur audité pendant une journée. Le tableau (4) résume les différents résultats obtenus en utilisant cette stratégie.

	Ensemble familier		Ensemble non familier
	Profils Appris	Profils non Appris	
Taux d'identification avec succès	46/46 (100%)	34/36 (94.44%)	–
Taux de confusion ou faux négatifs	0%	0%	0%
Taux de rejet avec succès	–	–	100%
Taux de faux rejet ou faux positifs	0 (0%)	2/36 (5,56%)	–

Tab. 4. Les performances du sytème avec la deuxième stratégie

Dans ce deuxième tableau, nous avons testé notre système sur 36 profils, les douze premiers profils sont utilisés pour le premier apprentissage (4 profils par utilisateur).

Les capacités du système sur le fichier proxy web log
En utilisant un apprentissage régulier au fur et à mesure de l'évolution des profils des utilisateurs, la méthode n'a trouvé aucun problème pour reconnaître les profils qui lui ont été présentés. Concernant sa généralisation, elle a pu identifié 34 nouveaux profils sur 36 et rejeter tous les autres profils de l'ensemble non familier.

Cependant, le temps de traitement pour la phase d'apprentissage varie selon la taille du fichier de log audité pendant les 4 jours d'apprentissage. Dans notre cas, il varie entre 2 à 3 secondes en moyenne. Le temps de détection et d'identification est de l'ordre de la dixième de seconde car la détection consiste juste en la projection du nouveau profil sur la base des profils propres et le calcul de distance entre son vecteur trait et les différentes classes (3 classes pour les 3 premiers utilisateurs).

6 Conclusion

En résumé, cette nouvelle méthode apporte les idées suivantes :

1. Elle introduit une nouvelle méthode de détection d'intrusion dans l'approche comportementale qui permet une bonne classification des différents utilisateurs sous Unix et elle a démontré sa capacité d'apprentissage de profils et de généralisation en utilisant les fichiers log,
2. Elle présente une solution simple à exploiter pour le problème de détection d'intrusion en utilisant l'analyse en composantes principales.

Il est à noter que cette méthode peut facilement détecter les intrus de type masquerader [3]. D'ailleurs, si un utilisateur change de poste et garde toujours le même profil alors son vecteur trait sera proche de sa vraie classe. Dans ce cas, il suffit juste de vérifier l'adresse IP de cette classe pour savoir de quel utilisateur il s'agit. Ainsi, cet utilisateur est détecté comme masquerader.

Références

1. F. Cuppens et al. Recognizing Malicious Intention in an Intrusion Detection Process. Second International Conference on Hybrid Intelligent Systems, Santiago, Chili, December 2002.
2. D. Denning : An Intrusion Detection Model, IEEE Transactions on Software Engineering, Vol. 13 (2), 1987, pp. 222,232.
3. J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James. P. Anderson Co., Fort Washington, Pennsylvania, April 1980.
4. N. Habra, B. Le Charlier, A. Mounji, I. Mathieu, Asax : Software architecture and rule based language for universal audit trail analysis, in Y. Deswarte, G. Eizenberg, J.J. Quinsquater (Eds.), Proc. 2nd Symp. on Research in Computer Security (ESORICS), Toulouse, Berlin, Lecture Notes in Computer Science, vol. 648, Springer, Berlin, November 1992.
5. S.R Snapp et al. : DIDS (Distributed Intrusion Detection System), motivation, architecture, and early prototype, Proc. 14th National Computer Security Conf., Washington, DC, October 1991, pp. 167-176.
6. H. Debar et al. : A neural network component for an intrusion detection system, Proc. 1992 IEEE Computer Society Symp. On research in security and Privacy, Oakland, CA, May 1992, pp. 240-250.
7. D. E Denning, P. G. Neumann : Requirements and model for IDES, a real time intrusion detection expert system, Technical Report , Computer science Laboratory, SRI International, Menlo Park, CA 1985.
8. R. Jagannathan et al. : System design document : Next Generation Intrusion Detection Expert System (NIDES). Technical Report A007/A0014, SRI International, Ravenswood Avenue, Menlo Park, CA 94025, March 1993.
9. S. Kumar, E. Spafford : A pattern matching model for misuse intrusion detection, Proc. 17th National Computer security Conf. October 1994, pp. 11-21.
10. P. Porras, R. Kemmerer : penetration state analysis, a rule based intrusion detection approach, Proc. 8th Annual Computer Security Applications Conf., November 1992, pp. 220-229.
11. K. Ilgun : Ustat, a real time intrusion detection system for UNIX, Proc. IEEE Symp. On Research on Security and Privacy, Oakland, CA, May 1993, pp. 16-28.
12. S. Smaha, Haystack : an intrusion detection system, 4th Aerospace Computer Security Applications Conf. October 1988, pp. 37-44.
13. H. S. Vaccaro, G. E. Liepins, Detection of anomalous computer session activity, Proc. IEEE symp. On Research in Security and Privacy, 1989, pp. 280-289.
14. L. Mé : Gassata, a genetic algorithm as an alternative tool for security audit trail analysis, presented at RAID, the first international workshop on Recent Advances in Intrusion Detection, October 1998.
15. I. T. Jolliffe. Principal Component Analysis. 2nd Edition, New York : Springer Verlag, 2002.
16. L. Mé. Audit de Sécurité par Algorithmes Génétique, University of Rennes 1, PhD Thesis, Order N° 1069, July 7th 1994.
17. H. Hotelling. Analysis of a complex statistical variables into principal components. Journal of Educational Psychology 24, pp. 417-441, 1933.

18. http ://www.squid-cache.org/
19. O. R. Zaïane, M. Xin, J. Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs in Proc. ADL'98 (Advances in Digital Libraries), Santa Barbara, April 1998.

Formats de fichiers et code malveillant

Philippe Lagadec

DGA/CELAR
philippe.lagadec@ifrance.com

Résumé Cet article se penche sur le cas particulier des fichiers, qui peuvent pénétrer sur un intranet par de nombreux moyens différents (web, e-mail, dis- quette, CDROM, ...), la plupart du temps sans réel filtrage. Une fois qu'il est ouvert par un utilisateur, un fichier peut facilement agir sans aucun contrôle sur un intranet et mettre en jeu sa sécurité. Afin de préciser les menaces, quelques-uns des formats de fichiers les plus classiques sous Windows seront présentés : exécutables, scripts, HTML, documents Office, PDF, ... Cela permet d'ébaucher une classification des différents formats de fichiers, afin de distinguer ceux qui sont inoffensifs et ceux qui présentent une menace potentielle. Il est ainsi possible de déterminer une politique de filtrage adaptée au réseau à protéger. Cet article présente enfin divers moyens techniques et organisationnels disponi- bles aujourd'hui pour contrer au mieux ces menaces.

1 Introduction

La plupart des réseaux d'entreprise connectés à Internet sont aujourd'hui relativement bien sécurisés : les éléments de filtrage tels que routeurs, pare-feux ou DMZ évoluées se sont démocratisés, avec selon les cas antivirus et détection d'intrusion.

Cependant même si la sécurité de niveau réseau est assurée, certains aspects des couches applicatives ne sont pas forcément bien maîtrisés. Cet article se penche sur le cas particulier des fichiers, qui peuvent pénétrer sur un intranet par de nombreux moyens différents (web, e-mail, disquettes, CDROMs, ...) la plupart du temps sans réel filtrage. Or une fois qu'il est ouvert par un utilisateur, un fichier peut facilement agir sans aucun contrôle sur un intranet et mettre en jeu sa sécurité.

Nous préciserons tout d'abord ces menaces liées à l'importation de fichiers, puis nous étudierons les formats de fichiers classiques dans un environnement Windows, afin de dresser une classification suivant les problèmes de sécurité qu'ils peuvent poser. Nous aborderons ensuite les moyens disponibles pour sécuriser un réseau vis-à-vis de ces fichiers.

2 Menaces liées aux fichiers

2.1 Importation de fichiers

Tout utilisateur importe régulièrement des fichiers sur son poste de travail, son ordinateur portable ou son PC personnel. Cela peut se faire par divers moyens :

- Depuis Internet, par e-mail, HTTP, FTP ou transfert de fichiers peer-to-peer.
- Depuis des disquettes et des CDROMs de provenances variées.

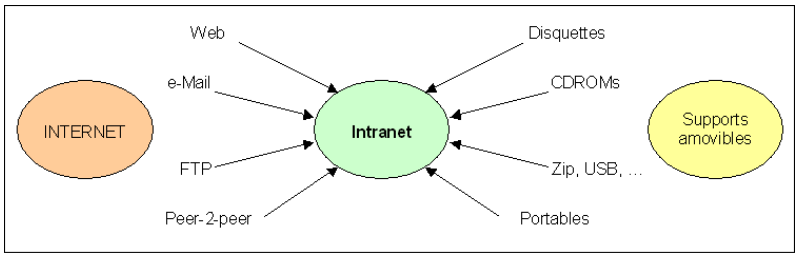


Fig. 1. Importation de fichiers sur un réseau.

– Depuis des supports amovibles comme Zip, des disques USB, ...
– Depuis un ordinateur portable connecté au réseau.

Pour les utilisateurs les plus consciencieux, ces fichiers sont d'abord vérifiés par un ou plusieurs antivirus, mis à jour toutes les semaines dans le meilleur des cas.

Ensuite, l'utilisateur ouvre les fichiers importés pour s'en servir. A ce stade il connaît uniquement le nom des fichiers, ce qui lui indique leur format à priori. En général il ne connaît pas vraiment la nature du contenu avant de les avoir ouverts. La seule certitude, c'est que les fichiers ne contiennent pas de virus connu.

Une fois ouverts, ces fichiers s'exécutent avec les même droits que l'utilisateur sur le réseau, sans que celui-ci puisse véritablement contrôler ce qui se passe. S'ils contiennent du code (exécutable, script ou macro), ce code peut agir au nom de l'utilisateur.

Un tel fichier peut donc faire beaucoup de choses, à l'insu de l'utilisateur et de l'administrateur, par exemple :

– Envoi de données confidentielles vers Internet.
– Installation d'une porte dérobée, permettant un accès réseau depuis l'extérieur.
– Installation d'un logiciel de commande à distance.
– Création d'un compte administrateur avec un mot de passe connu, si l'utilisateur en question est lui-même administrateur.
– Destruction ou falsification de données.
– Ecoute des mots de passe saisis au clavier ou circulant sur le réseau.

Dans ce cas, comment garantir l'intégrité d'un système d'information, voire sa disponibilité et sa confidentialité ?

Il existe de très nombreux formats de fichiers différents. Certains comme les exécutables posent des problèmes de sécurité évidents, car ils contiennent directement du code machine, et il est difficile de déterminer leur comportement sans les exécuter ou les désassembler.

D'autres formats peuvent dans certains cas contenir du code, comme les documents Office ou les pages HTML. Enfin, certains formats comme les images bitmap JPEG ou PNG ne présentent aucun problème de sécurité car ils ne peuvent contenir aucun code actif. Il est donc possible de classifier ces formats de fichiers, par exemple dans le but de filtrer les fichiers importés sur un réseau.

2.2 Définition d'un code malveillant

Avant d'étudier les formats de fichiers et les codes malveillants qu'ils peuvent contenir, il est nécessaire de préciser ce qu'est un code malveillant.

Un tel code agit sur le système d'information (soit sur le poste local soit sur le réseau), sans que l'utilisateur l'ait voulu ou bien qu'il y soit autorisé. Dans le domaine de la sécurité, il peut essentiellement porter atteinte au système :

- En intégrité : modification de fichier, de la base de registre, de la mémoire, d'une base de données, d'un annuaire (LDAP, Active Directory), d'autres services d'information, ...
- En disponibilité : atteinte à un service du poste local ou du réseau, afin de le rendre inaccessible ou de perturber son fonctionnement.
- En confidentialité : émission d'informations confidentielles du SI vers l'extérieur.

Un code malveillant fait donc généralement appel à des fonctions du système d'exploitation pour accéder aux fichiers, à la base de registre, au réseau, ...

La frontière entre un code normal et un code malveillant n'est cependant pas évidente. Elle est liée au contenu du système d'information et à sa politique de sécurité globale (ce que chaque utilisateur peut voir ou modifier sur le SI, les flux d'informations autorisés à sortir ou rentrer par le réseau, ...) Elle dépend aussi de l'intention de l'utilisateur au moment où il ouvre un fichier, et de sa connaissance du contenu de ce fichier.

Pour un fichier de type exécutable (binaire ou script), distinguer un code normal d'un code malveillant est quasiment impossible sans désassemblage, processus long, complexe et manuel.

Par contre dans le cas des fichiers de type " documents" qui contiennent du code, il est en théorie plus facile de distinguer les deux : en général un code non offensif ne fait que de l'affichage, du calcul ou de la fusion de données, ne modifie pas d'informations sensibles, et ne transmet pas d'informations confidentielles vers l'extérieur.

2.3 Classification des codes malveillants

Lorsqu'on ouvre un fichier, plusieurs cas peuvent se produire :

- S'il s'agit d'un fichier binaire exécutable (EXE, COM), il sera directement chargé en mémoire et exécuté.
- S'il s'agit d'un script (BAT, CMD, VBScript, JScript, ...), l'interpréteur correspondant sera appelé pour l'exécuter. Notons que certains interpréteurs ne sont pas installés par défaut sur un système Windows classique. (Perl, Python, TCL, ...).
- S'il s'agit d'un document, l'application associée sera ouverte. Si le document contient du code, celui-ci pourra être exécuté avec ou sans confirmation de l'utilisateur suivant les cas.

Du point de vue de la sécurité vis-à-vis d'un code malveillant contenu dans le fichier, nous pouvons donc distinguer plusieurs niveaux de risque, afin de classifier les formats :

1. Le format de fichier contient **toujours** du code, qui est **directement** exécuté à l'ouverture du fichier (cas des exécutables et scripts).
2. Le format contient **parfois** du code, qui peut s'exécuter **directement** (par exemple HTML avec Vbscript ou Javascript utilisant des vulnérabilités d'IE).
3. Le format contient **parfois** du code, qui ne peut s'exécuter qu'après *confirmation* de l'utilisateur (macros Word et Excel).
4. Le format contient **parfois** du code, qui ne peut s'exécuter qu'après une **action volontaire** de l'utilisateur (documents Office contenant des objets OLE).

5. Le format ne peut **jamais** contenir de code (images bitmap GIF, JPEG, PNG, ...)[1].

2.4 Associations de types de fichiers

Dans un environnement Windows, l'action effectuée lorsqu'un utilisateur ouvre un fichier dépend principalement de son extension. Chaque extension correspond à un type de fichier, auquel est associé un programme ou une ligne de commande. Par exemple, l'extension ".HTML" est par défaut associée à Internet Explorer, dont l'exécutable est "iexplore.exe". Cette association est inscrite dans la base de registre, dans la ruche "HKEY_CLASSES_ROOT". Sous Windows NT/2000/XP, les commandes FTYPE et ASSOC permettent d'afficher les associations de fichiers. Par exemple, pour afficher l'application associée aux fichiers ".VBS", on peut utiliser les deux commandes suivantes :

```
Assoc .vbs => cela donne ".vbs=VBSFile"
Ftype VBSFile => on obtient Wscript.exe
```

C'est un peu plus compliqué, mais il est aussi possible de lister toutes les extensions de fichiers associées avec un programme en utilisant les commandes suivantes dans un fichier batch. Voici un exemple pour Word :

```
ftype|find /i "word.exe" >types.tmp
FOR /F "delims==" %%t IN (types.tmp) DO (
assoc|find /i "%%t" >ext.tmp
FOR /F "delims==" %%e IN (ext.tmp) DO echo %%e
)
```

Cela montre que de nombreux types de fichiers peuvent être associés à un même programme. Dans le cas de Word, un document peut être renommé avec diverses extensions (doc, dot, dochtml, rtf, wbk) en ayant le même effet lorsqu'il est ouvert.

3 Formats de fichiers

Ce chapitre présente quelques formats de fichiers classiques dans un environnement Windows standard, et montre certains problèmes de sécurité qu'ils peuvent poser. Cette liste est loin d'être exhaustive, étant donné les milliers de formats de fichiers qui existent. Elle constitue cependant une bonne base pour définir une politique de filtrage pour une passerelle Web ou e-mail.

Elle pourrait être facilement étendue aux environnements Unix, Linux ou Macintosh en y ajoutant les formats de fichiers spécifiques à ces systèmes.

3.1 Fichiers texte ou binaires

On distingue généralement deux grands types de formats de fichiers : Les fichiers texte contiennent uniquement des caractères imprimables (codes ASCII 32 à 255 : lettres, chiffres, signes, ...) et des caractères spéciaux de mise en page (certains codes

[1] Nous considérons ici des codes malveillants directement activables

inférieurs à 32 : saut de ligne, tabulation, ...) Les octets qui les composent correspondent donc à un sous-ensemble des codes ASCII.

Les fichiers binaires sont tous les autres fichiers qui ne correspondent pas à cette définition. Ils peuvent donc contenir l'intégralité des codes ASCII (codes 0 à 255), en particulier les codes de 0 à 31.

3.2 Exécutables binaires : EXE, COM, SCR,...

Ces fichiers contiennent du code binaire directement exécutable par le processeur. Lorsqu'un utilisateur les ouvre, le code est immédiatement chargé et exécuté dans le contexte de l'utilisateur.

Ce sont donc les formats qui présentent le plus important problème de sécurité, puisque le code qui s'y trouve a un accès complet au système avec tous les droits de l'utilisateur. La plupart des virus et des chevaux de Troie sont des exécutables. Les fichiers EXE et leurs dérivés (SCR, CPL, OCX, DLL, ...) ont un format binaire, et possèdent une entête dont la structure est fixée. En particulier, les 2 premiers octets sont toujours "MZ" (initiales de Michael Zimmer, un des concepteurs de MS-DOS), sinon Windows refuse de les exécuter. Certains fichiers EXE au format MS-DOS 16 bits peuvent également débuter par "ZM". Un fichier EXE peut donc être reconnu par son nom et son contenu.

Les fichiers COM quant à eux ne possèdent pas d'entête, leur contenu est simplement du code exécutable. On peut noter qu'un fichier COM peut être conçu de manière à n'utiliser que des caractères imprimables, et avoir ainsi l'apparence d'un fichier texte. Seul le nom permet de détecter un fichier COM.

3.3 Fichiers de commande MS-DOS/Windows : BAT, CMD

Les fichiers BAT et CMD (appelés aussi "fichiers batch") sont au format texte et contiennent une liste de commandes qui sont exécutées séquentiellement lorsqu'on les ouvre. Ces commandes peuvent porter atteinte à la sécurité du système dans une certaine mesure, en lançant des outils présents sur le système.

A titre d'exemple, les 2 lignes suivantes créent un nouvel utilisateur "toto" avec un mot de passe "toto", et l'ajoutent au groupe des administrateurs (doit être lancé par un administrateur) :

```
net user toto toto /add
net localgroup administrateurs toto /add
```

3.4 PIF (Program Info File)

Un fichier PIF sert normalement à Windows pour stocker des informations sur un programme MS-DOS, afin de préciser dans quelles conditions celui-ci doit être exécuté : en fenêtre ou plein écran, s'il a besoin de mémoire étendue, etc... Il a un format binaire, et contient la ligne de commande de l'exécutable à lancer. Il peut donc exécuter une commande portant atteinte au système.

Il est important de noter que l'extension PIF est toujours masquée dans l'explorateur Windows, même si la case "masquer les extensions pour les types de fichiers connus" a été désactivée.

Un fichier .COM ou .EXE peut être renommé en .PIF. Dans la plupart des cas celui-ci sera alors directement exécutable même s'il ne possède pas la structure normale d'un fichier PIF.

3.5 Raccourcis Windows : LNK

D'une manière similaire aux fichiers PIF, les fichiers LNK contiennent une ligne de commande qui sera exécutée quand on double-clique sur le raccourci, commande qui peut porter atteinte au système.

L'extension LNK est également toujours masquée, et on peut observer dans certains cas qu'un fichier EXE ou COM renommé en LNK reste exécutable. Scripts du Windows Scripting Host : VBS, JS, VBE, JSE, WSF, WSH

Le Windows Scripting Host est un outil optionnel (installé par défaut sur les systèmes récents), qui permet d'ajouter de nouveaux langages de script à Windows. Il contient deux langages de base qui sont Vbscript et Jscript, mais il est possible d'en ajouter d'autres comme Perl ou TCL.

Ces langages sont plus évolués que les fichiers de commandes batch, et ils ont la possibilité d'accéder aux contrôles ActiveX installés sur le système, afin de contrôler toute application qui en offre la possibilité. Il est possible de manipuler les fichiers et la base de registre, mais aussi d'accéder aux logiciels comme Outlook express, Word ou Excel avec un langage de haut niveau. Par exemple le tristement célèbre virus "I Love You" était écrit en Vbscript, pour accéder facilement au carnet d'adresses d'Outlook Express et se répliquer par e-mail.

Les fichiers VBS et JS s'exécutent directement lorsqu'on les ouvre. Ils sont très rarement utiles pour les utilisateurs normaux, il est donc souvent recommandé de désactiver voire désinstaller le Windows Scripting Host pour éviter ce problème de sécurité.

Voici un exemple de code Vbscript qui crée un fichier, ce qui est une action potentiellement dangereuse :

```
Dim fso, tf
Set fso =
Wscript.CreateObject("Scripting.FileSystemObject")
Set tf = fso.CreateTextFile("c:\temp\test.txt", True)
tf.WriteLine("test VBscript.")
tf.Close()
```

Le même code, cette fois-ci en Jscript :

```
var fso, tf;
fso = new ActiveXObject("Scripting.FileSystemObject");
tf = fso.CreateTextFile("c:\\temp\\test.txt", true);
tf.WriteLine("Ceci est un test.") ;
tf.Close();
```

Les fichiers VBE et JSE correspondent respectivement à des fichiers VBS et JS qui ont été chiffrés par l'outil "Microsoft Script Encoder". Leur code n'est pas lisible en clair, cependant ils s'exécutent de la même façon.

Un fichier WSF est un fichier au format XML pouvant contenir un ou plusieurs scripts Vbscript ou Jscript. S'il est ouvert, les scripts seront exécutés séquentiellement. Voici un exemple de fichier WSF contenant 2 scripts :

```
<Job id="vbs">
<Script language="VBScript">
ok = msgbox("script 1")
</Script>
```

```
<Script language="VBScript">
ok = msgbox("script 2")
</Script>
</Job>
```

Un fichier WSH est associé à un fichier VBS ou JS : il sert à stocker des options pour ce script, comme par exemple le temps maximal d'exécution. Il est au format "INI' de Windows, et contient notamment une variable Path qui indique le chemin du script associé. Lorsqu'on lance le fichier WSH, le script associé est exécuté. Voici un exemple de fichier WSH qui fait référence à un script situé sur une autre machine du réseau :

```
[ScriptFile]
Path=\\serveur\pirate\script.vbs
[Options]
Timeout=0
DisplayLogo=0
```

3.6 Autres scripts : Perl, Python, AWK, KiXtart, PHP, TCL, bash, ...

Il existe de nombreux autres langages de script, cependant chacun nécessite que l'interpréteur correspondant soit installé sur la machine locale pour qu'il puisse être exécuté par l'utilisateur. On les retrouve rarement sur un système Windows standard.

Ces types de fichiers présentent donc un risque bien moindre que Vbscript, Jscript ou les fichiers de commandes batch. Ils doivent cependant être pris en compte si des interpréteurs sont installés sur le réseau.

3.7 Application et applets Java : fichiers .java, .class et .jar

Le langage Java est pseudo-compilé : les fichiers source ".java" ne sont pas directement interprétés comme les scripts, mais transformés en pseudo-code dans des fichiers ".class". Ces fichiers .class peuvent être alors interprétés par la machine virtuelle Java, appelée "JVM". Suivant sa conception, un fichier .class peut être soit une application Java autonome, soit une applet Java intégrée dans une page HTML.

Une applet Java s'exécute normalement dans une "sandbox" (bac à sable), et la JVM lui interdit toute action qui pourrait porter atteinte à la sécurité du système : lecture/écriture de fichiers ou du registre, accès réseau à d'autres machines que le serveur d'où l'applet a été téléchargée, etc... Il est cependant possible de réduire ces contraintes pour des applets de confiance, en utilisant la signature de code et des certificats cryptographiques. Ce système de sandbox est réputé fiable, même si certaines JVM ont souffert de quelques vulnérabilités qui permettaient à des applets d'échapper dans certains cas aux contrôles de sécurité.

Une application Java a quant à elle un accès complet au système local, tout comme les fichiers exécutables ou les scripts. Sur un système normal, l'extension .class n'est pas associée à la JVM, donc il ne se passe rien si l'utilisateur essaye d'ouvrir directement un tel fichier. Pour l'exécuter, le JRE (Java Runtime Environment) doit être installé, ce qui n'est pas le cas sur un système Windows par défaut. Il doit lancer explicitement l'application, en tapant une ligne de commande du type "java application.class". Un fichier .class pris tout seul ne pose donc pas de problème de sécurité en soi, sauf s'il a été volontairement associé au JRE, et que celui-ci est installé.

Les fichiers .jar, pour "Java Archive"», sont des archives compressées similaires aux fichiers Zip, qui contiennent plusieurs fichiers .class et d'autres fichiers nécessaires à leur fonctionnement. Si un JRE est installé, il peut prendre en charge un fichier .jar et exécuter l'application Java qui s'y trouve, mais là encore ce n'est pas le cas par défaut.

Les fichiers .java contiennent uniquement du code source Java, et ne sont pas directement exécutables. Ils ne posent donc aucun problème de sécurité.

3.8 Documents MS Office : Word, Excel, Powerpoint, Access, ...

Les documents Microsoft Office peuvent contenir des macros écrites en langage VBA (Visual Basic for Applications), qui ont souvent été utilisées par les concepteurs de virus. Le langage VBA permet de nombreuses manipulations sur les documents, ainsi que sur le système lui-même (fichiers et registre).

En nommant une macro de façon particulière, il est possible de la faire s'exécuter de façon automatique dès l'ouverture du document, voire même qu'elle s'installe dans le logiciel. (AutoOpen ou Document_Open pour Word, Auto_Open pour Excel, autoexec pour Access,...). Une macro peut aussi être associée à un objet dans le document, à un bouton, une touche du clavier ou un autre événement.

Les versions récentes d'Office intègrent quelques garde-fous pour protéger l'utilisateur contre les macros malveillantes. Par défaut, lorsqu'on ouvre un fichier Word, Excel ou Powerpoint contenant des macros, une boîte de dialogue s'ouvre pour permettre d'activer ou désactiver les macros. Cette sécurité peut cependant être ignorée voire retirée par l'utilisateur. Depuis Office 2000, il est toutefois possible de renforcer la sécurité, en n'autorisant que des macros ayant été préalablement signées. Les rapports [1] et [2] détaillent les problèmes de sécurité et les contre-mesures des documents Office 97 et 2000.

Il est important de remarquer que toutes les applications Office ne sont pas homogènes pour la gestion des macros. En particulier Access (au moins jusqu'à la version 2000) ne demande pas confirmation, et une macro nommée "autoexec" sera systématiquement exécutée à l'ouverture de la base de données. Access ne fait cependant pas partie de la version standard d'Office.

Les fichiers Office (sauf Access) possèdent une particularité : lorsqu'on renomme un fichier Office en lui donnant une extension qui n'est associée à aucun programme, par exemple ".xyz", et qu'on essaye d'ouvrir le fichier depuis l'explorateur, la bonne application (Word, Excel ou Powerpoint) sera quand même lancée. Cela n'est pas le cas avec les autres formats de fichiers, et cela ne se produit pas si on tente d'ouvrir le fichier Office renommé depuis l'invite de commande. Microsoft Office installe donc un module spécifique à l'explorateur qui examine le contenu d'un fichier lorsqu'il a une extension inconnue. En conclusion, pour reconnaître un fichier Office il ne suffit pas de regarder son nom.

3.9 Objets OLE

De nombreuses applications permettent l'utilisation du protocole OLE (Object Linking and Embedding), afin d'inclure un document de toute nature à l'intérieur d'un autre document. Il est par exemple possible d'inclure un tableau Excel dans un document Word. Il est également possible d'inclure un exécutable, qui sera lancé par un double-clic sur l'objet OLE.

La sécurité liée à l'exécution des objets OLE est assurée par chaque application. Ainsi Word demande confirmation à l'utilisateur avant de lancer un exécutable inclus, ce qui n'est pas le cas de WordPad.

Tout format de fichier supportant OLE est donc un conteneur, qui peut servir à camoufler un code malveillant.

3.10 Documents RTF

Le format RTF (Rich Text Format) est un format de Microsoft, qui permet d'exporter un document Word sous une forme "plus portable" car simplifiée. Certaines mises en forme sont supprimées, les liens hypertextes et les macros VBA sont retirées. RTF est donc souvent présenté comme un format statique, idéal pour l'échange de données car sans problème de sécurité. Il est cependant possible d'inclure un fichier exécutable dans un document RTF, sous forme d'objet OLE qui peut être activé par un double-clic. Le format RTF est donc un conteneur.

3.11 Shell Scrap : SHS

Lorsque l'on sélectionne un objet OLE dans un document, et qu'on le glisse avec la souris dans l'explorateur ou sur le bureau, on obtient un fichier SHS. Cette extension SHS est toujours masquée dans l'explorateur, et l'icône du fichier ressemble à celle d'un fichier texte. Si l'on essaye d'ouvrir ce fichier SHS, le contenu de l'objet OLE d'origine est directement ouvert, sans confirmation. S'il s'agit d'un exécutable, il est directement lancé.

3.12 HTML

Sur la plupart des systèmes Windows, les fichiers HTML sont associés à Internet Explorer. De nombreux logiciels comme Outlook Express font également appel à Internet Explorer pour afficher des contenus HTML, par exemple lorsqu'ils sont inclus dans des e-mails.

Les fichiers HTML et les e-mails au format HTML affichés par IE peuvent contenir des scripts (Vbscript ou Jscript, équivalent de Javascript), et faire appel à des objets ActiveX ou des applets Java.

Comme nous l'avons vu plus haut, les applets Java sont des fichiers .class ou .jar externes au fichier HTML, qui y fait juste référence. De plus les applets non signées s'exécutent dans un environnement sécurisé, la sandbox. A part s'il existe une vulnérabilité dans la JVM (et cela est déjà arrivé), un fichier HTML ne peut utiliser une applet Java comme code malveillant pour accéder au système.

Pour les objets ActiveX et les scripts, IE utilise un système de zones de sécurité, afin d'appliquer des restrictions plus ou moins fortes suivant la provenance des pages HTML. Pour un fichier HTML, lorsqu'un script tente d'effectuer une action pour accéder au disque, à la base de registre ou au réseau, ou lorsque la page HTML essaye de charger un objet ActiveX non signé, l'utilisateur reçoit une demande de confirmation, qui lui permet de bloquer le script ou le chargement de l'objet ActiveX. Internet Explorer a cependant un lourd passé : de nombreuses vulnérabilités ont permis de contourner le système de zones de sécurité, et ces failles ont été mises à profit dans la plupart des virus récents pour exécuter du code malveillant sans confirmation. Comme les anciennes versions d'Internet Explorer sont toujours largement répandues, il est nécessaire de

prendre en compte le risque d'exécution de code malveillant depuis les pages HTML, malgré les mesures de protection.

Les scripts peuvent être placés à de nombreux endroits dans un code HTML, notamment :

- Entre des balises `<SCRIPT>` et `</SCRIPT>`
  (exécution dès le chargement de la page).
- Dans l'URL d'un lien hypertexte : `<A HREF="javascript:...">`
  (exécution quand l'utilisateur clique sur le lien).
- Dans l'événement "Onload" de la zone Body : `<BODY ONLOAD="javascript:...">`
  (exécution dès le chargement de la page).
- Dans de nombreux évènements associés à des objets, par exemple :
  `<IMG SRC="une_image.gif" ONMOUSEOVER="javascript:...">`
  (exécution quand l'événement apparaît).

Le langage Javascript, développé au départ par Netscape, a des fonctionnalités limitées à l'affichage, au calcul, à la gestion des cookies et au contrôle du navigateur. Vis-à-vis du système, il ne pose donc pas à priori de problème de sécurité.

Par contre le langage Jscript, qui possède une syntaxe similaire à Javascript, offre des fonctions supplémentaires d'accès au système de fichiers, à la base de registre et aux contrôles ActiveX installés sur le système. Tout code Javascript est exécuté par Internet Explorer en tant que Jscript, il est donc possible d'écrire un code malveillant dans un simple lien `"javascript:..."`. Le langage Vbscript possède une syntaxe différente de Jscript (dérivée de Visual Basic), cependant les fonctionnalités et les risques sont identiques.

3.13 XHTML

XHTML est une nouvelle version du langage HTML, dont la syntaxe a été corrigée pour être conforme au standard XML. Cette syntaxe est plus stricte, et interdit certaines formes d'écritures pouvant mener à l'exploitation de vulnérabilités ou a du camouflage de code.

Les possibilités d'inclusion de scripts sont cependant les mêmes que pour HTML, il convient donc de traiter les fichiers XHTML avec autant de précautions.

3.14 XML

XML est un format conçu pour contenir des données, structurées sous forme d'arbre. A la différence de HTML, un fichier XML ne contient que des données brutes et pas de mise en forme. L'interprétation de ces données dépend de chaque application, qui peut définir son propre schéma (structure de données). Il est donc impossible de déterminer dans l'absolu si un fichier XML contient du code malveillant, à moins de savoir exactement comment les données seront exploitées par l'application associée.

Par défaut sur un système Windows, l'extension ".XML" est associée à Internet Explorer. Lorsqu'on ouvre un fichier XML brut, IE affiche simplement les données sous forme d'arbre à l'écran. Il n'y a pas de possibilité d'exécution de code dans ce cas.

Cependant XML introduit la notion de feuilles de style XSL. Une feuille de style XSL décrit les règles de transformation à appliquer pour afficher les données du fichier XML, par exemple pour obtenir une page HTML avec mise en forme. Si un fichier XML possède dans son entête une référence à une feuille de style XSL, et que le fichier XSL en question est accessible (soit en local, soit par le réseau), alors Internet Explorer

applique automatiquement la transformation, et le résultat est affiché. Dans ce cas, le résultat peut inclure du code malveillant, comme toute page HTML. De plus, il est possible d'inclure la feuille de style XSL à l'intérieur du fichier XML, ainsi un simple fichier XML peut très bien contenir du code malveillant HTML, qui s'exécutera à l'ouverture dans IE. Voici un très court exemple :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="#xsl" ?>
<xsl:stylesheet id="xsl"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
   <xsl:template match="/">
     <HTML><BODY><SCRIPT language="javascript">
       alert('Ceci est un jscript contenu dans XML.');
     </SCRIPT></BODY></HTML>
   </xsl:template>
</xsl:stylesheet>
```

Les fichiers XML posent donc les mêmes problèmes de sécurité que les fichiers HTML, avec des possibilités supplémentaires de camouflage à cause du mécanisme des feuilles de style XSL.

3.15 MHT (MIME HTML)

Les fichiers MHT correspondent au format MHTML défini dans [3]. Il s'agit d'un format similaire aux e-mails avec pièces jointes, permettant d'encapsuler une page HTML et les autres fichiers qui la composent (images, sons, ...). Il est donc possible d'y inclure un nombre quelconque de fichiers, de tous formats. Ces fichiers peuvent apparaître en clair, ou être codés suivant le standard MIME (Base64 ou quoted-printable, par exemple).

C'est notamment le format utilisé par Internet Explorer lorsqu'on sauvegarde une page HTML en tant qu' "Archive Web".

Par défaut sur un système Windows, l'extension ".MHT" est associée à Internet Explorer. Lorsqu'on ouvre un fichier MHT, IE affiche directement le premier fichier contenu, qui est normalement une page HTML.

Le format MHT est donc un conteneur, dans lequel peuvent être cachés des scripts ou tout autre fichier codés au format Base64.

Voici un exemple de fichier MHT, lançant un simple script non codé :

```
MIME-Version: 1.0
Content-Type: text/html
Content-Transfer-Encoding: 7bit

<body onload="alert('Ceci est un script dans une page
HTML, dans un fichier MHT.')">
```

3.16 Adobe Acrobat : PDF

Le format PDF a été créé par Adobe pour produire des documents lisibles sur de nombreux systèmes, en conservant une mise en forme homogène à l'écran et à

l'impression. Le logiciel d'affichage Acrobat Reader est fourni gratuitement, et il est aujourd'hui extrêmement répandu, PDF est donc devenu un standard.

A l'instar de RTF, PDF est souvent considéré comme un format sûr (plus sûr que Word et HTML), permettant d'échanger des documents riches sans risque d'infection par un virus.

Ce format possède cependant une fonction méconnue, qui permet à un document PDF d'ouvrir un fichier externe (y compris un exécutable) lorsqu'un événement se produit, par exemple à l'ouverture ou à la fermeture du fichier PDF. Avant d'exécuter une telle action qui pourrait porter atteinte au système, Acrobat Reader demande confirmation à l'utilisateur. Cette sécurité peut être considérée comme faible, car l'action par défaut est d'accepter l'exécution, et l'utilisateur peut choisir d'ignorer toutes les confirmations suivantes (tant qu'Acrobat Reader reste en mémoire).

Un fichier PDF peut également contenir des fichiers attachés, et ceux-ci peuvent être lancés par un double-clic de l'utilisateur après confirmation, tout comme les objets OLE dans Word. Ces fichiers attachés ne sont cependant accessibles que dans la version complète d'Acrobat, et non dans Acrobat Reader. Ce logiciel étant beaucoup moins répandu, cette fonctionnalité est difficilement utilisable pour un code malveillant.

Le document [4] fourni par Adobe détaille ces deux problèmes de sécurité, et apporte quelques éléments de sécurisation.

Le format PDF doit donc également être considéré avec précaution, même si le risque est à priori moins grand que pour les formats vus précédemment.

3.17 Synthèse

Le tableau suivant permet de synthétiser les caractéristiques des différents formats abordés du point de vue de l'exécution de code malveillant. La colonne risque correspond aux 5 niveaux de classification des codes malveillants listés en début de chapitre :

1. Le format de fichier contient **toujours** du code, qui est **directement** exécuté à l'ouverture du fichier.
2. Le format contient **parfois** du code, qui peut s'exécuter **directement**.
3. Le format contient **parfois** du code, qui ne peut s'exécuter qu'après **confirmation** de l'utilisateur.
4. Le format contient **parfois** du code, qui ne peut s'exécuter qu'après une **action volontaire** de l'utilisateur.
5. Le format ne peut **jamais** contenir de code.

4 Solutions

Il n'existe pas aujourd'hui de solution unique pour se protéger globalement du risque d'exécution de code malveillant depuis des fichiers. Il est nécessaire de mettre en place de nombreuses mesures pour améliorer la sécurité de chaque système. Les paragraphes suivant listent les principales solutions envisageables à l'heure actuelle.

4.1 Sécurisation du poste de travail

La sécurisation de chaque poste de travail est la première mesure technique à mettre en place. Elle implique la sécurisation du système d'exploitation et des logiciels installés, ainsi que leur mise à jour régulière.

Format	Extensions	Risque	Conteneur	Type
exécutable binaire	EXE, SCR	1	oui	binaire
exécutable binaire	COM	1	oui	binaire/texte
fichier batch	BAT,CMD	1		texte
raccourci MS-DOS	PIF	1		binaire
raccourci Windows	LNK	1		binaire
scripts WSH	VBS, JS, VBE, JSE, WSF, WSH	1		texte
code source Java	JAVA	5		texte
classe Java	CLASS	4		binaire
archive Java	JAR	4	oui	binaire
Word	DOC, DOT, WBK, DOCHTML	3	oui	binaire
Excel	XLS, XL?, SLK, XLSHTML,	3	oui	binaire
Powerpoint	PPT, POT, PPS, PPA, PWZ, PPTHTML, POTHTML	3	oui	binaire
Access	MDB, MD?, MA?, MDBHTML	1	oui	binaire
RTF	RTF	4	oui	texte
Shell Scrap	SHS	1	oui	binaire
HTML	HTML, HTM, ...	2		texte
XHTML	XHTML, XHT	2		texte
XML	XML, XSL	2		texte
MHTML	MHT, MHTML	2	oui	texte
Adobe Acrobat	PDF	3	oui	texte

4.2 Mise à jour du système et des logiciels

Le passé a montré que la plupart des virus et des attaques mettant en jeu du code malveillant tiraient parti de vulnérabilité dans le système d'exploitation Windows et notamment dans son navigateur Internet Explorer. Il est donc crucial de mettre à jour chaque système d'exploitation pour corriger les failles connues. Cela nécessite un suivi régulier et rigoureux, qui prend beaucoup de temps.

C'est également le cas de la suite Microsoft Office et des autres logiciels qui sont installés sur la plupart des postes clients, comme Acrobat Reader.

4.3 Sécurisation du système et des logiciels

Les systèmes Windows et les logiciels sont rarement installés avec une sécurité optimale par défaut. Il est donc important d'appliquer les recommandations pour améliorer la sécurité de l'ensemble, et combler certaines failles. De nombreuses check-lists de sécurité sont disponibles sur Internet pour mener à bien ce travail.

4.4 Internet Explorer et Outlook Express

La sécurisation d'IE et d'OE passe essentiellement par le renforcement de la sécurité des zones d'Internet Explorer. Le site [5] propose une sécurisation de ces deux logiciels.

4.5 Acrobat Reader

Certaines clés de registre peuvent être positionnées pour empêcher l'exécution automatique de fichiers externes, comme le montre le document [4]. Ces clés varient suivant la version d'Acrobat ou d'Acrobat Reader installée, par exemple :

- Pour Acrobat Reader 4.0 :
  ```
  HKEY_CURRENT_USER\Software\Adobe
  \Acrobat Reader\4.0\AdobeViewer\SecureOpenFile = 1
  ```
- Pour Acrobat Reader 5.0 :
  ```
  HKEY_CURRENT_USER\Software\Adobe
  \Acrobat Reader\5.0\AdobeViewer\SecureOpenFile = 1
  ```
- Pour Acrobat 5.0 :
  ```
  HKEY_CURRENT_USER\Software\Adobe
  \Adobe Acrobat\5.0\AdobeViewer\SecureOpenFile = 1
  ```

4.6 Office

La sécurisation d'Office 2000 passe avant tout par la mise en place du niveau de sécurité pour les macros, afin de n'autoriser que les macros signées (menu Outils/Macro/Sécurité).

Les rapports [1] et [2] apportent une liste de contre-mesures détaillées pour améliorer la sécurité d'Office 97 et 2000.

4.7 Antivirus

Un antivirus mis à jour très régulièrement (toutes les semaines pour les postes clients) est évidemment une mesure indispensable pour se protéger des codes malveillants connus. Certains antivirus intègrent une méthode de détection heuristique capable de détecter certains codes ayant un comportement de virus, cependant cette approche ne suffit pas pour détecter un code malveillant dans le cas général.

4.8 Passerelle de filtrage réseau : e-Mail, Web, FTP, ...

Il existe de nombreuses solutions de filtrage réseau à base de passerelles ou proxies, afin de s'assurer qu'aucun contenu actif ne peut pénétrer par e-mail, consultation web ou transfert de fichiers FTP.

La plupart des solutions proposent une analyse antivirus à la volée ainsi qu'un filtrage par type de fichier importé. Cependant ce filtrage est souvent basé uniquement sur le nom des fichiers ou sur leur type MIME déclaré. Or nous avons vu qu'un type de fichier peut souvent être renommé de diverses manières, ou même que certains types peuvent avoir une extension quelconque tout en restant actifs (documents Office). De plus, de nombreux formats de fichiers jouent le rôle de conteneurs, et peuvent donc camoufler du code malveillant.

Un filtrage de fichiers efficace nécessite donc une analyse par contenu et par nom. Cette analyse doit être récursive afin de traiter les formats conteneurs.

4.9 Suppression de code, conversion de format

Pour les codes inclus dans des documents Office ou HTML, il est en théorie possible de distinguer un code inoffensif d'un code potentiellement dangereux, car le langage de programmation est connu, et les commandes "à risque" qui accèdent au système de fichier, à la base de registre ou au réseau peuvent être identifiées. Dans la pratique, les outils qui proposent ce type de détection possèdent cependant des limites, dues à l'évolution des langages de script et aux possibilités de camouflage de code.

Certains produits de filtrage (comme eSafe ou MimeSweeper) et certains antivirus (comme F-Prot) proposent la suppression complète des portions de code dans les formats connus, en particulier HTML et les documents Office. Cette suppression n'est généralement pas parfaite et peut parfois être contournée par des techniques de camouflage, cependant il s'agit d'une fonction intéressante pour assurer un service tout en évitant de nombreux risques.

Pour supprimer le code de ces formats de fichiers, il est également possible de convertir les fichiers dans un autre format compatible présentant moins de risques, comme RTF ou PDF, ou encore le format texte pur. Cette conversion est cependant souvent très imparfaite (perte de mise en forme), et très contraignante pour l'utilisateur.

4.10 Supports amovibles

Pour un système très sensible, il est parfois possible de neutraliser tous les périphériques amovibles : lecteur de disquette, CDROM, ports USB, série et parallèle, SCSI, ... Cependant ce n'est pas envisageable dans un cadre général. Certaines solutions logicielles existent, par exemple pour limiter l'accès aux supports amovibles à l'administrateur.

Une solution intermédiaire serait sans doute plus adaptée, par exemple pour assurer un filtrage des types de fichiers importés depuis les lecteurs amovibles. Cela nécessiterait un outil fonctionnant sur le principe des "antivirus temps réel", qui vérifierait chaque fichier avant ouverture, avec suppression éventuelle des codes exécutables.

4.11 Contrôle des actions au niveau système

Pour obtenir un niveau de sécurité plus élevé, et empêcher tout fichier importé d'exécuter une action malveillante, l'idéal serait à priori de contrôler chaque action effectuée au niveau système, en distinguant les fichiers suivant leur provenance et la confiance accordée par l'utilisateur ou l'administrateur. Cela pourrait par exemple passer par la signature de code ou des fichiers de données, en associant à chaque fichier des droits.

Il existe des produits comme Tiny Personal Firewall qui associent un pare-feu personnel (filtrage réseau par application) à une sandbox (environnement restreint) pour l'exécution des programmes. Chaque exécutable du système est reconnu de façon unique par une empreinte de type MD5, et il est associé à un profil d'exécution qui lui donne accès à tout ou partie des fonctions du système. Il est ainsi possible de laisser les outils et applications de confiance fonctionner normalement, tout en restreignant les autres exécutables. Dès qu'un exécutable inconnu est lancé par l'utilisateur, celui-ci est prévenu par un message d'alerte et il peut lui attribuer un profil ou bien le bloquer.

Cette approche est pour l'instant relativement nouvelle dans l'environnement Windows, et les outils actuels sont difficilement déployables dans un cadre général. L'attribution optimale des profils est une tâche complexe, et l'utilisateur peut facilement

trouver le système trop contraignant s'il est mal configuré. De plus, le filtrage est exclusivement réservé aux fichiers exécutables, il ne couvre donc pas toutes les menaces abordées dans cet article.

4.12 Contrôle d'intégrité régulier

Il existe de nombreux logiciels capables de vérifier l'intégrité d'un système. Cette protection se limite généralement aux fichiers qui ne doivent pas être modifiés, en particulier les exécutables du système d'exploitation et des applications, ainsi que les fichiers de configuration sensibles.

Il est possible de mettre à profit ces outils pour détecter toute apparition de fichier exécutable ou de script dans des zones de disque où cela est anormal. Certains outils proposent également la vérification de clés de registre, ce qui permet de détecter les codes malveillants utilisant les clés de type "Run" pour être activés à chaque démarrage du système.

Le contrôle d'intégrité doit cependant être considéré comme une mesure complémentaire, surtout destinée à détecter les symptômes d'une attaque réussie.

4.13 Mesures organisationnelles

Des mesures organisationnelles sont évidemment nécessaires, car le risque d'exécution de code malveillant provient avant tout de l'utilisateur qui importe les fichiers ou qui les ouvre. Voici quelques mesures utiles :

- Sensibilisation régulière des utilisateurs, si possible avec démonstration concrète de code malveillant dans des documents standards.
- Sensibilisation renforcée des administrateurs : ils doivent éviter au maximum d'ouvrir des fichiers importés lorsqu'ils sont connectés sous un compte d'administration. Ils doivent donc utiliser au maximum un compte utilisateur nominatif non privilégié.
- Interdiction d'ouvrir des fichiers importés ou des e-mails sur un serveur.
- Tout fichier importé doit être vérifié par au moins un antivirus à jour.
- Tout fichier suspect doit être signalé à la personne en charge de la sécurité du réseau.

5 Conclusion

La menace liée aux fichiers importés sur un intranet est réelle, que ceux-ci proviennent des connexions réseau avec l'extérieur ou de supports amovibles. En effet de nombreux formats de fichiers peuvent contenir du code malveillant, qui s'exécute avec les droits de l'utilisateur qui les ouvre.

Cependant cette menace est habituellement mal ou sous-évaluée, et les outils disponibles sont aujourd'hui peu adaptés pour assurer la sécurité globale d'un réseau d'entreprise vis-à-vis des fichiers importés. Il reste donc beaucoup de recherche à mener dans ce domaine.

Il existe cependant de nombreuses mesures techniques et organisationnelles qui peuvent être mises en place afin d'améliorer la sécurité et sensibiliser les utilisateurs au problème du code malveillant dans les fichiers.

Références

1. *Microsoft Office 97 Executable Content Security Risks and Countermeasures*, NSA. *http ://www.nsa.gov/snac/emailexec/guides/eec-3.pdf*
2. *Microsoft Office 2000 Executable Content Security Risks and Countermeasures*, NSA. *http ://www.nsa.gov/snac/emailexec/guides/eec-4.pdf*
3. RFC 2557, *MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)*
4. Carl Orthlieb, *Adobe Acrobat Security Issues : The Open File Action and File Attachment Annotations*, *http ://www.adobe.com/products/acrobat/pdfs/OpenFilenAttach.pdf*
5. Patrick Chambet, *Sécurisation d'Internet Explorer et d'Outlook Express*, *http ://www.chambet.com/publications/ie-oe-security/index.html*

Le Spyware dans Windows XP

Nicolas Ruff

EdelWeb
Nicolas.ruff@edelweb.fr

1 Introduction

1.1 Contexte

Le SpyWare, en lien avec le respect de la vie privée, est un sujet à la mode en ce moment - tout particulièrement dans ce contexte de lutte contre le terrorisme où nos libertés individuelles sont menacées.

En France des lois de plus en plus restrictives sont adoptées (Loi Sécurité Quotidienne, Loi sur la Confiance dans l'Economie Numérique), dont certaines autorisent des formes d'espionnage "légal" (écoute, conservation de traces).

Le géant Microsoft fait naître de nombreuses angoisses compte-tenu de l'hégémonie du système d'exploitation Windows et de la facilité avec laquelle celui-ci pourrait se transformer en instrument d'espionnage mondial (si ça n'est déjà le cas).

De nombreuses alertes - dont la médiatisation n'est pas toujours proportionnelle au risque réel (ex. "supercookie" Windows Media), ainsi que les nouveautés de Windows XP telles que le "Product Activation", ont contribué à diminuer la confiance des utilisateurs finaux que nous sommes. Sans parler des initiatives TCPA et Palladium, dont il ne sera pas question ici, mais qui préfigurent un avenir "connecté".

1.2 Objectif

L'objectif de cette présentation est de faire un tour d'horizon des principales nouveautés de Windows XP dont la fonction est de près ou de loin dépendante d'une communication avec un serveur Web Microsoft. Une analyse factuelle des sites contactés, des informations échangées et des possibilités d'exploitation de ces informations par Microsoft sera réalisée –lorsque cela est possible.

Cette présentation souhaite rester objective sans alimenter de fantasmes, tout en mentionnant les zones d'ombre qui persistent des mécanismes échappant à l'analyse. Enfin des solutions concrètes et applicables pour limiter les flux à destination d'Internet seront proposées en guise de conclusion.

1.3 Moyens

Le document de référence de cette étude est un "white paper" Microsoft de près de 200 pages dédié au thème des communications avec Internet :

"Using Windows XP Pro SP1 in a Managed Environment : Controlling Communication with the Internet"

Ce document a été décortiqué et validé sur différentes plateformes de test au sein du laboratoire R&D de EdelWeb. Les résultats de ces travaux, ainsi que d'autres menés par des sociétés tierces (tels que les auteurs des outils "Ad-Aware" ou "XP AntiSpy") sont présentés ici.

2 Noyau

3 Installation

Dès l'installation, Windows recherche une connexion réseau afin d'accéder aux sites de mise à jour et d'activation produit.

Windows dispose pour cela de 2 méthodes :

- Configuration manuelle
- Autodétection de la configuration réseau, selon les méthodes décrites ci-dessous

Méthode 1 : via des options DHCP

Si la configuration de l'interface réseau a été obtenue depuis un serveur DHCP, Windows envoie un message DHCP "Inform" à ce serveur pour lui demander les options suivantes :

- Informations envoyées
  - 12 Hostname = "machine_name"
  - 53 Message Type = "Inform"
  - 60 Vendor = "MSFT 5.0"
  - 61 Client ID = Ethernet + MAC Address
  - 55 Parameters (cf. ci-dessous)
- Informations demandées
  - 1 : subnet
  - 3 : router
  - 6 : DNS
  - 12 : hostname
  - 15 : domain name
  - 31 : router discovery
  - 33 : static route
  - 43 [paramètres spécifiques au vendeur] Non documenté
  - 44, 46, 47 : NetBT configuration
  - 249 [extension Microsoft] Non documenté
  - 252 [extension Microsoft] Option WPAD (cf. Q296591)

Méthode 2 : via une requête DNS

Windows utilise l'extension WPAD (*Web Proxy Auto Discovery*) : il effectue une résolution DNS sur le nom wpad.¡domaine¿, où ¡domaine¿ instancie tous les domaines connus du client.

Remarque : d'après la documentation disponible, les mécanismes d'autoconfiguration ont été améliorés dans Windows 2003.

3.1 Activation

Présentation générale Il ne faut pas confondre "activation" et "enregistrement" du produit.

- L'activation permet d'obtenir une licence définitive à partir de la clé de licence collée sur le boîtier du CD. Elle est obligatoire sous 90 jours. Les clés de licence dites "en volume" outrepassent cette fonction. L'objectif principal annoncé par Microsoft est la lutte contre le piratage.

– L'enregistrement permet de déclarer auprès de Microsoft l'installation de logiciels. Cette opération est facultative.

Détails techniques Le processus d'activation du produit est complexe et très bien documenté par le site `http://www.licenturion.com/xp/`. Pour résumer, la clé de licence temporaire est utilisée en conjonction avec les paramètres ci-dessous et d'autres éléments tels qu'une clé Microsoft et un aléa pour générer un ID d'installation. A cet ID correspond une clé de licence définitive qui ne peut être calculée que par le support Microsoft. Les algorithmes utilisés (MD5 et SHA-1 entre autres) sont à sens unique et ne permettent pas de reconstituer les informations initiales.

Paramètres matériels pris en compte :

– Numéro de série de la partition système
– Adresse MAC de l'interface réseau
– Chaîne d'identification du CD-ROM
– Chaîne d'identification de la carte graphique
– CPU ID
– Chaîne d'identification du disque dur
– Chaîne d'identification de la carte SCSI
– Chaîne d'identification du contrôleur IDE
– Modèle de CPU
– RAM installée (en puissances de 32 Mo)
– Système amovible ou non

Dès lors que plus de 3 des paramètres ci-dessus sont modifiés, le produit doit être réactivé.

L'outil d'enregistrement est MSOOBE.EXE (`%WinDir%\System32\OOBE`) - pour "*Out Of the Box Experience*". Celui-ci supporte 2 méthodes de transmission : le téléphone ou Internet. Dans ce dernier cas, c'est le site `http://wpa.one.microsoft.com/` qui est contacté.

Le stockage de la clé de licence défintive s'effectue dans :

```
%WinDir%\System32\wpa.dbl
HKLM\SYSTEM\WPA
```

Un journal des opérations se trouve dans `%WinDir%\setuplog.txt`

Pour plus d'informations, se reporter au site `http://www.microsoft.com/piracy/basics/activation/`

A titre indicatif l'enregistrement du produit s'effectue via le site `http://reg.register.microsoft.akadns.net/` (noter le DNS dynamique!).

3.2 Explorer.exe

"Explorer" est l'interface graphique utilisateur par défaut. Il existe de nombreuses interactions entre Explorer et le monde extérieur dans la configuration par défaut :

– Les raccourcis réseau ("Favoris réseau") et Web sont vérifiés à l'ouverture de session et lors de tout rafraîchissement (ex. touche F5).
– Option (active par défaut) "rechercher automatiquement les dossiers et imprimantes partagées"
– Option "cette copie de Windows est-elle légale?" affichant une page du site Microsoft

L'outil "assistant recherche" du menu démarrer est particulièrement représentatif de ce point de vue :

- Son interface est complètement Web (HTML + VBE)
- Le site de recherche par défaut est `http://ie.search.msn.com/` (personnalisable - il est possible d'utiliser Nomade, etc.)
- Le répertoire de stockage des fichiers de l'application est `%WinDir%\srchasst`
  - Ceux-ci se mettent à jour automatiquement depuis Internet à chaque usage de la fonction !
- Paramétrable par la clé de base de registre "*Use Search Asst*".
- Le délai de conservation des journaux côté serveur (recherches effectuées) annoncé par Microsoft est de 1 an.
- Ce composant affiche de la pub ...

Pour mémoire, il semble judicieux de rappeler ici que l'interface Explorer souffre de nombreux problèmes de sécurité autres que les accès Internet :

- L'interface Explorer par défaut est une page Web, personnalisable à l'aide du modèle "folder.htt". Elle partage son moteur de rendu (et ses vulnérabilités) avec Internet Explorer.
- Certaines extensions de fichier ne sont pas affichées par défaut même si l'option globale est activée (ex. ".SHS", "`.<GUID>`")
- Il est possinble d'exécuter des fichiers indépendamment de leur extension via la ligne de commande (ex. renommer un .EXE en .PDF permet toujours de le lancer via un CMD).
- L'ordre de recherche par défaut des exécutables est dangereux puisqu'il inclut le répertoire courant avant les répertoires système
  - Pour les DLLs, ce comportement est paramétrable par la clé de base de registre "*SafeDllSearchMode*".
- Etc...

Toutes ces failles étant ou pouvant être exploitées par du code malveillant (ex. virus).

3.3 Aide et support

La fonction d'aide et support dispose elle aussi d'une interface complètement Web, stockée dans le répertoire :

"`%WinDir%\PCHealth\HelpCtr\`".

Certaines sections proviennent directement d'Internet :

- Rubrique "*Le saviez-vous ?*", mise en cache dans le répertoire : "`%WinDir%\PCHealth\HelpCtr\Config\`" (fichiers "*NewsSet.xml*" et "`\News\NewsVer.xml`"), et issue des liens
  - `http://go.microsoft.com/fwlink/?LinkID=11`
  - `http://windows.microsoft.com/windowsxp/newsver.xml`
- Recherche dans MSDN (qui transmet la langue et le type de produit installé)

Ce fonctionnement est paramétrable via la clé de base de registre "*Headlines*" et les "*Options de recherche*".

Enfin une des fonctions les plus impressionnantes est la possibilité de prise en main du poste par Microsoft via la fonction "Remote Assistance". Pour cela Microsoft dispose du compte préinstallé "*SUPPORT_388945a0*", membre du groupe "*HelpServicesGroups*". Il suffit à l'utilisateur de se connecter au site `https://webresponse.one.microsoft.com/`.

A noter que cette fonction est extensible par les OEM.

3.4 WindowsUpdate

WindowsUpdate est le site de distribution des mises à jour logicielles (incluant les mises à jour de sécurité) pour les produits Windows / Internet Explorer. Ce site peut être consulté manuellement, via un raccourci du menu démarrer, ou via les fonctions de type "*Dynamic Update*", "*Auto Update*", etc. – les mécanismes sous-jacents sont identiques.

Attention : ce site ne diffuse aucune mise à jour pour d'autres produits tels que Office, SQL, Exchange, etc. – c'est une des raisons de la propagation du ver SQL/Slammer.

Le fonctionnement de ce site est relativement complexe et repose sur plusieurs composants :

- Un contrôle ActiveX signé : "*UpdateClass*" (de taille ≈ 100 Ko)
  - `http://v4.windowsupdate.microsoft.com/CAB/x86/unicode/iuctl.CAB`
- Un site Web
  - `http://www.windowsupdate.com/` (alias)
  - `http://windowsupdate.microsoft.com/`
- Un moteur de traitement partagé entre client et serveur
  - Script côté client
  - Fichier XML côté serveur
    - `https://v4.windowsupdate.microsoft.com/getmanifest.asp`
    - `https://v4.windowsupdate.microsoft.com/consumerdrivers/getmanifest.asp`
- Des répertoires de travail côté client
  - `C:\Program Files\WindowsUpdate`
  - `C:\WUTemp`
- Des journaux d'installation
  - Historique des installations IUHIST.XML
  - `%WinDir%\Windows Update.log`
  - `%WinDir%\Setupapi.log` (journal global des installations)

Une analyse des échanges réseau confirme qu'aucune information nominative ou sensible ne transite sur le réseau. On notera toutefois les points suivants :

- WindowsUpdate, ainsi que d'autres fonctions système telles que le rapport d'erreur, exploitent le service "*Upload Manager*", qui effectue des transferts en arrière plan. Ce service présente des zones d'ombre car :
  - Il effectue des transferts de manière asynchrone donc difficilement analysables
  - Il est démarré par SVCHOST donc difficile à filtrer même avec un firewall personnel
  - Son API n'est pas documentée
- Les communications avec le serveur n'utilisent pas HTTPS pour le téléchargement des correctifs. Ceux-ci sont signés, mais les mécanismes de vérification de la signature ne sont pas connus (serait-il possible de présenter n'importe quel exécutable signé ou celui-ci doit-il provenir de Microsoft ?)

A titre anecdotique, on notera les points suivants :

- Il existe une adresse IP "en dur" dans le contrôle ActiveX : 207.46.226.17. Cette adresse ne correspond pas à une machine accessible depuis Internet.
- Commentaire tiré d'une page WindowsUpdate
  - *// Do not Remove this "else". Bug 16783 (If u remove this else, then for IE5 when we redirect to another page in above line, then it flashes an Action Cancelled page for a sec)*

3.5 Rapport d'erreur

Il existe 2 types de rapport d'erreur : le rapport d'erreur applicatif et le rapport d'erreur système (noyau). Dans les 2 cas Windows propose de remonter l'information sur le site Microsoft `http://watson.microsoft.com/`.

Les informations suivantes figurent dans un rapport d'erreur applicatif :

- Adresse IP (lors de la transmission)
- Product ID
- Minidump (documenté dans le Platform SDK)
  - Threads (informations "standard" et "étendues").
  - Modules (chargés et déchargés).
  - Données d'allocation mémoire (32 et 64 bits).
  - Gestionnaire d'exceptions.
  - Informations système.
  - Commentaires.
  - Handles.
  - Fonctions exportées.

  Windows 2003 Données du processus (ID, temps d'exécution).
  - Champs "réservés" (inutilisés dans Windows XP)

Les informations suivantes figurent dans un rapport d'erreur système :

- Adresse IP (lors de la transmission).
- Informations matérielles.
- Processeurs, RAM.
- Drivers installés et drivers chargés (verbeux).
- Informations logicielles (OS, version, langue).
- Message d'erreur.
- Contexte d'exécution.
- Pile noyau.

Ces informations sont transmises au reboot suivant à l'aide du service "*Upload Manager*".

Dans les deux cas aucune information sensible ne transite volontairement dans le rapport d'erreur, toutefois les "dumps" mémoire peuvent fort bien contenir des bribes de documents, des clés ou des mots de passe. Il est regrettable que l'utilisateur ne puisse pas sélectionner individuellement les informations qu'il souhaite transmettre, comme c'est le cas avec le service de rapport d'erreur de Netscape par exemple.

Le composant responsable de la génération du rapport est DrWatson :

`%WinDir%\System32\dwwin.exe)`.

La transmission s'effectue à l'aide des protocoles HTTP et HTTPS (pour le contenu du rapport uniquement).

Il est intéressant de connaître le site "*Corporate Error Reporting*" (`http://oca.microsoft.com/`), qui permet la consultation des rapports transmis pendant 180 jours.

3.6 Authentification Passport

"*Passport*" est une solution de SSO à l'échelle du Web, développée par Microsoft et intégrée nativement aux dernières versions de Windows (XP, 2003), Internet Explorer (6.0) et IIS (6.0). Cette authentification est d'ores et déjà indispensable pour accéder aux services suivants :

- Services Microsoft (MSDN, Beta Previews, etc.).
- "Spin-offs" Microsoft : MSN, Hotmail, Messenger ...
- Sites partenaires (liste complète sur `http://www.passport.net/`)

Le seul concurrent direct de cette initiative est le projet Liberty Alliance (`http://www.projectliberty.org/`) qui n'en est qu'à ses débuts.

Le principe de fonctionnement repose sur un serveur central d'authentification (base de données utilisateurs) et l'utilisation de cookies sur le poste client. Les sites centraux sont :

- `http://register.passport.net/`
- `https://login.passport.com/`
- `https://nexus.passport.com/` (remarque : "*nexus*" est un terme utilisé dans Palladium)

Les risques pour la confidentialité des données nominatives fournies au système Passport (allant jusqu'à des numéros de carte bleue) sont très importants :

- La base de données d'informations nominatives est partagée entre tous les partenaires - l'utilisateur est censé conserver un niveau de contrôle sur la diffusion de l'information mais rien ne lui garantit que ses "préférences" sont respectées par le système central.
- Ce système permet un "tracking" à des fins marketing de l'activité utilisateur.
- Les risques de vol d'information par des tiers malveillants sont réels, puisque des vulnérabilités ont déjà identifiées par le passé : ex. cookie "PPTProf=..." contenant des informations en clair.

Pour plus d'informations on se reportera au Passport SDK. A titre d'information sur les vulnérabilités :

`http://www.tcpdemux.com/products/netintercept/casestudies/passport`

3.7 Login Web

Ce chapitre couvre deux modes très différents de gestion des authentifiant par Internet Explorer :

- L'authentification native HTTP (de type ".htaccess").
- L'authentification applicative (formulaires).

Authentification native Les modes d'authentification supportés par IE 6.0 SP1 sont :

- Anonymous (pas d'authentification)
- Basic (mot de passe en clair - RFC2617)
- Basic sur connexion SSL
- Digest Authentication (MD5 avec secret partagé - RFC2617)
- Challenge/Response
  - NTLM
  - Passport
- Client Certificates (certificats clients SSL).
- Fortezza (solution à base de certificats).

Le risque est bien entendu qu'un authentifiant connu du système (ex. login Windows) soit passé par défaut dans un contexte de connexion inapproprié (ex. accès à un site Internet). Suite à des avis de sécurité sur le sujet, les paramètres d'authentification par défaut sont désormais :

- Zone Internet, Sites sensibles : demander le mot de passe.

– Zone Intranet, Sites de confiance : login automatique (avec le login Windows!). Les risques sont donc limités au réseau interne. On notera que le client Telnet présentait le même type de comportement dangereux (cf. MS00-067).

Authentification applicative Internet Explorer propose plusieurs options de "saisie semi-automatique" :
- Adresses Web.
- Contenu des formulaires.
- Logins dans les formulaires.
- Mots de passe dans les formulaires.

Les mots de passe sont stockés dans le "*Protected Storage*", c'est-à-dire la clé :
"`HKCU\Software\Microsoft\Protected Storage System Provider`" (invisible par défaut, même aux administrateurs).

Suite à de nombreux problèmes de sécurité dans les versions antérieures de Windows, ce "*Protected Storage*" offre désormais une API de stockage sécurisé unique pour les applications. Cet emplacement de stockage est chiffré avec le mot de passe de login Windows.

Il est toutefois possible (sous certaines conditions) d'accéder aux données contenues dans cet emplacement (cf. outils "*IE Password Revealer*", sites LostPassword, Elcomsoft, etc.).

Il est donc recommandé de désactiver toute forme de saisie semi-automatique dans Internet Explorer.

3.8 Synchronisation horaire

Par défaut les postes XP utilisent une synchronisation horaire
- Dans le cas d'une machine en domaine, le serveur par défaut est le contrôleur de domaine défini comme source de temps.
- Dans le cas d'une machine en "*Workgroup*", le serveur par défaut est "*time.windows.com*" (alternativement "time.nist.gov"). L'intervalle de mise à jour par défaut est de 1 semaine.

Ce comportement est paramétrable dans la clé de base de registre
"`HKLM\System\CCS\Services\W32Time`".

Le protocole NTP standard est utilisé. Aucune information indésirable n'est transmise.

4 Composants préinstallés

4.1 Windows Media Player

La version de Windows Media Player livrée avec Windows XP SP1 est la 8.0. Curieusement celle-ci n'est pas téléchargeable sur le site de Microsoft, seule les versions 6.4, 7.0 et 9.0 étant publiques.

Windows Media est typiquement une application faisant un usage immodéré d'Internet, par exemple pour les fonctions suivantes :
- Acquisition de licences (DRM).
- Accès à des services "en direct" (contenu à la demande, radios).
- Base de métadonnées CD et DVD (en lecture/écriture).

- Téléchargement de codecs.
- Mises à jour logicielles.
- Skins et visualisations.
- "*Media Library*", "*Media Guide*", *Newsletter MSN*, ...

Le site de référence pour tous ces accès est `http://www.windowsmedia.com/`.

Les risques associés sont très importants. Outre les problèmes d'atteinte à la vie privée, Windows Media étant un superbe outil de marketing personnalisé, il existe des problèmes de sécurité intrinsèques :

- Le lecteur Windows Media possède un identifiant unique (GUID), utilisé techniquement pour assurer la qualité de service sur les serveurs de contenu à la demande. Windows Media étant un composant ActiveX scriptable par des tiers, ce GUID permet d'identifier un poste de manière unique via un navigateur : il s'agit de la notion de "*supercookie*".
- Les "*skins*" et visualisations sont des archives ZIP incluant du contenu actif, d'où un risque d'exécution de code malveillant.

4.2 Internet Explorer

Windows XP SP1 est livré avec la version 6.00.2600.1106 d'Internet Explorer. Bien que les couches superficielles du logiciel puissent être supprimées, il n'est effectivement pas possible de désactiver le moteur de rendu HTML contenant la majorité des bogues, celui-ci étant exploité par d'autres outils tels que Explorer.

Les accès Internet inattendus effectués par IE sont les suivants :

- Page initiale, permettant l'élaboration de statistiques d'installation.
  - `http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=msnhome`
- Spyware "*Alexa*" (détecté par l'outil "*Ad-Aware*").
  - En lien avec l'option "*effectuer des recherches depuis la barre d'adresses*".
- Option "*Vérifier les signatures des programmes téléchargés*".
- Option "*Vérifier la révocation des certificats*".
- Option "*Vérifier la révocation des certificats de l'éditeur*".
- Option "*Vérifier automatiquement les mises à jour de IE*".
- Option Windows "*Mise à jour des certificats racine*".
  - Si un certificat SSL signé par une autorité inconnue est présenté, une mise à jour de la base des autorités racine est déclenchée depuis le site `http://www.download.windowsupdate.com/msdownload/update/v3/static/trustedr/en/authrootseq.txt`

Internet Explorer dans sa configuration par défaut communique donc régulièrement avec des sites Microsoft, toutefois aucune information sensible n'est échangée. A noter qu'Internet Explorer est aussi une source privilégiée pour l'installation de Spywares "*tierce partie*" par le biais des mécanismes suivants :

- Gestion des cookies (par défaut le navigateur utilise P3P).
- Options "*Activer les extensions tierce partie*", "*Activer l'installation à la demande*", "*Éléments installables du bureau*".
- Bogues permettant d'exécuter du code ...

4.3 Windows Messenger

Windows XP SP1 est livré préinstallé avec Windows Messenger 4.7.

Ce composant utilise indifféremment et simultanément les 3 services d'annuaire suivants :

- Exchange 2000 (si configuré).
- Serveur SIP (Session Initiation Protocol) - RFC 2543.
- Serveur Microsoft avec authentification Passport.
  - En direct : `http://messenger.hotmail.com:1863/`.
  - Via un proxy : POST. `http://gateway.messenger.hotmail.com/gateway/gateway.dll?Action=open&Server=NS&IP=messenger.hotmail.comHTTP/1.1`

Le protocole de base est HTTP, celui-ci servant à encapsuler 2 sous-protocoles propriétaires :

- Des commandes de type XYZ [paramètre 1] [paramètre 2] [...].
- Des données de type MIME propriétaire (ex. "*application/x-msn-messenger*", "*text/x-msmsgsprofile*", ...) véhiculant des données partiellement brouillées selon un algorithme inconnu.

Les risques associés à l'utilisation de ce composant sont :

- La divulgation d'informations personnelles en clair (via le protocole HTTP).
- Un système à serveur central donc adapté à la traçabilité et au contrôle.
- Un niveau d'informations échangées inconnu à cause du brouillage des données.

Exemple de contenu capturé :

- `Content-Type: text/x-msmsgsprofile; charset=UTF-8`
- `EmailEnabled: 1`
- `MemberIdHigh: 9xxxx`
- `MemberIdLow: -2114xxxxxx`
- `lang_preference: 1036`
- `preferredEmail: xxxxxxx@hotmail.com`
- `country: FR`
- `PostalCode: 75010`
- `Gender: m`
- `Kid: 0`
- `Age: 26`
- verb+BDayPre : 2+
- verb+Birthday : 2.821600e 004+
- verb+ Wallet : 0+
- verb+Flags : 1027+
- verb+sid : 507+
- verb+kv : 4+
- verb+MSPAuth : 4n3llLtj1DTLjIKvsjAeFx3NL3kmxyhl5V5207HY!tFCSReUcu...+
- verb+ClientIP : 212.xxx.xxx.xxx+
- verb+ClientPort : 0+

5 Office XP

Les problèmes de confidentialité liés à la suite Office XP ne seront que brièvement évoqués, puisque le sujet a été traité dans le magazine MISC n°7 ("*La fuite d'information dans les documents propriétaires*").

Les principaux reproches adressés à la suite Office sont :

- Forte intégration avec le système Windows.
  - Ex. Word devient l'éditeur HTML par défaut.
  - Forte intégration des produits entre eux.
  - Ex. envoyer un document Word avec Outlook modifie les propriétés du document.

Parmi les risques bien documentés on peut citer :
- La verbosité des propriétés du document (auteur, temps d'édition, chemins UNC).
- L'enregistrement de l'historique du document (versions antérieures).
- Les "*Word bugs*".
- Les risques liés aux macros et l'absence de solution satisfaisante.
- Le vol de données par les champs de fusion.
- L'utilisation de l'adresse MAC comme GUID.
- Etc...

6 Les solutions

Tous les problèmes évoqués précédemment ne sont pas sans solution (heureusement). A l'aide des possibilités offertes par le système lui-même, il est possible de modifier le paramétrage par défaut (souvent insatisfaisant) et de désactiver les accès Internet des composants :
- Via l'interface graphique.
- Via des clés de base de registre.
- Via les GPO.
- Via les "Administration Kits" (ex. IEAK).

A noter que Windows possède un paramètre de configuration global du Proxy, qui permet de rediriger les accès Internet indus vers "*/dev/null*". Il reste ensuite à utiliser des logiciels gérant des paramètres Proxy personnalisés (ex. Netscape).

Enfin d'autres mesures pourraient être :
- Désintaller les composants cachés (fichier SYSOC.INF).
- Utiliser la fonction de restriction d'exécution.
- Mettre en place des miroirs internes (ex. MSUS).

Lorsque le système s'avère insuffisant pour bloquer un composant spécifique, il est possible d'utiliser des outils tiers tels que firewall personnel ou logiciel de configuration "*anti-spyware*".

Une solution radicale consiste à isoler les systèmes Windows XP de tout accès Internet, ...

7 Conclusion

Le sujet des interactions entre Windows XP et les sites Microsoft est loin d'être clos (voir annexe A),...

Windows XP SP1 communique régulièrement avec des sites Internet, de manière plus ou moins documentée et/ou configurable. Ces fonctions sont activées par défaut mais dans la plupart des cas désactivables.

Une étude plus poussée montre que les informations collectées par Microsoft sont individuellement peu significatives, mais leur recoupement permettrait d'obtenir un puissant outil de marketing personnalisé. Seules quelques fonctions (telles que Remote Assistance ou Passport) mettent en péril de façon significative la sécurité du système ou des informations qu'il contient.

D'autre part les informations transmises bénéficient d'un niveau de protection très hétérogène (protocoles HTTP ou HTTPS, chiffrement, brouillage, protocole propriétaire, etc...).

On notera avec plaisir qu'il existe des moyens de se protéger, le plus simple étant de ne pas renseigner l'adresse de son Proxy au niveau de Windows.

Références

1. *Using Windows XP Pro SP1 in a Managed Environment : Controlling Communication with the Internet*, `http://technet.microsoft.at/includes/file.asp?ID=4668`
2. *XP Anti-Spy*, `http://www.xp-antispy.de/`
3. *Windows XP shows the direction Microsoft is going*, `http://www.hevanet.com/peace/microsoft.htm`
4. *Microsoft Secrets*, `http://www.securityoffice.net/mssecrets/`

A Sujets non traités

- "*Application Help*" / "*Driver Protection*" / Assistant Compatibilité. Microsoft maintient une base d'applications incompatibles et de correctifs : `APPHELP.SDB + SYSMAIN.SDB / DRVMAIN.SDB`.
  Cette liste est mise à jour par WindowsUpdate.
- "*Device Manager*".- Les pilotes signés peuvent être mis à jour en 1 click. Cette fonction est gérée par WindowsUpdate.
- Journal d'événements.- La plupart des événements système contiennent un raccourci vers un site explicatif : `http://go.microsoft.com/fwlink/events.asp`.
  Ce site est configurable via les clés suivantes :
  - *MicrosoftRedirectionURL*.
  - *MicrosoftRedirectionProgram*.
  - MicrosoftRedirectionProgramCommandLineParameters.
- Associations de fichiers.- Cliquer sur un fichier dont l'extension n'est pas associée provoque la redirection vers un site Microsoft : `http://shell.windows.com/fileassoc/nnnn/xml/redir.asp?ext=AAA` (Nnnn = langue, AAA = extension)
  Cette option est configurable via la clé NoInternetOpenWith.
- Jeux "*on line*".- Se connectent au site `http://www.zone.msn.com/`
- Netmeeting.- Se connecte à un serveur ILS au choix (par défaut : *netmeeting.microsoft.com*). Les ports utilisés sont : TCP/389, TCP/522, TCP/1503, TCP/1720, TCP/1731 + ports dynamiques.
- "*Online Device Help*", Plug-and-Play.- Aide en ligne pour la recherche de drivers si un périphérique inconnu est détecté ou lors de l'insertion de nouveaux périphériques. Transmet le profil matériel du périphérique (PnP ID). Site `http://www.microsoft.com/windows/catalog/`
- Outlook Express 6.
- Universal Plug-and-Play.- Requêtes UDP/1900 pour la détection de matériel réseau.
- MSN Explorer.- Portail Internet Microsoft.

B Sites Microsoft cités dans la présentation

Microsoft.com

- `http://oca.microsoft.com/`
- `http://go.microsoft.com/fwlink/?LinkID=11`
- `http://go.microsoft.com/fwlink/events.asp`

- http://watson.microsoft.com/
- http://windows.microsoft.com/windowsxp/newsver.xml
- http://windowsupdate.microsoft.com/
- http://wpa.one.microsoft.com/
- http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=msnhome
- http://www.microsoft.com/windows/catalog/
- http://www.microsoft.com/piracy/basics/activation/
- http://v4.windowsupdate.microsoft.com/CAB/x86/unicode/iuctl.CAB

MSN.com, hotmail.com

- http://messenger.hotmail.com:1863/
- http://gateway.messenger.hotmail.com/
- http://ie.search.msn.com/
- http://www.zone.msn.com/

Passport.net

- http://www.passport.net/
- http://register.passport.net/

WindowsUpdate

- http://www.windowsupdate.com/
- http ://www.download.windowsupdate.com/msdownload/update/v3/static/trustedr/en/authrootseq.txt

Autres

- http://reg.register.microsoft.akadns.net/
- http://www.windowsmedia.com/
- http://shell.windows.com/fileassoc/nnnn/xml/redir.asp?ext=AAA

Conférences invitées

Les enjeux de la sécurité des systèmes d'information

GDI Jean-Louis Desvignes

Ecole Supérieure
et d'Application des Transmissions
Quartier Leschi - BP 18
35998 Rennes Armées

A Introduction

C'est pour moi un très grand plaisir d'ouvrir ce cycle de conférences consacré à la SSI : c'est d'abord avec une certaine émotion que je retrouve l'amphi de cette belle école quelques vingt-cinq ans après l'avoir quittée. C'est ici que je me suis réellement initié aux joies des systèmes informatiques et c'est ici d'ailleurs, que j'ai découvert que ces machines étaient foncièrement sournoises : déjà à l'époque, lorsqu'elles avaient enfin accepté d'effectuer une tâche (écrite en FORTRAN ou en COBOL), il n'était pas certain que celle-ci fût réellement et complètement accomplie et il n'était pas exclu en revanche que ces automates évolués n'en eussent pas exécuté une autre qui ne leur était pas demandée.

Aujourd'hui, ce n'est plus une suspicion, c'est une certitude !

C'est ensuite un plaisir de me retrouver dans ce milieu délicieusement paranoïaque du monde de la sécurité. Comme c'est agréable de pouvoir frissonner ensemble en évoquant les dangers qui nous guettent chaque fois que nous nous laissons aller à jouer de notre clavier ou à flirter avec une souris ("*caresser le mulot*" aurait rectifié notre Président) en étant certains d'être compris à demi-mot ! Quel bonheur de pouvoir fustiger l'inconscience de ces responsables qui mettent en péril la vie de leur société, voire des services de l'Etat, par leurs coupables négligences ! Et quels frissons nous parcourent lorsque nous évoquons les mille et une manières que peuvent employer les Big Brothers potentiels de la planète pour épier nos moindres faits et gestes : les coups de fil que nous donnons et tous nos déplacements par l'exploitation des capacités des réseaux cellulaires, les lieux où nous dégainons notre carte bancaire, nos excès de vitesse sur les autoroutes, bientôt, en raison de la trahison des tickets de péage ; et toutes nos petites manies à cause des traces que nous laissons sur le WEB ou des petits cadeaux empoisonnés que nous récoltons au gré de nos promenades sur la toile. Ah ! Quel vertige s'empare de nous quand nous pensons que toutes ces informations, qui constituent des pans entiers de notre vie, sont stockées à notre insu, des semaines, des mois, dans certains cas des années. Alzheimer peut bien frapper, nous sommes protégés, notre mémoire est externalisée !

On peut effectivement prendre les choses avec bonne humeur et se dire qu'après tout, toutes les possibilités qu'offrent les nouvelles technologies constituent des avancées propres à satisfaire le mieux possible nos besoins de consommateurs paresseux, pressés et exigeants, ou à garantir notre sécurité vis à vis des mauvaises gens. Mais on peut tout aussi bien les aborder avec effroi en constatant que le périmètre de notre vie privée s'est rétréci comme peau de chagrin, davantage encore depuis le 11 septembre 2001, et

que ce n'est pas fini ! Certaines annonces relatives aux nouvelles générations de puces ont en effet de quoi nous faire frémir, même s'il n'est pas encore envisagé de nous les greffer sous la peau, ...

Pourtant, une discipline, une science, un art, je ne sais trop comment la qualifier, est censé nous apporter la sérénité et nous permettre de redevenir maîtres de ce que nous souhaitons ou divulguer ou au contraire protéger parmi les informations qui nous concernent ou qui nous sont confiées à un titre ou à un autre : c'est **la sécurité des système d'information** ou, pour adopter le langage de ce séminaire, **la sécurité des technologies de l'information et de la communication**.

B Qu'est-ce que la SSI ?

La première expression, née au milieu des années 80, désigne l'ensemble des techniques propres à garantir les informations traitées ou transmises par un système d'information, au sens large, en termes de **confidentialité, d'authenticité, d'intégrité et de disponibilité** comme tout un chacun le sait dans cette noble assemblée.

Permettez-moi simplement de souligner **l'importance de la disponibilité** des informations qui implique, de facto, celle du système qui les traite lui-même. Dès lors la SSI concerne bien évidemment tout système d'information, quand bien même celui-ci ne traiterait aucune information confidentielle : ce n'est pas parce que le PABX d'une entreprise voit transiter des informations sensibles qu'il doit être protégé (si tel était le cas ses dirigeants seraient coupables de légèreté) mais parce que sans lui, l'entreprise ne peut plus travailler. Par extension, tout système pilotant un processus doit être l'objet d'une protection SSI, qu'il s'agisse des télécommandes d'un lanceur spatial, du système de contrôle d'une centrale nucléaire, du système de gestion des stocks d'une usine automobile, etc,... qui doivent être protégés comme l'ordinateur de la secrétaire de direction, celui de notre médecin traitant ou la puce de notre carte de paiement. "*Vaste programme !*" comme aurait dit le général de Gaulle. Et pourtant, c'est bien de cela qu'il s'agit car les enjeux de la SSI sont énormes.

Si vous le permettez, je commencerai par évoquer ses principales composantes et les évolutions qui sont intervenues sur chacune d'elles avant d'examiner les enjeux.

La SSI, c'est l'art de combiner un ensemble de mesures préventives et curatives, à la fois au plan technique et organisationnel en vue de faire face aux menaces que l'on aura pris soin, au préalable, d'identifier et de hiérarchiser. Ce n'est pas à vous que je vais apprendre que toute entreprise de sécurisation doit débuter par une analyse des risques et des menaces : que dois-je protéger, contre qui ou quoi ? Pourtant, combien de sociétés ou d'organismes se laissent-ils encore séduire par toutes sortes de solutions techniques coûteuses ne répondant pas à leur véritable besoin de sécurité. Rappelons-nous que, bien souvent, de simples mesures organisationnelles bien appliquées suffisent à contrer la majorité des risques, surtout lorsque ceux-ci peuvent provenir aussi bien de l'intérieur que de l'extérieur de l'entreprise.

Mais je suppose que si vous êtes venus à ce séminaire, vous savez cela très bien, et c'est surtout de technique que vous souhaitez traiter.

Sur ce plan je rappelle que la SSI repose sur trois techniques de base, par ordre d'apparition : la cryptographie, l'anti-compromission électromagnétique et la sécurité informatique. Ces trois techniques de protection sont aujourd'hui étroitement imbriquées, comme le sont d'ailleurs de plus en plus les télécommunications et l'informatique, ce qui implique dorénavant une approche globale de la SSI.

C Au commencement était la Cryptographie...

Première apparue dans l'histoire de l'humanité, la cryptographie, qui n'avait guère évolué pendant deux millénaires – on en était resté au bon vieux principe des systèmes symétriques de partage d'une convention secrète, ce qui rendait délicate toute utilisation de masse – a connu sa grande révolution avec l'invention, au milieu des années soixante dix, des systèmes asymétriques, dits "*à clefs publiques*. Cette découverte, née à l'occasion de la compétition qui allait être remportée par le fameux DES (Data Encryption Standard), d'ailleurs symétrique, d'IBM, a mis quelque temps à entrer en application, mais a largement dépassé aujourd'hui son objectif initial : comme on le sait, selon que l'on utilise un tel procédé, par exemple le célèbre R.S.A. (du nom de ses inventeurs : Rivest, Shamir et Aldeman), dans un sens ou dans un autre, on obtient soit un système de chiffrement, soit un système de signature assurant l'authentification des correspondants, l'authenticité et l'intégrité des informations. En combinant de surcroît ce procédé asymétrique peu performant en terme de débit d'information avec un procédé symétrique, on dispose de la boîte à outil du parfait cryptologue, prêt à satisfaire le besoin de sécurité exprimé pour toutes les formes de transaction électronique.

Naturellement les choses ne sont pas aussi simples, et l'installation d'un système de sécurité reposant sur l'usage des clefs publiques pour un vaste réseau de correspondants tel que celui du réseau Santé sociale, celui du monde bancaire ou celui de la Défense, se heurte à de nombreuses difficultés, notamment d'organisation des infrastructures de gestion de clefs (IGC). Le seul problème de la désignation d'une **autorité de certification** et de sa reconnaissance par d'autres a déjà alimenté bien des débats.

Quant à la protection réellement assurée par les procédés cryptologiques, celle-ci est naturellement au cœur des enjeux. Depuis la nuit des temps le combat entre l'épée et la cuirasse était resté équilibré : décrypter la scytale ou Jules César est à la portée d'un enfant sachant lire et écrire. Casser le chiffre allemand pendant la Première guerre mondiale, le capitaine Painvin l'a fait ; venir à bout de l'ENIGMA était un sacré défi, mais les Anglais de Bletchey Park, bien aidés par les services polonais et français, y sont parvenus et c'est sans doute grâce à cela que la bataille de l'Atlantique, véritable tournant de la seconde guerre mondiale, a été remportée. Pourtant, à un certain moment, face à des algorithmes solides, soumis à la sagacité de la communauté internationale des cryptologues, on a vraiment cru que la cuirasse l'avait définitivement emporté. Qu'une nouvelle puissance de calcul se profile à l'horizon pour mener à bien l'exploration exhaustive de toutes les combinaisons d'une clef, (un nouveau supercalculateur, des machines massivement parallèles ou des milliers d'ordinateurs personnels connectés à travers l'Internet) il semblait suffisant d'augmenter de quelques bits la longueur de la clé, en théorie, pour se remettre à l'abri pour plusieurs années (de 40 bits on passait à 56, de 56 à 64, etc...)[1]

[1] Pour les systèmes asymétriques, la problématique est différente : s'il s'agit de factoriser des nombres premiers (c'est-à-dire de retrouver les deux nombres premiers à partir de leur produit quand ce sont des nombres de plus d'une centaine de chiffres, 1024, 2048 bits, etc..) certes, on peut compter sur la puissance de calcul mais aussi sur la découverte d'un meilleur algorithme de factorisation, une astuce mathématique en somme, qui pourrait subitement décrédibiliser les systèmes les plus répandus. Pour certains, casser le RSA est leur raison de vivre, pour ceux qui l'utilisent massivement, comme les banquiers, la perspective d'une telle percée est un véritable cauchemar !

D Des clefs longues certes, mais ....

Mais la longueur des clefs n'est pas tout dans un système de sécurité. Vous le savez bien. Cependant, dans le débat sur la libéralisation de la cryptologie qui faisait rage il y a quelques années, on a, pour simplifier les dossiers à l'intention des politiques, quelque peu caricaturé la problématique et finalement tout le monde s'est polarisé sur ces fichues longueurs de clefs alors que les vrais problèmes étaient ailleurs. Il y a parmi vous des spécialistes qui savent que concevoir un bon système de chiffrement n'est déjà pas si aisé, mais que le réaliser et le mettre en œuvre pour qu'il soit suffisamment ergonomique et utilisable par un quidam en toute sécurité est un autre challenge ! A quoi bon s'encombrer d'une clef de 128 bits, si seulement 8 bits sont effectivement utilisés, si votre clef se promène quelque part sur votre ordinateur ou si elle est transmise à votre insu avec vos messages ?

Or, et c'est là l'un des effets néfastes de la libéralisation de la cryptologie tellement applaudie en 1999 lorsqu'elle a été annoncée : ce que certains politiques ont pris pour un remède miracle contre les actes de piratage de toutes sortes contre lesquels ils voulaient nous protéger (rappelez-vous ECHELON !) s'est en fait traduit par **un envahissement de produits de sécurité aussi efficaces qu'un placebo**. La plupart des logiciels de cryptographie grand public affichant pourtant des clefs de belle taille ne résistent pas aux investigations d'un stagiaire de l'un des prestigieux cours du CFSSI, du mastère spécialisé de SUPELEC ou de l'ENST. La preuve avait été établie, bien avant le 11 septembre, que les modules cryptographiques d'un grand éditeur de logiciel étaient "*plombés*" à la demande de son gouvernement, on imagine ce qu'il peut en être aujourd'hui ...

Mais, quand bien même le module cryptographique serait de bonne facture, celui-ci peut dans bien des cas être comparé à une porte blindée que l'on fixerait sur des cloisons en placoplâtre. A quoi bon, faut-il encore le souligner, s'embêter à chiffrer ses fichiers avant de les envoyer si ceux-ci sont accessibles lorsqu'ils sont encore en clair, à partir de l'Internet, en utilisant l'une des multiples failles du système d'exploitation devenu le standard mondial dont on ne veut pas nous donner les sources. J'y reviendrai tout à l'heure.

E Le débat sur la cryptographie : un écran de fumée ?

Vous aurez compris, à travers mes propos, que je considère que le problème de la cryptologie, s'il est loin d'être secondaire, n'en a pas moins servi à occulter celui, plus délicat, de **la sécurité informatique**. Pendant que l'on guerroyait entre spécialistes de fraîche date sur la longueur des clefs, d'autres imaginaient comment rester maîtres des systèmes informatiques, quand toutes les informations de la planète seraient chiffrées. On perçoit bien l'importance relative des deux domaines à l'aulne des mesures de restriction au commerce des systèmes informatiques sécurisés. Alors qu'ils libéralisaient les produits cryptologiques apparemment forts, les E.U. maintenaient l'embargo sur les systèmes évalués au plus haut niveau de sécurité selon les critères de l'ORANGE BOOK.

Il est vrai que sur le plan de la sécurité informatique, il y a fort à faire si l'on veut pouvoir se mettre un jour au volant de son ordinateur et prendre les autoroutes de l'information, pour utiliser une expression désuète, en toute sécurité. Vous connaissez certainement cette boutade émanant d'un grand constructeur d'automobiles : "si on avait construit les voitures comme on a construit l'informatique, on roulerait encore en

trottinette" ! C'est un jugement un peu sévère, mais il est vrai qu'il y a peu de temps, lorsqu'un bogue était découvert dans un logiciel, on nous demandait d'être patient, et d'attendre la version suivante pour le voir corrigé. Aujourd'hui, grand progrès : on a la possibilité de souscrire un contrat nous donnant le droit d'accéder à des "*patchs*" correctifs. Pauvres constructeurs automobiles qui sont encore obligés de rappeler à grands frais des milliers d'autos pour modifier un boulon ! La facilité de réparation n'est cependant pas un gage de qualité de service au bout du compte. On s'habitue à ce genre de rustines et à cette accumulation de couches de logiciels qui encombrent nos machines et nous obligent à en acheter sans cesse de plus puissantes pour des tâches qui, dans bien des cas, n'ont pas fondamentalement évolué.

F Des livres de différentes couleurs

Les travaux américains sur la sécurité informatique, le COMPUSEC comme ils disent, furent concrétisés au début des années 80 par la publication de l'"*ORANGE BOOK*" qui établissait les critères permettant de classer les systèmes informatiques selon le niveau de confiance que l'on pouvait leur accorder.

Très vite quatre pays européens ont compris que ces critères pourraient fausser la compétition économique et se sont efforcés de définir leurs propres critères : il y eut bientôt un livre de couleur différente par pays avant que les quatre ne se décident à harmoniser ces critères qui devinrent les "*ITSEC*". Ceux-ci, heureusement, ne différaient pas des critères américains uniquement par la couleur mais par leur pertinence.

Mon premier travail en arrivant au SCSSI fut, d'ailleurs, de mettre en place le schéma national de certification des produits de sécurité évalués selon ces critères. Cela consistait, en particulier, à faire accréditer par le COFRAC les laboratoires d'évaluation selon la norme ISO/IEC 17025 et à les agréer selon des critères gouvernementaux. Mais, en parallèle nous avions déjà engagé des travaux pour aboutir à la convergence américano-européenne. Les "*critères communs*" furent publiés en 1999 et un accord de reconnaissance mutuelle fut alors signé entre six pays : ces critères allaient ensuite être adoptés comme une norme ISO (ISO/IEC 15408).

Un grand pas avait été franchi en matière d'instauration de la confiance réciproque. Je dois dire que j'étais particulièrement fier de l'action des équipes françaises dans cette entreprise. A côté du SCSSI, les laboratoires du CNET repris par le LETI, du CELAR et d'autres laboratoires privés avaient acquis une solide expérience en évaluation. Dans un domaine particulier, celui des cartes à puce sécurisées, nous étions, et j'espère que nous le sommes encore, leaders mondiaux.

G La carte à puce : meilleur rapport coût/efficacité en matière de sécurité

Il est inutile, je crois, de signaler l'importance de ce domaine pour tout ce qui touche à la sécurité et les enjeux colossaux de ce marché. La carte à puce est en effet l'élément qui a le meilleur rapport cout/efficacité pour améliorer de manière significative le niveau de sécurité d'un système Or, les Américains, pour diverses raisons en particulier juridiques et commerciales, n'ont pas cru immédiatement à cette technologie. Je me souviens que la première fois où l'ordre du jour d'une réunion avec la NSA a comporté l'item "*smart cards*", ce devait être en 1986, nos collègues américains ont déposé sur la table une calculette genre convertisseur d'euros pour personnes âgées, tandis que nous

exhibions la première CP8 de BULL. Inutile de vous dire qu'ils ne nous ont pas pris au sérieux avec notre bout de plastique.

Aussi, quel plaisir ai-je ressenti, lorsque fin 99, le patron de la branche INFOSEC de la NSA, m'a demandé s'il pourrait bénéficier de notre aide parce que le Département de la Défense avait décidé de doter tout son personnel de carte à puce. En lui répondant que nous, petits français, étions prêts à apporter notre assistance à la colossale NSA, je pensais à l'histoire de cette souris se promenant dans la savane à côté de son copain l'éléphant et qui, se retournant, s'exclamait : "*tu as vu toute la poussière que nous faisons ?*".

H Des tentatives de déstabilisation

Pourrons-nous préserver cette avance ? La lutte est féroce. Il y a eu des tentatives de déstabilisation et de discrédit de notre technologie par des laboratoires commandités par de grands éditeurs de cartes ; il y a eu la publication sur le NET de la liste de toutes les vulnérabilités des cartes. Si le pirate Serge Humpich avait eu cette liste, il ne lui aurait pas fallu quatre années de labeur pour fabriquer ses fausses cartes (les "*YES CARDS*").

Aujourd'hui, d'un côté, il y a des tentatives américaines pour mettre la main sur ce secteur notamment par les prises de participation dans Gemplus (et probablement des transferts de brevets), de l'autre, l'initiative conjointe INTEL /MICROSOFT sur la puce FRITZ et le logiciel PALADIUM, pourrait bien assécher, comme le craint Ross Anderson, le marché de la carte à puce, si les fonctions que celle-ci assure peuvent être reprises par celle-là. L'horizon est donc loin d'être clair.

Quittons les cartes pour revenir aux systèmes d'exploitation les plus répandus et dont on ne veut pas nous donner les sources.

I Et les logiciels libres ?

Ce refus est peut-être, tous comptes faits, pure charité, car je ne connais pas grand monde aujourd'hui, à part les Chinois, en mesure de se lancer dans l'investigation des quelques dizaines de millions d'octets que comportent les différentes strates des packs de M.S ! Certes on peut y découvrir quelques bogues accidentels ou non, mais on n'aura jamais l'assurance qu'il n'en subsiste pas. Ah ! Que notre bon vieux MINITEL était rassurant sur ce point ! Pas d'intelligence, pas de malice ! Comme les fantassins !

Pour faire face à ce double problème de monopole et de défiance, une voie existait pourtant. Celle-ci n'a pas suffisamment été explorée quand il était encore temps de ne pas se livrer corps et âme à ce cher, très cher Bill ! Celle des logiciels libres. Certains Etats ont pourtant manifesté des velléités de le faire, comme l'Allemagne, j'ai même appris que la Tunisie s'était attelée à ce challenge. En France, est-ce à cause de l'échec du tristement célèbre Plan Calcul que le gouvernement, qui a lancé le PAGSI, et l'Administration, qui s'informatise pourtant à marche forcée, se sont montrés si frileux et sans ambition sur ce dossier ? Toujours est-il que même un département ministériel tel que celui de la Défense, généralement soucieux de préserver l'indépendance et la sécurité de notre outil militaire, ne s'est pas engagé avec conviction dans la voie des logiciels libres. Je connais une armée qui a même choisi, pour des raisons certes louables d'uniformisation et de simplicité de mise en œuvre, d'utiliser Windows pour ses serveurs quand la grande majorité du marché plébiscite LINUX !

Cela dit, il ne faut sans doute pas attendre des miracles ni des économies aussi substantielles que celles que certains prétendent faire en recourant aux logiciels libres. D'abord l'investissement intellectuel pour les organismes qui les utilisent n'est pas à négliger. Ensuite, la faveur dont ils sont l'objet depuis quelques temps a eu tôt fait d'estomper leur caractère libre. Nombre d'offres reposent désormais sur des solutions, certes à base de logiciels libres, mais exigeant pour être intégrées et maintenues, des prestations grassement rémunérées et débouchant sur des produits propriétaires. C'est normal.

J Une ambition européenne ?

Pourtant, je persiste à penser qu'un recours aux logiciels libres est possible et serait hautement bénéfique, tant en termes de coût qu'en termes de confiance. J'ajoute que c'est à l'Administration de donner l'exemple : elle regorge de gens compétents qui trouveraient un intérêt immense à se lancer dans cette aventure. Naturellement, ce que je dis pour notre pays prendrait encore plus de sens si une telle action était réellement soutenue à l'échelle de l'Union européenne. Nous avons réussi dans les domaines de l'énergie nucléaire, de l'aéronautique et de l'espace, à gagner notre indépendance, nous allons lancer GALILEO, le concurrent du GPS, pourquoi ne tenterions-nous pas de conquérir notre autonomie dans les technologies de l'information si, comme on nous l'assène depuis des années, l'information est devenue la matière première la plus précieuse ?

Une première étape a été franchie il y a peu de temps : une agence européenne de la sécurité des systèmes d'information va voir le jour. Il y a quelques années, je l'appelais de mes vœux, tant j'étais ulcéré de voir nos systèmes de protection les plus évolués traverser l'Atlantique pour se faire évaluer dans une agence de l'OTAN située en fait au sein de la NSA. Certains pays étaient même prêts à s'en remettre à celle-ci pour choisir des équipements de chiffrement destinés à un usage purement européen ! Je précise qu'il ne s'agissait pas de la Grande Bretagne, ...J'avais, à l'époque, préconisé d'utiliser les compétences des trois ou quatre pays ayant une compétence en cryptologie, en créant dans un premier temps une sorte d'agence virtuelle s'appuyant sur les accords de reconnaissance mutuelle des certificats de sécurité qui venaient d'être signés ... Je vois avec satisfaction que nous avons franchi un grand pas et j'espère que le représentant de la DCSSI pourra nous dire quel rôle la France entend jouer dans cette nouvelle agence.

K BIG BROTHER n'est pas loin !

Car la vigilance s'impose plus que jamais : il y a quelque temps déjà, dans le but de lutter contre le piratage de logiciels, de grandes firmes américaines avaient imaginé de graver dans le silicium des microprocesseurs un numéro d'identité qui aurait pu servir à vérifier, au moment d'une connexion volontaire ou indirecte avec le site de Microsoft, que vous aviez bien acquitté les droits de vos licences en ouvrant toute grande la porte de votre disque dur à votre éditeur préféré. A l'époque, je me souviens qu'un slogan publicitaire avait été détourné : "*la NSA et ... MICROSOFTen avaient rêvé, INTEL l'a fait*".

Une levée de boucliers des mouvements libertaires s'en était suivie et l'on avait fait mine de faire machine arrière. Depuis il y a eu le 11 septembre, un deal dans un grand procès américain est intervenu, et le gendarme du monde ne s'embarrasse plus de scrupules pour assurer sa sécurité. C'est dans ce contexte que se profile l'arrivée de

nouvelles puces qui pourraient bien, si l'on n'y prend garde, perfectionner le dispositif précédent. Je sais que les puces "*FRITZ*" que j'ai déjà évoquées, suscitent des débats et je suis certain que ce thème sera abordé ici même.

Pour résumer à l'intention de ceux qui n'auraient pas encore entendu parler de ces nouvelles plates-formes de INTEL et du logiciel PALLADIUM qui lui serait associé, il s'agit de faire rentrer dans une puce appelée TCPA qui signifie *Trusted Computing Platform Alliance* une partie du logiciel PALLADIUM également nommé NGSCB (*Next Génération Secure Computing Base*) incorporé dans les futures versions de WINDOWS (et déjà intégré dans XP) de manière à rendre impossibles les opérations de piratage de toute nature. Ceci, en procédant à des contrôles systématiques locaux et **en ligne** de votre configuration matérielle et logicielle. Ce contrôle pourrait aller jusqu'à l'effacement des programmes illicites et même des données obtenues par un traitement illicite!

Toutes sortes de dérives de ce système censé lutter contre le piratage sont imaginables et je ne saurais trop vous conseiller de lire les articles et la foire aux questions de Ross Anderson sur le sujet. Les conséquences de cette innovation peuvent êtres extrêmement lourdes, autant en terme économique qu'en termes de souveraineté des Etats (quid des systèmes de la Défense utilisant WINDOWS ?) ou de libertés individuelles. Les mouvements libertaires qui s'étaient mobilisés contre les projets précédents (CLIPPER et numéro unique) seront-ils capables de résister une nouvelles fois dans le nouveau contexte sécuritaire ? Rien n'est moins sûr. Aux Etats-Unis comme partout, l'heure n'est pas propice aux états d'âme face aux atteintes à la liberté. Mais je suis sûr que le représentant de MICROSOFT qui nous fait l'honneur de participer à ce séminaire se fera un plaisir de nous rassurer.

Cela dit, si en matière de SSI il faut parfois se méfier de ses amis un peu envahissants, il ne faut pas perdre de vue que nous avons tous un ennemi commun, les gens malveillants qui utilisent les TIC à des fins délictueuses ou criminelles. Je salue au passage la création de l'office central de lutte contre la criminalité liée aux technologies de l'information et de la communication, dont la responsable est venue faire une conférence il y a quelques mois lors de la rentrée de l'ESAT.

L Des "cyber-vigies et des cyber-pompiers"

Quels que soient, en effet, les progrès que nous réaliserons en matière de sécurité informatique en amont, c'est-à-dire dans la conception, la réalisation et l'évaluation des systèmes de protection, je pense que la cuirasse comportera toujours quelques faiblesses que des pirates, qu'ils soient en culotte courte, en blouse blanche ou en tenue bariolée, tenteront d'exploiter à des fins ludiques, malveillantes, frauduleuses, cupides ou stratégiques.

C'est pourquoi l'autre volet de la SSI, celui du secours aux victimes, c'est-à-dire celui des "cyberpompiers" ou "cybervigies", doit également se développer. Nous avions pris quelque retard en France sur ce plan. Pendant longtemps, seul le CERT RENATER veillait à la sécurité du réseau et des centres informatiques du ministère de l'Education Nationale et de la Recherche. Puis vint ensuite un centre privé, le CERT IST mis en place par le CNES et quelques grandes sociétés.

En 1999, profitant honteusement de la crainte du bogue de l'an 2000 que certains s'acharnaient à élever au rang de cataclysme mondial programmé, je réussis à faire passer l'idée de la création d'un CERT qui travaillerait au profit de l'Administration dans son ensemble : je fis miroiter que celui-ci pourrait porter assistance aux sinistrés

du jour de l'an. Certes, il ne fut d'aucune utilité ce jour-là mais on n'a pas souvent une telle occasion, dans l'Administration, de pouvoir faire débloquer des fonds pour un projet utile dans un laps de temps aussi court !

Aujourd'hui, le CERTA existe et, si j'en crois le rapport du député Bernard Carayon, il rend d'éminents services. A présent, des sociétés se développent autour de prestations du même type au profit des entreprises, je ferai intentionnellement de la publicité à notre voisine, la société AQL, parce que nous sommes partenaires, avec SUPELEC et l'ENSTB, dans le montage du mastère SSI qui se déroule ici et qui connaît un grand succès. Je compte aussi sur le représentant de la DCSSI pour nous faire un point de situation sur la montée en puissance du réseau de confiance qui doit exister au sein de l'Administration.

J'ai parlé de cryptologie et de sécurité informatique et je n'ai encore rien dit de l'**anti-compromission électromagnétique**. Les spécialistes savent qu'il s'agit de la lutte contre le phénomène des signaux parasites émis par nos machines, ordinateurs, imprimantes, téléphones, fax, etc ... et qui peuvent, dans certaines circonstances, se révéler porteurs d'une information confidentielle. Je dirai que cette menace, dans un monde de connexion et de rayonnement volontaire à outrance –le règne du sans fil aussi bien en téléphonie qu'en microinformatique, le WIFI, etc...) apparaît bien secondaire dans les priorités. Il y a tant d'autres méthodes d'attaque plus facile à mettre en œuvre qu'il paraît possible de faire l'impasse sur cette vulnérabilité. Cependant, lorsque toutes les autres portes ont été verrouillées, et si l'enjeu en vaut la peine, ce mode d'action ne doit pas être négligé. On voit des pirates qui se donnent beaucoup de mal pour récupérer le code secret des cartes bancaires par exemple en dissimulant des micro caméras avec des émetteurs dans les distributeurs automatiques de billets. Pourquoi ne chercheraient-ils pas à récupérer ces codes en détectant les parasites émis par le clavier de ceux-ci ? D'ailleurs certaines méthodes d'attaque des cartes utilisant l'analyse de leur consommation électrique peuvent être classées dans ce que, au sein de l'OTAN, on nomme la menace "TEMPEST".

En fait, toutes les méthodes d'attaque physiques et logiques peuvent se combiner et c'est bien pour cela qu'il faut considérer un système dans sa globalité quand on veut le protéger. Je me souviens que M. MORENO, l'un des inventeurs de la carte à puce, était venu me voir début 2000, après avoir lancé de manière médiatique son défi concernant l'invulnérabilité de sa carte. Il offrait un million de francs à celui qui réussirait à percer son secret mais à condition de ne pas utiliser des moyens d'investigation autres que logiques et mathématiques. Comme si un cambrioleur se fixait des règles déontologiques et s'interdisait l'usage du chalumeau pour ouvrir un coffre ! Or, en moins de cinq minutes, une équipe spécialisée était capable de lire son code ...

Je pense avoir fait un rapide tour du domaine que vous allez continuer à approfondir tout au long de ces trois jours.

Pour ce qui est des enjeux, ce sera plus rapide. Je distinguerai les enjeux pour l'Etat, pour les entreprises et pour les citoyens.

M Les enjeux pour l'ETAT

Pour un Etat, la maîtrise de l'information conditionne sa liberté d'appréciation des situations et donc sa liberté d'action. Par conséquent, son système d'information, au sens large, ne doit souffrir aucune perturbation, aucune pollution et naturellement aucune intrusion. A contrario, l'Etat doit posséder les moyens de "compléter" son information en empruntant aux autres, en toute discrétion, voire en contrariant, le cas

échéant, le système d'information de ses adversaires. C'est donc un devoir pour lui de se mettre en posture d'assurer son **indépendance** et, si possible, sa supériorité dans ce domaine.

Or, aujourd'hui, sommes-nous capables d'assurer notre indépendance en informatique, domaine qui conditionne quasiment tous les autres ? Certes, nous exhibons de superbes engins dans le domaine aéronautique et spatial mais de nombreux composants sensibles intervenant dans leurs systèmes d'identification, de positionnement et de communication, systèmes sans lesquels un aéronef ne peut voler, ne sont-ils pas d'origine étrangère ? Ce besoin d'indépendance technologique doit impérativement être pris en compte si l'on veut parler d'Europe de la Défense.

En outre, le rôle d'un Etat n'est-il pas aussi d'assurer la protection et le bon fonctionnement des infrastructures du pays ? Le bogue de l'an 2000 a été l'occasion de recenser l'ensemble des systèmes critiques. Ce que le bogue n'a pas affecté fort heureusement, ne pourrait-il pas l'être par une bande de hackers résolus. Le Pearl Harbour informatique évoqué aux Etats-Unis bien avant le 11 septembre est toujours d'actualité : pourquoi serions-nous à l'abri d'un "Cyber Waterloo" ou d'un "digital Trafalgar" ?

Enfin et je m'arrêterai là, le rôle de l'Etat n'est-il pas d'assurer la sécurité publique en prévenant ou en permettant de châtier les crimes et délits commis contre ou en utilisant les technologies de l'information ? Après la période d'euphorie qui avait vu les apôtres de la nouvelle économie l'emporter sur les sécuritaires, le balancier s'est un peu inversé. Dans le débat sur la libéralisation de la cryptologie, après avoir été près de tout lâcher, les gouvernements semblent s'être ravisés et tentent de renforcer leur contrôle mais, sauf pour les Etats-Unis, dans de moins bonnes conditions qu'auparavant.

Je ne terminerai pas ce chapitre consacré aux enjeux pour l'Etat, sans rappeler cette citation du général de Gaulle : "*la défense est la raison d'être de l'Etat. Il ne saurait s'y soustraire sans risquer de se détruire lui-même*". Cette défense, qui doit prendre en compte les nouvelles menaces, impose aujourd'hui un effort considérable en faveur de la sécurité des systèmes d'information.

N Les enjeux pour les entreprises

Pour les entreprises, les enjeux dépendent de leur taille, de leur activité, de leur marché et, bien sûr, de l'âpreté de la concurrence. Lorsqu'un concurrent est capable, pour s'emparer d'un secret de fabrication, de prendre le risque d'une agression physique de grande ampleur avec camion bélier et commando nocturne, il est facile d'imaginer qu'il possède les moyens de se livrer au jeu des intrusions informatiques. Je fais là allusion à un cas réel qui a touché une grande entreprise dont la culture sécuritaire est bien établie. Que penser alors des entreprises qui, soumises à une concurrence comparable, ne subissent pas de telles attaques physiques ? Sont-elles véritablement hermétiques aux attaques informatiques ?

Quoiqu'il en soit, le risque pour une entreprise est évident : il va de la perte d'image, pas forcément agréable en cas de tagage du site Internet de la société, à la perte de marchés, jusqu'à à sa disparition pure et simple. Notons que sans être victime de malveillance, certaines sociétés mettent la clef sous la porte à la suite d'un sinistre informatique. Il y a quelques années, j'avais rédigé un article à la demande du rédacteur en chef de la revue "*Risques*", un magazine des assureurs, dans lequel je préconisais d'instaurer un bonus pour les entreprises utilisant des systèmes d'information sécurisés. J'ai eu le plaisir de voir que cette idée faisait son chemin, mais pas forcément en France.

Pour les entreprises qui travaillent dans les TIC, la sécurité des systèmes d'information est de plus en plus déterminante. La demande est de plus en plus forte et les clients de plus en plus exigeants. Les opérateurs de télécommunication comme les services postaux développent leur offre. La culture sécuritaire commence à se répandre même s'il faut parfois ramener à la raison certains paranos ou rectifier le besoin de sécurité affiché.

Pour les sociétés dont l'activité est la sécurité, parmi lesquelles je place les grands de l'électronique, toute l'industrie de la carte à puce et les sociétés de service ou de conseil, il semble que le marché ait commencé à décoller malgré une conjoncture sinistre pour les TIC. Pour ces sociétés, on peut prévoir encore de belles années compte tenu de la permanence des attaques mais surtout des infections virales qui font de gros dégâts et grèvent lourdement la productivité des systèmes et services informatiques des entreprises. Cependant, il se pourrait que l'activité des petites sociétés traitant de SSI soient progressivement ralenties, si d'aventure, les solutions sécuritaires élaborées par les grands groupes monopolistiques se révélaient finalement efficaces et acceptées par le public.

O Les enjeux pour le citoyen

Pour le citoyen, l'enjeu est d'abord de pouvoir se servir de son ordinateur sans se soucier perpétuellement de la mise à jour de ses "*patchs*" de sécurité, de ses antivirus, et sans être obligé de faire le tri des messages non sollicités qu'il reçoit. Ensuite, il veut avoir confiance dans les outils de protection que l'on met à sa disposition soit pour envoyer du courrier confidentiel, soit pour sécuriser ses transactions. Enfin, il souhaite protéger un minimum sa vie privée. Or, sur ce plan, la situation, comme je le disais en introduction, ne s'est pas beaucoup améliorée. Il est vrai que dans le contexte actuel, il est prêt à se laisser violer un peu si, en contre partie, cela lui procure davantage de sécurité pour lui et sa famille.

C'est pour cela que ceux dont la tâche est de veiller au respect de nos libertés individuelles et à la protection des données personnelles doivent se montrer doublement vigilants pour éviter les dérives et les abus. La commission informatique et liberté, la célèbre CNIL, souvent décriée pour les contraintes qu'elle impose à tous les concepteurs de systèmes informatisés, est notre rempart contre ces abus et ces dérives. Son action paraît excessive parfois parce que nous sommes dans une démocratie bien établie. J'ai personnellement toujours la crainte de changement pouvant survenir à la suite d'une grave crise et qui nous replacerait dans une situation telle que celle qu'ont connue nos pères. Je préfère qu'on ne facilite pas trop la tâche à ceux qui n'auraient pas la même conception que nous de la démocratie.

P Conclusion

En introduction, je vous ai dit mon plaisir de me retrouver parmi vous, les spécialistes de la sécurité des systèmes d'information, dont, malgré mon changement de fonctions, je me sens toujours très proche. A présent je voudrais vous dire ma satisfaction de constater que nous avons cessé de prêcher dans le désert. Certes, une foule de responsables n'a pas encore pris la juste mesure des enjeux de ce domaine en pleine évolution pour ne pas dire en pleine révolution. Et de nombreux problèmes n'ont pas encore reçu de solution. Néanmoins, je perçois comme un frémissement, pour plagier l'un de

nos anciens ministres. Certains dirigeants ont saisi que, sans SSI maîtrisée par nous-mêmes, il n'y avait pas de véritable indépendance ni de véritable liberté. Souhaitons qu'ils agissent avant qu'il ne soit trop tard. A vous, chacun à votre niveau et là où vous exercez vos talents, de les convaincre de le faire !

BGP et DNS : attaques sur les protocoles critiques de l'Internet

Nicolas Dubée

Secway/BD Consultants
ndubee@secway.fr

A Introduction

Le 11 septembre 2001 n'aura fait que renforcer un constat de nombreux gouvernements : une nation est tributaire d'un certain nombres d'infrastructures critiques telles que les pôles énergétiques, les moyens de communication et d'information nécessaires à la continuité économique et administrative du pays. Aux Etats-Unis, Internet a été jugé critique dès 1996 par l'administration Clinton. Une commission d'experts fut alors mandatée pour évaluer menaces et risques pesants sur la Toile. Deux acronymes sont revenus régulièrement tout au long des nombreux rapports en résultant : DNS et BGP. Ainsi, le groupe de hackers L0PHT déclarait-il en 1998 devant le Sénat américain pouvoir semer le chaos sur Internet en moins de 30 minutes, sans doute en référence à l'un de ces deux protocoles. Si DNS est maintenant maîtrisé en terme de connaissance des risques et contre-mesures, BGP relève encore du domaine mystique des protocoles au cœur d'Internet. Nous tâcherons donc dans cette présentation d'éclairer le lecteur quant au fonctionnement et aux vulnérabilités connues du protocole BGP. Nous en déduirons les risques liés à BGP et les pistes de sécurisation actuelles.

B Aperçu de BGP

Une des caractéristiques les plus innovantes des protocoles Internet a été l'arrivée du routage dynamique et complètement logique du trafic. Les acteurs clefs de ce concept sont les routeurs : véritables équipements réseaux destinés à aiguiller des paquets, les routeurs disposent de tables internes leur indiquant par où envoyer le paquet pour qu'il arrive au bon destinataire. Si cette tâche d'aiguillage est simple sur de petites topologies (on préconfigure le routeur pour envoyer le paquet où on souhaite), elle devient complexe dès lors que les routes peuvent changer en fonction de la vie du réseau : c'est le cas par exemple de réseaux connectés à d'autres par plusieurs chemins. On a donc introduit des protocoles permettant de reconfigurer automatiquement les tables de routage en fonction d'évènements tels que la perte de connexion sur une route ou l'équilibrage du trafic entre plusieurs routes.

Ces protocoles sont appelés IGP (*Interior Gateway Protocol*) lorsqu'ils fonctionnent sur des réseaux internes, ou EGP (*Exterior Gateway Protocol*) autrement. Les protocoles communément trouvés sont RIP, OSPF, ... Cependant, la topologie d'Internet a atteint une complexité telle qu'un protocole de routage aussi simple que RIP était impossible à utiliser. En effet, la cascade de chemins mais surtout de responsabilités administratives était telle qu'il a été décidé de regrouper les réseaux non plus d'après leur préfixe (c'est-à-dire d'après leur adresse réseau, combinaison adresse IP+masque

de réseau), mais en groupes de préfixes sous la même autorité appelés AS : Autonomous Systems identifiés par des numéros. Par exemple, un prestataire d'accès Internet dispose de son propre AS regroupant son réseau ainsi que les réseaux de ses petits clients. Chaque AS a alors la charge de dire au monde entier quels réseaux il héberge afin que les autres sachent par où accéder à ces réseaux.

Le protocole BGP fut conçu dans ce but, annoncer les réseaux et échanger les informations de routage entre AS. Pour ce faire, des sessions sont établies entre les routeurs externes d'un AS et ceux de ses voisins. Ces routeurs (appelés pairs ou peers) dialoguent par échange de messages au format BGP, dont le fonctionnement exact est décrit dans le RFC 1771 (la forme utilisée actuellement étant la version 4 depuis 1994-1995). Ces sessions BGP sont en fait de simples connexions TCP établies de manière permanente par un routeur vers le port 179 de l'autre et sur laquelle sont échangés 4 types de messages : Open, Update, Notify, Keepalive.

Alors qu'Open, notify et keepalive ne sont pas à proprement parler porteurs d'information de routage, le message Update sert à véhiculer les nouvelles annonces ou les préfixes à retirer.

En BGP, toute annonce porte sur un préfixe : on annonce au reste du monde l'arrivée sur Internet du réseau 12.34.56.0/24 par exemple. Le message est alors propagé après vérification aux pairs du pair, chaque AS traversé y ajoutant son propre numéro. Un chemin d'AS est construit avec la progression du message sur Internet : chaque annonce qui arrive sur un pair contient la liste des AS à traverser pour arriver au préfixe annoncé. Une table de routage globale est ainsi construite grâce au seul protocole BGP.

A l'heure actuelle, on estime qu'environ 120000 préfixes Internet sont annoncés par ces mécanismes. Aucune alternative n'est facilement possible puisqu'elle nécessiterait de revoir tout le mécanisme d'annonces, constitué uniquement autour de BGP.

C Vulnérabilités

On comprend donc bien que BGP est critique pour le fonctionnement d'Internet. On peut d'ailleurs à ce propos citer quelques incidents comme l'incident AS7007 en 1997 (du numéro de l'AS d'où provenait l'incident) qui a résulté sur une panne globale d'Internet pendant 2 heures. Cette panne était liée uniquement à une mauvaise configuration d'un routeur chez un hébergeur Internet en Floride, le mécanisme de distribution BGP entraînant une diffusion de l'erreur. En terme de vulnérabilités sécuritaires, les problèmes du protocole BGP sont nombreux et bien connus. On distingue en général les grands points suivants :

- Authentification : pas d'authentification requise entre les pairs. Même s'il est possible de configurer son routeur pour s'authentifier auprès de ses pairs, ceci est purement optionnel et de toute façon insuffisant puisque le canal TCP en lui-même n'est pas sécurisé.
- Autorisation : pas de mécanisme simple pour la vérification de la pertinence des informations trouvées dans les messages. Un routeur peut par exemple annoncer le préfixe qu'il souhaite, même un ne lui appartenant pas. Le pair recevant cette annonce n'a aucun moyen simple de faire la vérification, surtout s'il se trouve au cœur d'Internet, loin de la source de l'annonce.

Ces deux grandes classes de vulnérabilités sont amplifiées par le fait que comme vu précédemment, BGP ne dispose d'aucune vérification et sécurisation cryptographique du flux TCP : il est dès lors possible d'intercepter, modifier, insérer, rejouer, un flux BGP pour exploiter les deux grandes classes précédentes.

D Risques associés

Quels sont les risques associés à ces vulnérabilités? Les risques de déni de service sont évidemment les plus importants, mais on peut aussi imaginer l'altération de routes vers des serveurs ou réseaux sensibles dans un but de piratage (captation des flux notamment,...) Nous allons maintenant étudier les différents risques sous-jacents aux vulnérabilités exposées précédemment :

- Déni de service direct sur le routeur : tout routeur BGP peut être la cible d'un déni de service classique tel que le SYN flood. Le routeur est alors soit complètement planté, soit très lent. Dans ce cas, il ne peut indiquer en temps voulu à son pair qu'il est toujours en vie (message Keepalive). Dans ce cas, le pair considère que le routeur est planté, et purge les routes associées. Une nouvelle session est alors négociée par un des pairs, et l'attaque peut recommencer. La faisabilité d'une telle attaque est haute si le routeur n'est pas particulièrement protégé ou dimensionné, notamment au niveau de sa file d'attente des connexions en cours de négociation.
- Réinitialisation (RST) de la session BGP entre deux pairs : la session BGP étant transportée sur un flux TCP classique, on peut envisager la réinitialisation de cette connexion par envoi (spoof) de paquets RST savamment choisi. Il en résulte alors comme précédemment la purge des routes associées au pair, puis la tentative de redémarrage d'une session. Cependant, en terme de faisabilité, cette attaque nécessite la connaissance de nombreux paramètres : l'adresse IP des deux pairs, le port source de la connexion TCP, la fenêtre des numéros de séquence, et le TTL à 1 (ce qui peut être obtenu en voyant combien de hops on traverse avant d'arriver au routeur cible). Cette attaque est donc très délicate à réaliser mais théoriquement possible en aveugle, sans accès au réseau des pairs. Les paragraphes suivants montreront que même si l'attaque RST décrite ici est perçue comme simple à réaliser, elle est en fait très délicate à mettre en œuvre et bien moins préoccupante que d'autres.
- Spoofing complet de sessions : on crée en aveugle de toutes pièces une session de peering BGP avec une victime, vers laquelle on peut injecter des routes, en enlever,... En pratique, on se retrouve dans la même situation de faisabilité que l'attaque précédente, avec en plus la nécessité d'avoir le numéro de séquence TCP exact et tous les attributs BGP requis par le peer qu'on essaie de flouer. Cette attaque relève donc de la gageure, sachant qu'il faut en plus paralyser temporairement le peer qu'on essaie de spoofer.
- Hijacking de sessions existantes : variation de l'attaque précédente, mais au lieu de créer complètement une session, on réutilise une session existant entre deux peers dans laquelle on injecte des paquets de type update ou notify. Les paramètres à déterminer sont les mêmes que précédemment, même s'il ne suffit là que d'un paquet, la négociation TCP ayant déjà été faite par les peers officiels. On voit donc bien que les attaques en aveugle sont très difficiles à réaliser, en raison notamment du fait que BGP fait dialoguer entre eux des pairs préconfigurés : l'administrateur a préalablement réglé dans chaque routeur l'adresse du peer distant. L'attaque d'un point de vue technique se résume donc à un spoof TCP classique, maintenant très difficile à faire. On peut cependant imaginer que ces attaques soient réalisables si le réseau des pairs est écouté par le pirate. Dans ce cas, il lui vaudrait bien mieux attendre que l'administrateur se connecte à l'un des routeurs, la plupart du temps en telnet!

Les attaques suivantes nécessitent d'avoir accès à un routeur BGP officiel. Elles ne sont donc plus aveugles comme précédemment, mais sont beaucoup plus facilement réalisables. Il convient en effet de se souvenir que BGP est un protocole massivement distribué, où les pairs ne s'authentifient et ne s'autorisent que très faiblement. Le pirate peut donc compromettre un routeur BGP situé chez un hébergeur faible, pour l'utiliser afin d'envoyer de fausses annonces. Voyons les effets possibles de telles annonces.

- Désagrégation : en BGP, les préfixes plus spécifiques (c'est à dire plus longs) sont prioritaires. Ainsi, une annonce pour un réseau /24 à l'intérieur d'un /16 prendra précédence sur l'annonce du /16, et seule l'annonce /24 sera gardée pour le préfixe correspondant. Un agresseur peut donc complètement désagréger un grand réseau en envoyant des annonces pour des parties de ce grand réseau. Ces annonces plus spécifiques vont être prises en compte et le réseau va être complètement partitionné en fonction des annonces faites. Le risque associé est principalement un déni de service, mais on peut imaginer qu'un agresseur extraie d'un réseau important une petite plage contenant des serveurs sensibles, qu'il fait renvoyer chez lui (voir attaques suivantes). Notons à propos de la désagrégation l'expérience AS7007 en avril 1997 où un modeste prestataire en Floride a désagrégé pratiquement l'ensemble d'Internet, rendant le réseau des réseaux inopérant pendant deux heures et fortement perturbé pendant la journée.
- Injection de routes : les préfixes annoncés n'étant pas systématiquement vérifiés par le pair, il est possible d'annoncer à son pair un préfixe pour lequel on n'a pas autorité. Couplé avec le principe de priorité des préfixes vu précédemment, un agresseur peut annoncer le préfixe contenant sa victime. Tout le trafic venant de la zone touchée par l'annonce et destiné à la victime sera routé vers l'agresseur, qui aura tout loisir sur la suite : abandon complet (donc déni de service), Man-in-the-Middle avec manipulation des données avant réexpédition vers la victime, masquerade du serveur sensible, ... D'autre part, une attaque indirecte est possible : pourquoi ne pas annoncer par exemple le préfixe de plusieurs des root-servers DNS ? L'injection de route peut être moins ciblée et porter sur des blocs normalement d'utilisation bien spécifique comme les DUSA (blocs réservés tels que 192.168.0.0/16) ou même les blocs non alloués, entraînant une saturation des tables BGP sur les routeurs touchés et même des perturbations sur les réseaux utilisant un adressage Interne.

Le point le plus critique des attaques précédentes est qu'elles peuvent justement ne pas être des attaques ! En effet, la plupart des incidents répertoriés actuellement sur BGP sont des erreurs de configuration résultant sur l'injection de routes erronées ou la propagation d'annonces de réseaux privés. Notons finalement qu'il est justifié de s'interroger sur l'homogénéité du parc des core routers Internet. Une faute d'implémentation de la stack BGP sur ces matériels pourrait mettre à terre bon nombre d'entre eux, étant donné le peu de diversité du parc en terme de constructeurs différents. Un constat similaire a été fait début 2003 sur l'homogénéité logicielle des root-servers DNS : tous étaient équipés du logiciel BIND. Il a été décidé de passer un des root-servers sur NSD, concurrent de BIND ne partageant aucun code avec ce dernier.

E Conclusion et recommandations

Les risques liés à BGP sont donc autant des attaques que des erreurs de configuration. Les enjeux sont la survie de blocs entiers du réseau Internet, mais même si le

danger peut sembler énorme, la compétence globale des personnels aux échelons sensibles liés à BGP et les mesures prises actuellement sur la majorité des équipements devraient permettre de limiter l'impact de telles attaques.

Les alternatives à BGP sont pour l'instant trop complexes à déployer, et toutes les infrastructures nécessaires ne sont de toute façon pas disponibles. Citons cependant la piste la plus sérieuse pour le futur de BGP : Secure-BGP (S-BGP), actuellement en phase de maturation technologique. S-BGP authentifie les peers et sécurise le flux par un canal IPsec. D'autre part, chaque annonce est autorisée par un mécanisme de certification type PKI, dont les autorités pressenties sont les organismes de gestion des blocs d'adresses : ARIN, APNIC, RIPE, ... Quelques projets concurrents tels que SOBGP sont en lice mais il est difficile de dire si l'un d'entre eux peut faire face à S-BGP.

S-BGP n'étant pas encore utilisable en situation réelle, la sécurisation de BGP se résume à l'application des fameuses BCP (Best Common Practices) :

- forcer l'authentification MD5 des peers - filtrer les annonces venant de ses clients, pour vérifier qu'ils annoncent bien les préfixes leur ayant été assignés
- filtrer les annonces sortant de chez soit, pour vérifier que seules les classes possédées sont attribuées
- filtrer les annonces venant d'autres AS, pour vérifier si possible qu'elles correspondent bien aux peers, et sinon qu'elles ne contiennent pas de classes DUSA ou de blocs non assignés.
- protéger ses routeurs
- ne pas avoir de route par défaut
- faire le peering sur une interface secondaire ou sur un loopback.

Des Trappes dans les Clés

Éric Wegrzynowski

Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille
F-59655 Villeneuve d'Ascq Cedex
Eric.Wegrzynowski@lifl.fr

Résumé On mesure souvent la sécurité d'un système cryptographique à l'aide de la taille des clés utilisées. Cette mesure est-elle un véritable indice de sécurité ? Qu'elles soient publiques ou secrètes, des clés peuvent présenter des faiblesses volontaires ou non. Das le premier cas, les faiblesses peuvent provenir d'un mauvais générateur d'alea. Dans le second cas on parle de trappe.

A Introduction

Jamais il n'a été autant fait usage de cryptographie qu'aujourd'hui. Le développement des technologies de l'information et de communication a accru la nécessité de protéger les données. Diplomates et militaires, autrefois principaux utilisateurs, partagent maintenant ce besoin avec les entreprises et le grand public (cartes bancaires, sites Internet sécurisés,).

L'une des fonctions de la cryptographie (et historiquement la première) est d'assurer la confidentialité des données et des transactions. À cette fin, de nombreux systèmes de chiffrement ont été conçus, qui rendent inaccessibles l'information à toute personne non autorisée.

Conformément au principe énoncé en 1883 par Auguste Kerckhoffs, la sécurité d'un système de chiffrement ne doit pas reposer sur le secret de sa procédure, mais uniquement sur le secret d'un paramètre utilisé à chacune de ses mises en œuvre, paramètre appelé *clé*. Il est à noter que la plupart des systèmes de chiffrement utilisés de nos jours (du moins ceux à usage civil) respectent ce principe, les algorithmes les définissant étant publics (le cas du chiffrement utilisé dans le protocole GSM et celui du système de protection des œuvres cinématographiques sur DVD sont des exceptions récentes, qui ont connu le succès que l'on sait). L'élaboration de standards cryptographiques fait même appel à des évaluations publiques (c'est le cas de l'AES ([1]) aux Etats-Unis, et du projet NESSIE ([5]) en Europe).

La sécurité des systèmes de chiffrement repose donc d'une part sur la robustesse des algorithmes sous-jacents, leur résistance aux différentes attaques connues (cryptanalyse différentielle, linéaire, ...), et d'autre part sur la taille de l'espace des clés et la qualité de leur génération. C'est essentiellement à ce second aspect que nous nous intéresserons ici, en laissant de côté le problème de l'hypothétique faiblesse des algorithmes[1].

[1] D'autres facteurs interviennent aussi dans la sécurité que nous n'envisagerons pas ici (implémentations, matériels, utilisateurs...)

Après avoir évoqué l'importance d'utiliser un bon générateur d'alea, nous évoquerons la possibilité de concevoir des générateurs de clés délibérément biaisés. Nous montrerons sur des exemples comment on peut construire de tels générateurs aussi bien pour des systèmes de chiffrement à clé secrètes que des systèmes à clé publique, permettant à leurs auteurs de retrouver (assez) facilement les clés produites.

B Entropie des clés

On distingue deux familles de systèmes de chiffrement. Les systèmes symétriques, ou à clés secrètes, avec lesquels la même clé est utilisée aussi bien pour le chiffrement que pour le déchiffrement, et les systèmes asymétriques, encore appelés à clés publiques, qui se distinguent des systèmes précédents par l'utilisation de deux clés différentes : l'une, publique, sert au chiffrement, l'autre, privée, au déchiffrement.

Les clés des systèmes symétriques sont des chaînes binaires plus ou moins longues (56 bits pour le DES, 128 à 256 bits pour l'AES, 40 à 2048 bits pour RC4). Le plus souvent aucune contrainte n'est imposée pour ces clés : toute chaîne binaire peut convenir, et en choisir une peut se ramener à tirer à pile ou face autant de fois que le nombre de bits de la clé.

En revanche, dans le cas des systèmes asymétriques, les clés, même si on exprime aussi leur taille en bits, ne sont jamais des chaînes binaires quelconques. Au contraire elles possèdent une structure mathématique telle que la clé privée puisse déchiffrer toute information chiffrée avec la clé publique correspondante. Leur génération ne peut donc pas se limiter à une succession de tirages à pile ou face.

B.1 Cas des systèmes symétriques

Un système de chiffrement symétrique se doit d'offrir un espace de clés suffisamment vaste pour éviter une attaque par recherche exhaustive de la clé utilisée. De telles attaques ont été menées avec succès ces dernières années contre des systèmes utilisant de "petites" clés. Citons le cas des deux étudiants de l'école Polytechnique, parvenus en 1995 en utilisant les ordinateurs de l'école à mener à bien une recherche exhaustive dans un espace de clés de 40 bits (système RC4 utilisé dans Netscape) en une trentaine d'heures. Mentionnons aussi, l'attaque menée contre le DES avec un espace de clés de 56 bits par l'Electronic Frontier Foundation qui a construit en 1997, pour un coût d'environ 200.000 dollars, une machine (EFF DES Cracker ([3])) capable d'examiner plus de 92 milliards de clés à la seconde et parcourant donc l'espace des clés en un peu plus de 4 jours en moyenne.

Il est communément admis aujourd'hui qu'afin d'échapper à une recherche exhaustive, il faut utiliser des clés de 80 bits au minimum (l'AES propose des clés de 128, 192 ou 256 bits).

B.2 Cas des systèmes asymétriques

L'espace des clés d'un système asymétrique est plus complexe que celui d'un système symétrique, car les clés doivent posséder une structure mathématique bien précise. Les systèmes asymétriques reposent sur la notion de fonctions à sens uniques (dont l'existence n'a pas encore été prouvée), c'est-à-dire de fonctions faciles à calculer, mais calculatoirement difficiles à inverser. Ainsi, s'il est facile de multiplier deux nombres

entiers, il s'avère souvent très difficile de factoriser un nombre entier. Par exemple, le calul du produit des deux nombres de 78 chiffres

$$\begin{array}{c}
102639592829741105772054196573991675900 7 \\
165678080380668033419335217907113077 79 \\
\times \\
106603488380168454820927220360012878679 2 \\
079585759892915222706082371930628086 43 \\
= \\
109417386415705274218097073220403576120 0 \\
373294544920599091384213147634998428893 4 \\
784717997257891267332497625752899781833 7 \\
970765372440271467435315933543338 97
\end{array} \tag{1}$$

est quasiment instantané sur un ordinateur aujourd'hui, en revanche la factorisation du nombre de 155 chiffres ainsi obtenu a demandé en 1999 un calcul distribué sur 300 ordinateurs pendant plusieurs mois ([6]).

Pour le système RSA, la fonction à sens unique utilisée est l'exponentiation modulaire. Rappelons rapidement le fonctionnement de RSA.

La partie publique d'une paire de clés RSA est la donnée d'un nombre entier n (le *modulus*) produit de deux nombres premiers p et q distincts (non publics), et d'un nombre e premier avec $\varphi(n) = (p-1)(q-1)$ (i.e. $pgcd(e, \varphi(n)) = 1$). La partie privée d est l'inverse de e modulo n, (i.e. $d = e^{-1} \pmod{n}$).

Pour chiffrer un message m (supposé codé sous forme d'un nombre inférieur à n) avec la clé publique (n, e) du destinataire, l'expéditeur calcule le nombre $c = m^e \pmod{n}$. Des algorithmes efficaces de calculs d'exponentiation modulaire (*square and multiply*) font de l'opération de chiffrement une fonction facile à calculer. En revanche, le calcul de m en connaissant c, n et e uniquement est très difficile[2].

Le destinataire peut déchiffrer le message qu'il reçoit grâce à sa clé privée d en effectuant le calcul $c^d \pmod{n}$. Sa clé privée lui ouvre donc une porte dérobée permettant une inversion facile de l'exponentiation effectuée au chiffrement.

Pour qu'un système asymétrique résiste aux attaquants, il faut que le calcul de la clé privée à partir de la seule connaissance de la clé publique soit lui aussi difficile. Dans le cas de RSA, cela implique une difficulté de factoriser l'entier n, car si un attaquant parvient à factoriser n, il est alors en mesure de calculer la clé privée. Compte tenu des derniers records de factorisation (cf ci-dessus), on considère que l'entier n doit être d'une taille d'au moins 1024 bits (chacun de ses facteurs premiers ayant au moins 512 bits).

B.3 Des clés faibles pour RSA

En 1990, Michael Wiener ([8]) a découvert que certaines clés RSA sont faibles.

Les opérations effectuées par le chiffreur ne sont pas exactement les mêmes que celles effectuées par le déchiffreur. En effet, si elles sont de même nature (exponentiation modulaire), elles ne se font pas avec le même exposant (e pour le chiffrement, d pour le déchiffrement). La quantité de calcul à effectuer croît avec l'exposant. Un débat a eu lieu qui discutait de savoir lequel du chiffreur ou du déchiffreur devait effectuer le plus de calcul. Autrement dit, lequel des deux exposants e et d doit être le plus petit.

[2] du moins dans les connaissances actuelles

La découverte de M. Wiener consiste en une procédure qui permet de calculer la clé privée à partir de la clé publique, à condition que la partie privée soit inférieure à la racine quatrième du modulus de la clé publique (i.e. comprend quatre fois moins de chiffres que n). Cette procédure, s'appuyant sur le développement en fractions continues de e/n, est très efficace (cf une implémentation en MAPLE dans l'annexe A). Cette découverte a mis un terme au débat : la clé privée doit être grande.

B.4 Entropie

La taille des clés des systèmes de chiffrement est une condition nécessaire de sécurité, mais certainement non suffisante. Ceux-ci sont souvent accompagnés de générateurs de clés, et la sécurité de l'ensemble du système dépend aussi de la qualité de ce dernier, en particulier son entropie.

L'entropie d'un générateur est égal au nombre moyen de questions binaires (questions à réponse oui/non) qu'il faut poser pour retrouver une clé. On peut montrer que l'entropie est maximale lorsque la distribution de probabilités sur l'espace des clés induite par le générateur est uniforme. Et dans le cas de clés de systèmes symétriques, cette entropie maximale coïncide avec la taille des clés.

Un système se trouve affaiblit s'il utilise un générateur d'entropie non maximale. Cela peut se produire dans le cas où la génération de clés s'appuie sur un mot de passe, car l'entropie du générateur est alors égale à l'entropie de l'espace des mots de passe, le plus souvent bien inférieure à ce que laisse espérer la taille des clés. Un générateur peut voir son entropie diminuée par l'utilisation d'un mauvais générateur de nombres aléatoires qui induit une distribution non uniforme sur l'espace des clés.

Ces deux cas d'affaiblissement de l'entropie du générateur ne sont pas dûs à une intention des concepteurs et/ou programmeurs. Ils sont conséquence du mode de fonctionnement (cas de l'utilisation de mots de passe) ou du défaut du générateur d'alea utilisé, défaut qui peut très bien ne pas être connu lors de la conception. Il s'agit alors d'une faiblesse non intentionnelle.

En revanche, on peut imaginer qu'un producteur envisage délibérément d'introduire des faiblesses dans ses systèmes cryptographiques afin de pouvoir prendre connaissance de toute information chiffrée avec eux. Un exemple d'introduction de telle faiblesse a été révélée il y a quelques années, lorsque la presse a dévoilé que la NSA avait introduit dans les systèmes de chiffrement commercialisés par la société suisse CryptoAG un mécanisme qui révélait la clé utilisé dans le chiffrement.

Il est possible de concevoir des générateurs de clés dont l'entropie est considérablement affaiblie de telle sorte qu'elle rend envisageable une recherche exhaustive ou bien un moyen de calcul de la clé. Nous appelons *générateurs à trappe* ces générateurs intentionnellement affaiblis. Nous en présentons deux exemples.

C Trappes dans les clés secrètes

Nous allons illustrer sur un exemple un générateur de clés secrètes dont l'entropie est bien inférieure à la taille des clés qu'il engendre. Ce générateur ne couvre qu'une infime partie de l'espace de toutes les clés, et offre la possibilité de mener une recherche exhaustive de la clé utilisé pour chiffrer un message à qui en connaît le principe (i.e. la trappe qu'il contient).

Considérons l'espace de toutes les clés de n bits. En définition l'opération d'addition des clés comme le ou-exclusif bit à bit, icet espace possède une structure algébrique d'espace vectoriel sur le corps à deux éléments $\mathbb{F}_2 = \{0, 1\}$ de dimension est n.

Le générateur annoncé n'engendre que des clés d'un sous–espace vectoriel de dimension $k < n$ (autrement dit un $[n, k]-$code). Pour cela, il suffit de fixer une fois pour toute k clés $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$, linéairement indépendantes. Pour engendrer une clé $\mathbf{c}$ de n bits, il suffit de tirer au sort k bits $b_1, b_2, \ldots, b_k$ et de calculer la combinaison linéaire $\mathbf{c} = \sum_{i=1}^{k} b_i.\mathbf{c}_i$.

Par exemple, prenons $n = 10$ et $k = 4$. Fixons les quatre clés linéairement indépendantes

$$\mathbf{c}_1 = \texttt{100011001}, \mathbf{c}_2 = \texttt{010001011}, \mathbf{c}_3 = \texttt{001010101}, \mathbf{c}_4 = \texttt{000101100}$$

En supposant que les quatre bits tirés au hasard sont $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1$, la clé produite par le générateur est

$$\begin{aligned} b_1.\mathbf{c}_1 + b_2.\mathbf{c}_2 + b_3.\mathbf{c}_3 + b_4.\mathbf{c}_4 &= \texttt{010001011} + \texttt{001010101} + \texttt{000101100} \\ &= \texttt{011110010} \end{aligned}$$

Ce générateur n'engendre seulement que 2^k clés de taille n sur les 2^n possibles normalement. Si les coefficients b_i sont tirés avec une distribution uniforme, l'entropie du générateur est de k bits au lieu de n. Si $k \approx 50$, cette différence peut être exploitée pour quiconque connaît la base fixée, avec un effort comparable à celui menée par l'EFF contre le DES.

La trappe introduite dans ce générateur n'amène–t–elle pas des biais statistiques détectables ? Avec Vincent Benony à l'université de Lille, nous avons construit un tel générateur pour les paramètres $n = 128$ et $k = 50$ en choisissant correctement les k clés de base fixées. Nous avons généré un échantillon de 2^{25} clés, et effectué quelques tests statistiques classiques (distribution des poids (nombre de 1 dans les clés), répartition des clés octet par octet (une clé de 128 bits est constituée de 16 octets), collisions (répétition d'une clé dans l'échantillon)). Aucun d'eux n'a révélé le biais (cf quelques résultats à l'annexe B).

Cependant, il existe un moyen efficace de détecter le biais en exploitant la structure algébrique de l'espace des clés. En effet, si on génère $N > n$ clés de taille n avec un générateur d'entropie maximale, la probabilité que la matrice $N \times n$ obtenue avec une clé par ligne a un rang maximal (égal à n) avec une probabilité p égale à ([7])

$$p = \prod_{i=0}^{n-1} (1 - 2^{i-N}) > 1 - 2^{n-N} + 2^{-N}$$

On voit donc que la probabilité de ne pas avoir une matrice de rang maximal décroît exponentiellement avec N (avec $N = n + 10$, cette probabilité est inférieure à $\frac{1}{1000}$).

Si nous utilisons un générateur biaisé tel que celui décrit ci–dessus, il suffira de calculer le rang de matrices obtenues avec un peu plus de n clés pour constater qu'elles ne sont pas de rang n et ainsi détecter le biais.

Afin d'échapper à ce test du rang, on modifie simplement le générateur. Juste avant qu'il ne retourne la combinaison linéaire des clés de base, on la modifie en lui ajoutant un vecteur bruit choisi au hasard de poids limité t. Avec l'exemple précédent ($n = 10$,

$k = 4$), on calcule un vecteur bruit de poids maximal $t = 1$. Si on obtient par exemple $\mathbf{e} = \texttt{000010000}$, le générateur retoune la clé

$$\mathbf{c}' = \mathbf{c} + \mathbf{e} = \texttt{011110010} + \texttt{000010000} = \texttt{011100010}$$

L'introduction du bruit fait que l'espace des clés produites par ce générateur ainsi modifié, n'est plus un code linéaire (car plus nécessairement stable par addition des clés), mais le code engendré par cet espace est le code de dimension n, c'est à dire l'espace entier des clés de n bits. Ce qui explique que le test du rang ne puisse plus s'appliquer.

Le nombre de clés pouvant être produites par ce générateur dépend d'un paramètre lié au code sous-jacent : la distance minimale d séparant deux clés du code ([4]). Si celle-ci est est supérieure à $2t+1$, alors chaque clé ne peut être produite que d'une seule façon, et le nombre de clés est égal à $2^k . \sum_{i=0}^{t} \binom{n}{i}$. On en déduit que si la distribution de probabilité sur les vecteurs bruits est uniforme, l'entropie du générateur est alors égale à $k + \log_2 \sum_{i=0}^{t} \binom{n}{i}$ bits.

Nous avons repris notre générateur étudié ci-dessus ($n = 128$ et $k = 50$), et nous avons appliqué les mêmes tests avec $t = 1$ et $t = 2$ (l'entropie du générateur est alors de $57,01$ et 63.01 bits respectivement). Les résultats de ces tests n'ont rien révélé non plus.

D Trappes dans les clés publiques

Il semble bien plus difficile, voire impossible, de produire un générateur de clés publiques contenant une trappe permettant de calculer la clé privée correspondante. Il n'en est rien comme l'ont mis en évidence Claude Crépeau et Alain Slakmon ([2]) à propos des clés RSA.

Le générateur qu'ils ont conçu s'appuie sur les clés RSA faibles révélées par M. Wiener ([8]), c'est–à–dire celles dont la partie privée est trop petite. Ces clés faibles sont de probabilité trop faible pour être obtenues par un générateur non biaisé. Produire de telles clés serait ainsi facilement repérable. L'idée de Crépeau et Slakmon consiste à masquer des clés faibles pour les transformer en clés d'apparence ordinaire.

La trappe de ce générateur est tout simplement un entier pair fixé M dont la taille est celle du modulus n de la clé à engendrer. La génération d'une paire de clés RSA consiste à construire une paire de clés faibles $\{(n, e_1), d_1\}$ (vérifiant donc $d_1 < \sqrt[4]{n}$) telle que $e = e_1 + M$ soit inversible modulo $\varphi(n)$. Une fois une telle paire trouvée, le générateur produit la paire clé $\{(n, e), d\}$ (avec $d = e^{-1} \pmod{\varphi(n)}$) dans laquelle, avec une probabilité très forte, l'entier d est du même ordre que n.

Toute personne connaissant la trappe M est en mesure de calculer d à partir de n et e. En effet, il est facile de retrouver $e_1 = e - M$ et d_1 avec la technique de Wiener. Une fois connue une paire d'exposants public/privé, il est possible de factoriser le modulus n grâce à un algorithme probabiliste. La factorisation de n établie, il ne reste plus qu'à calculer l'exposant privé d correspondant à l'exposant public e (cf une réalisation de ce générateur et de l'utilisation de la trappe en MAPLE dans l'annexe C).

Ce générateur est donc bien la preuve qu'il est possible d'introduire une trappe même dans des clés aussi structurée que des clés RSA. Existe-t-il un moyen de détecter la trappe contenue dans ce générateur ?

La réponse affirmative a été trouvée par Serge Vaudenay en étudiant la distribution des réductions de e modulo un petit nombre premier qui divise $\varphi(n)$ (cette étude est tout à fait réalisable, étant donné que celui qui génère la paire de clés peut connaître la factorisation de n). Cette distribution se distingue de celle obtenue avec un générateur ordinaire.

E Conclusion

Aussi solides que peuvent être les algorithmes d'un système de chiffrement, celui–ci peut toujours être fragilisé par une production de mauvaises clés qui peut être involontaire (clés faibles, génération fondée sur un mot de passe, mauvais générateur d'alea), ou au contraire totalement délibéré (introduction d'une trappe dans le générateur).

Nous avons donné deux exemples de générateur à trappe, l'un pour le cas de clés de systèmes symétriques, l'autre pour le système RSA. Il y en a certainement beaucoup d'autres.

La possibilité de produire de tels générateurs est inquiétante pour les utilisateurs. Elle est un argument supplémentaire pour l'ouverture des codes des programmes informatiques, surtout ceux touchant à la sécurité des systèmes d'information.

Références

1. AES. Advanced Encryption Standard. `http ://csrc.nist.gov/CryptoToolkit/aes`.
2. Claude Crépeau and Alain Slakmon. The hidden small exponent method for generating pseudo-rsa keys. `http ://crypto.cs.mcgill.ca/~crepeau/RSA`.
3. Electronic Frontier Foundation. *Cracking DES*. O'Reilly, 1998. `http ://www.eff.org`.
4. F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
5. NESSIE. New European Schemes for Signatures, Integrity, and Encryption. `http ://www.cryptonessie.org`.
6. RSA155. Factorization of a 512-bits rsa key using the number field sieve. `http ://www.inria.fr/actualites/1999/rsa155.html`, 1999.
7. Antoine Valembois. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires*. PhD thesis, Université de Limoges, 2000.
8. Michael J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transaction on information theory*, 36(3), may 1990.

A La procédure de Wiener contre les clés faibles RSA

Afin de faciliter sa tâche dans les opérations de déchiffrement (ou bien de signature), un titulaire d'une paire de clés RSA peut souhaiter disposer d'un petit exposant privé d. En 1990, Michael Wiener a montré comment il est possible de calculer d en connaissant n et e seulement [8].

Cette attaque repose sur le développement en fraction continue du rationnel e/n. Si d ne dépasse pas la racine quatrième de n environ (c'est-à-dire si d a quatre fois moins

de chiffres que n), alors l'une des réduites du développement en fraction continue admet d pour dénominateur.

Voici un exemple traité en quelques secondes avec le logiciel MAPLE

```
# La clé publique est
> n := 11568742249488722994349005430362788259441140008903388165938 9\
         45823083913541544248694433175028270705735138390501731981 55\
         18219733441860767640867004225260526772403436953261919442 22\
         34070409320176244470672676087861485750413182473515607668 64\
         45038494073721266549156734426375138934805688512598474499 43\
         98392503648667529:

  e := 25721244518021878749600346452936422514825173496607690947 2104\
         58499822530829079073321948517571476394495871823168636529 62\
         92081764108064486119179289022290076518772648688080658257 85\
         12977269578183779400097761734183260692804643357850464093 27\
         06784208938537319946762765511360042518180491580881837672 77\
         1725313111888425:

# Développement en fraction continue de e/n et calcul des réduites
> convert(e/n,confrac,'reduites'):

# Chiffrement d'un message quelconque avec la clé publique
> M := 12345: C := M&^e mod n:

# Recherche de la réduite dont le dénominateur est d,
> i := 1:
> while C&^denom(reduites[i]) mod n <> M do i := i+1: od:

# La clé privée est
> d := denom(reduites[i]);

               5026679430400095888066820465589965505 93

# Vérification
> m := rand(0..n)():
> c := m&^e mod n:
> m - (c&^d mod n);

                                  0
```

B Un générateur de clés de 128 bits à trappe

Générateur sans bruit : La figure 1 ci-dessous montre la distribution des poids (nombre de bits à 1) dans un échantillon de 2^{25} clés obtenues avec une distribution uniforme dans un code de longueur 128 et de dimension 50. Cette répartition est tout à fait conforme à la distribution théorique (loi binomiale $\mathcal{B}(128, \frac{1}{2})$).

Les figures de la table 1 qui suivent, montrent pour chacun des 16 octets constituant une clé, la distribution des clés de l'échantillon selon la valeur de cet octet. On remarque une moyenne qui vaut approximativement 131000 et la théorie prévoit $\frac{2^{25}}{2^8} = 131072$.

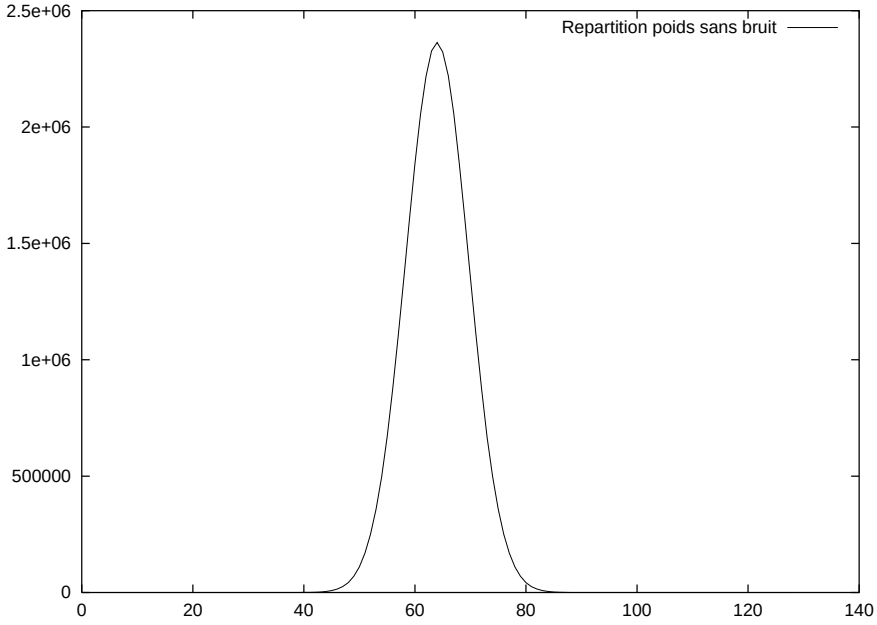
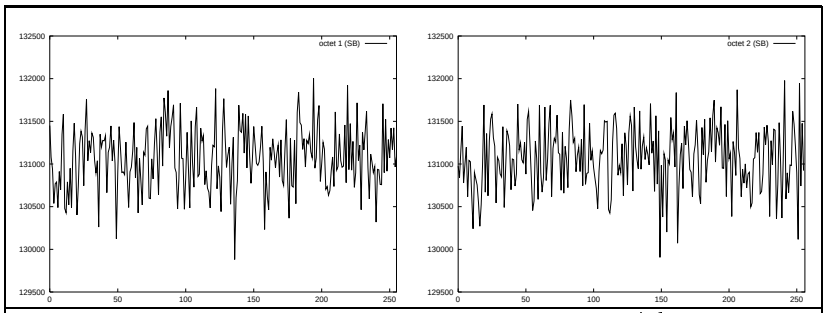


Fig. 1. Répartition des poids

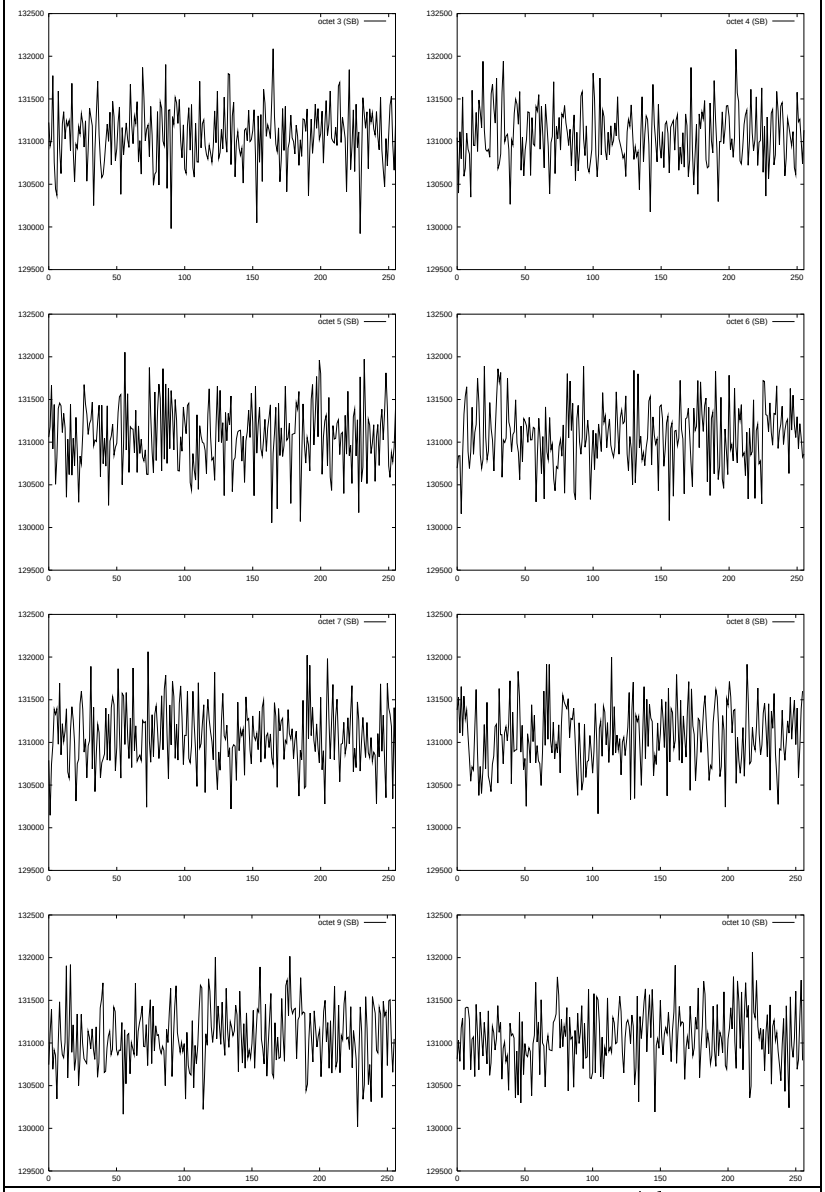
TAB. 1: Répartition des 16 octets



suite à la page suivante

TAB. 1: Répartition des 16 octets

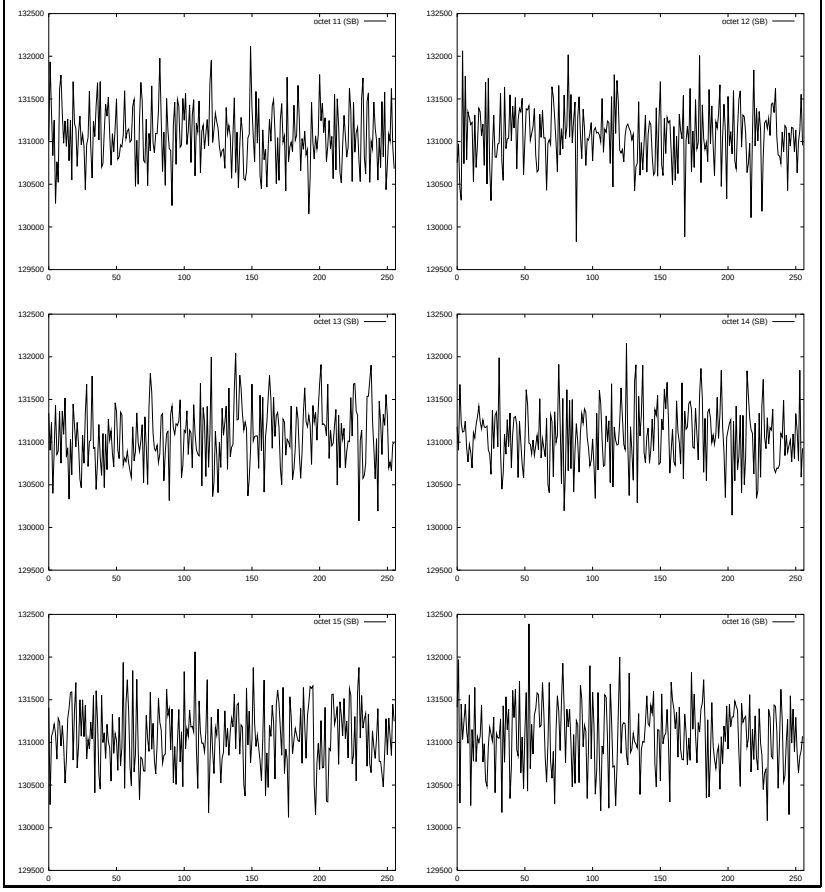
suite de la page précédente



suite à la page suivante

TAB. 1: Répartition des 16 octets

suite de la page précédente



Générateur avec bruit : Nous avons effectué des mesures analogues pour un générateur de clés choisies dans le même code bruité (avec des bruits de poids 1 et 2). Les résultats sont sensiblement identiques au cas sans bruit.

C Un générateur de clés RSA à trappe

On peut fabriquer un générateur à trappe qui produit des clés qui échappent en apparence à l'attaque de Wiener, sauf pour celui qui connaît la trappe. Ce générateur a été imaginé par C. Crépeau et A. Slakmon ([2]).

Voici le principe de la génération des clés avec le logiciel MAPLE :

```
# Initialisation du générateur de nombres aléatoires
```

```
> randomize():
# La taille des clés
> t := 1024:
# La trappe est un entier pair fixé inférieur à n
> M := 2*rand(2^(t-3)..2^(t-2))():
# On génère un modulus à partir de deux nombres premiers
> p := nextprime(rand(2^(t/2-1)..2^(t/2))()):
> q := nextprime(rand(2^(t/2-1)..2^(t/2))()):
> n := p*q:
> phi := (p-1)*(q-1):

# Puis on cherche un petit exposant privé
> d1 := 2*rand(2^(t/8-1)..2^(t/8))()+1:
> while igcdex(d1,phi,'e1')<>1 do d1 := d1+2;  od:
> while igcdex(e1+M,phi,'d')<>1 do
>   while igcdex(d1,phi,'e1')<>1 do d1 := d1+2; od:
> od:
# L'exposant privé est
> d := d mod phi:
# et le public est
> e := e1 + M:
```

Voici maintenant comment en connaissant la clé publique (n, e) et la trappe M on peut calculer la clé privée d.

```
# On commence par retrancher la trappe à e
> eprime := e-M:
# Ensuite on procède à l'attaque de Wiener sur la clé publique (n,e')
> convert(eprime/n,confrac,'reduites'):
> M := 12345: C := M&^eprime mod n:
> i := 1:
> while C&^denom(reduites[i]) mod n <> M do i := i+1: od:
# La clé privée est
> dprime := denom(reduites[i]);

          dprime := 6464948192129253280438336138206427965 35

# Ensuite, connaissant e' et d', on factorise n par une méthode probabiliste :
> u := eprime*dprime-1:
> while irem(u,2,'quot') = 0 do u := quot od:
> a := rand(2..n-1)(): while igcd(a,n)<>1 do a := rand(2..n-1)(); od:
> b := a&^u mod n:
> b2 := b*b mod n:
> while b2<>1 do
>   b := b2:
>   b2 := b*b mod n:
> od:

# on obtient les deux facteurs premiers
> p1 := igcd(b-1,n);q1 := igcd(b+1,n);
```

```
  p1 := 120328816104064584615853410051996559447373355205796383077\
        4542753034442728404920110514856119586932656913443763386080\
        490976884973517035845875132356193547179


  q1 := 104914027821708873342070815770733210136874164414257449402\
        6587007077346585764276789222815647080934040353167445328684\
        9045830490929788379445206857608471686497
# vérification
> n-p1*q1;

                                  0

# Il ne reste plus qu'à calculer l'exposant privé d correspondant à e
> phi1 := (p1-1)*(q1-1):
> igcdex(e,phi1,'d2'): d2 := d2 mod phi1:

#Vérification
> d-d2;

                                  0
```

Droit pénal et cybercriminalité : la répression des infractions liées aux TIC

Anne Cantéro

Cabinet Caprioli Avocats
9, avenue Henri Matisse,
F-06200 Nice
cabinet.caprioli-avocats@laposte.net

Pour des raisons techniques de dernière minute l'article d'Anne Cantéro figure en page 13.

Guerre de l'information et guerre informatique

Yves Correc

DGA/CELAR

A Introduction

L'exposé qui va suivre vise à jeter un autre regard sur la sécurité des systèmes d'information.

L'approche traditionnelle de la SSI est en effet trop souvent axée sur une défense statique, qui n'est pas sans rappeler une forme de « ligne Maginot informatique ». Mais de quoi faut-il se défendre ? Il est bien connu que l'homme est un loup pour l'homme. C'est donc de guerre que nous allons parler, et nous allons tenter de considérer le problème dans sa globalité, dans le cadre plus général de la guerre de l'information, sous les angles historique, technique, puis stratégique, avant d'en déduire une caractérisation de ce qui pourrait être la guerre informatique, dont une facette défensive est justement la SSI.

B Quoi ? Contexte historique

Quel est tout d'abord le problème ? Suivant la vision d'Alvin Toffler, nous pouvons parcourir à grandes enjambées l'histoire de l'humanité en la découpant en trois époques.

Durant l'ère agraire (des origines au début du 18ème siècle), le pouvoir appartenait à celui qui possédait la terre et les bras pour l'exploiter. La guerre se gagnait par attrition humaine. Enfin la réflexion militaire de Sun-Tzu est incontournable.

Durant l'ère industrielle, le pouvoir est passé à celui qui possède les moyens de produire en masse diverses ressources matérielles. Cette époque qui s'achève à la fin du 20ème siècle a vu la révolution industrielle, la mécanisation, la naissance des états-nations, et l'évolution de la guerre vers la destruction de masse. On peut mentionner Napoléon et Clausewitz dans le domaine militaire.

L'ère de l'information, qui est la nôtre, est le résultat de ce qu'il est convenu d'appeler la révolution du silicium. Celle-ci a permis le développement de systèmes informatiques qui constituent aujourd'hui une forme de système nerveux du monde moderne. Le pouvoir est passé à ceux qui savent utiliser et contrôler cette information, et au-delà des états vers des entités comme les multinationales. L'illustrent aussi bien le pouvoir des media que la délocalisation des tâches de production vers le tiers- monde. Si l'on doit citer un seul nom, mentionnons Libicki.

Ce découpage est bien sûr arbitraire et simplificateur, mais il est significatif, et souligne la révolution que nous vivons aujourd'hui. Cette révolution de l'information se caractérise par l'apparition d'une capacité inégalée, mais sans cesse croissante, à collecter de grandes quantités de données, à en extraire une information pertinente, à en déduire et partager une connaissance de la situation globale qui permettra ensuite de décider et d'agir au mieux.

Certains reconnaîtront ici le parcours de la pyramide de la connaissance, qui reste parfaitement intemporelle dans son essence. Gengis Khan communiquait déjà, mais

au moyen d'archers à cheval. Ce qui est nouveau, c'est le fantastique accroissement des capacités de communication et de traitement qui résulte de l'invention et du développement de l'électronique moderne. Qu'est-ce que cela va changer dans l'art du commerce, ou dans l'art de la guerre ? Simplement le fait que ces technologies vont augmenter considérablement l'efficacité de leurs utilisateurs, mais aussi les mettre à la merci de leurs défaillances : les gains comme les pertes en seront multipliés d'autant dans cette redite du vieux mythe de Faust.

Pour revenir à l'art de la guerre, il importe d'évoquer ici le concept de « révolution dans les affaires militaires » (RMA : revolution in military affairs). Une RMA correspond à l'apparition d'une technologie dont l'usage militaire bien compris confère à son inventeur un avantage décisif.

Les spécialistes s'accordent à distinguer trois RMA principales : l'armée napoléonienne, l'arme nucléaire, et enfin — nous y voilà — la collecte et le traitement massif de l'information sous toutes ses formes (corollaire militaire de la révolution de l'information précitée). Il est bien sûr possible de reconnaître des RMA de niveau plus bas, telles que l'introduction du couple char-avion (blitzkrieg), ou celle du porte-avions comme épine dorsale des flottes à la place du cuirassé, mais une grande constante de cette notion reste la nécessité de disposer à la fois de la technologie, et de la bonne doctrine d'emploi. L'histoire est là pour nous le rappeler.

Nous vivons donc que nous le voulions ou non la RMA consécutive à la révolution de l'information, et il importe d'en appréhender les conséquences.

C Pourquoi ? Contexte technique

Nul ne peut nier que les systèmes informatiques constituent de nos jours une sorte de système nerveux électronique des nations modernes. Ces systèmes, incontournables et omniprésents, ont rendu de ce fait les nations modernes vulnérables à des attaques ciblées d'un rapport efficacité/coût redoutable. Une analogie peut être trouvée dans le monde biologique avec les effets des gaz neurotoxiques.

Comment cela est-il possible ? Le dénominateur commun de tous ces systèmes est l'informatique, une informatique développée dans le cadre d'une économie de marché, où les critères conditionnant la survie des entreprises sont davantage les coûts, voire la mode, que la sécurité. Il en résulte que ces systèmes comportent de nombreux défauts, dont l'exploitation peut avoir des conséquences plus ou moins graves pour les systèmes eux-mêmes, l'information qu'ils traitent, et indirectement les éléments du monde physique qu'ils contrôlent.

Nous allons donc examiner, sur un cas d'école, les différents types d'attaques susceptibles d'être portées. Nous pourrons les classer suivant trois niveaux de gravité, de pénétration dans le système visé : l'écoute, l'intrusion, et la prise de contrôle (cette classification est parfaitement arbitraire, et n'a pour but que de clarifier le propos).

Le premier niveau se limite à l'écoute passive (et donc le plus souvent indétectable) des flots de données, des signaux physiques qui transitent sur les media utilisés. Ceux-ci peuvent être les réseaux informatiques, dont un hostile espionnera les flux après s'y être relié, aussi bien que les liaisons satellitaires, les faisceaux hertziens, ou les lignes téléphoniques. Dans tous les cas, des matériels adéquats existent qui permettent l'interception (on lira avec profit les rapports sur Echelon commandités par le parlement européen).

Le second niveau, plus actif, considère les diverses possibilités d'intrusion dans les systèmes visés. Cette intrusion peut avoir pour but de rechercher de l'information

(fouille), de la rendre indisponible en provoquant la panne du système support (attaques en déni de service), ou de s'insérer dans le dialogue entre acteurs du système (vol de session TCP/IP, interception GSM). L'intrusion peut se faire au niveau des machines informatiques elles-mêmes, en exploitant des fonctions cachées ou des comportements anormaux des logiciels. L'existence de ces portes dérobées résulte souvent d'erreurs de programmation, d'un certain folklore informatique (« œufs de Pâques »), ou d'une action délibérée (télémaintenance, espionnage, ...).

Le troisième niveau va plus loin, en exploitant la parfaite connaissance que peut avoir l'agresseur du système, pour en prendre le contrôle en vue de l'amener à lui fournir toute information souhaitée, ou de perturber les processus physiques qu'il pilote. Cette action repose sur l'usage de techniques sophistiquées, matérialisées dans un bestiaire informatique malveillant (chevaux de Troie, virus, zombies, ...).

Mais la technologie n'est pas tout, et il ne faut pas à ce stade oublier des méthodes aussi simples que le vol (portables, ...).

En résumé, le domaine de mise en œuvre des attaques est extrêmement large, depuis l'utilisateur humain jusqu'au composant électronique, en passant par les logiciels et les matériels informatiques. Nous illustrerons seulement les deux extrémités de ce spectre, avec une communication téléphonique suffisamment convaincante pour amener un administrateur système à prendre en toute bonne foi les mesures qui le mettront à la merci de son agresseur (action psychologique), et la possibilité d'implanter sur toutes les machines d'un réseau de PC le petit programme qui sur ordre effacera son BIOS (le logiciel qui lui permet de démarrer). Le problème peut prendre une ampleur catastrophique du fait que ce BIOS en théorie non effaçable réside le plus souvent dans un composant soudé sur la carte mère du PC, et que sa restauration risque d'impliquer le retour chez le constructeur de toutes les machines du réseau (action logicielle, avec effets physiques).

Le constat final nous est donné par Robert Stratton : *If houses were built like computers, the first woodpecker to come along would bring down civilization.*

D Pourquoi ? Contexte stratégique

Venons-en maintenant au volet stratégique de l'analyse. Pourquoi nos systèmes d'information sont-ils des cibles ? Quelles en sont les conséquences ?

Nous avons noté, au début de cet exposé, le caractère incontournable qu'ils ont pris dans les nations modernes. C'est justement cette fonction neurocorticale de traitement d'une information vitale qui fait leur force, et leur faiblesse.

Depuis sensiblement une décennie un courant de pensée issu des États-Unis a tenté de formaliser les réflexions sur ce thème autour du concept fédérateur de guerre de l'information. Ce terme est en fait une traduction inexacte de *information warfare* dont l'US Army donne une excellente définition : *actions conduites pour obtenir la supériorité dans le domaine de l'information, en altérant l'information adverse, ses processus basés sur l'information, et ses systèmes supports, tout en protégeant nos propres informations, processus et systèmes.*

Ce concept a connu de nombreuses déclinaisons.

Nous citerons pour mémoire le découpage très théorique entre guerre pour, contre, et par l'information.

Arquilla et Ronfeldt ont distingué le contrôle de l'information pour influencer la perception de la société (Net warfare), les pressions sur le système politique (Politi-

cal warfare), les mesures économiques (Economic warfare), et les opérations militaires (Cyber warfare). Ce découpage repose en fait sur les cibles envisagées.

Schwartau utilise un découpage par niveaux liés à la taille de l'entité cible, suivant qu'il s'agit de la personne (I), de l'entreprise (II), ou de l'état (III), reprenant pour ce dernier les quatre catégories mentionnées précédemment.

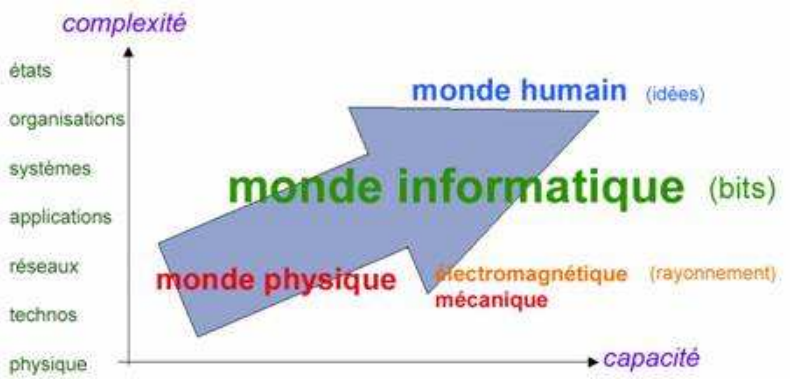
Libicki propose dans son texte fondateur de 1995 (*What is information warfare ?*) un découpage plus abouti, toujours suivant les cibles visées ou manipulées, en sept catégories : Command & Control Warfare (centres et liaisons), Intelligence Based Warfare (capteurs), Electronic Warfare (communications et signaux), Psychological Warfare (hommes), Hacker War (informatique, au niveau physique ou syntaxique), Information Economic Warfare (économie), Cyber War (informatique, au niveau sémantique). Il envisage ensuite la guerre de l'information comme une des trois composantes de l'*Information Dominance*, avec le renseignement et le *Command & Control*.

Cette supériorité dans le domaine de l'information est le but visé par Alberts, Cebrowski et al. dans la mise en réseau généralisée du *Network Centric Warfare*.

Citons enfin la vision chinoise atypique mais réaliste d'un champ de bataille sans limites (*Unrestricted warfare*) où interfèrent toutes les composantes, militaires ou non, du monde moderne.

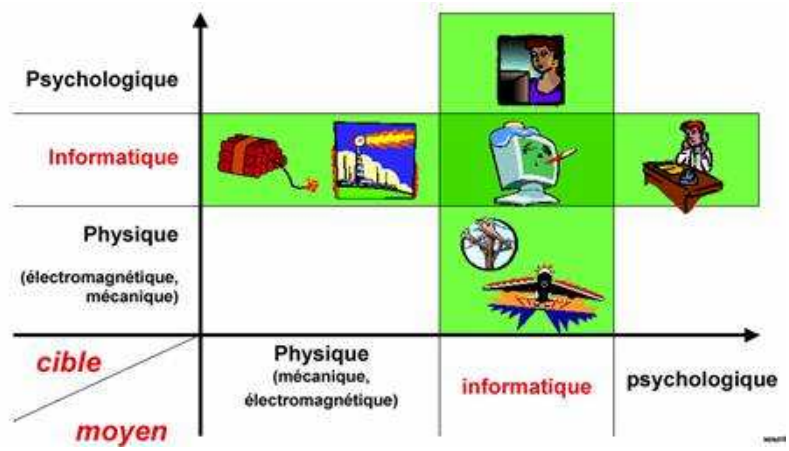
L'exposé des nombreux points de vue sur la guerre de l'information ne concourt pas à en clarifier la taxonomie, car l'omniprésence de l'information dans tous les processus n'en facilite pas vraiment la présentation... Que peut-on retenir de tout cela qui nous soit utile ?

Rappelons-nous que l'objectif visé est une meilleure compréhension de la SSI, et que le dénominateur commun sera par conséquent l'informatique, le moteur issu de la révolution du silicium. Nous allons pour simplifier les choses découper le monde en trois couches (pourquoi trois ? Parce qu'il faut bien trancher quelque part, et que ce découpage va faciliter notre exposé. Nous verrons plus loin qu'il est possible de le raffiner en fonction du domaine étudié). La première couche est celle du monde physique, mécanique ou électromagnétique. La seconde est celle du monde informatique, de l'information structurée manipulée par les systèmes informatiques. La troisième est celle du monde humain, des idées. Ces trois couches « à la ISO », physique, logique,



et psychologique, contiennent tous les objets du monde qui nous entoure, et sont le théâtre de toutes leurs interactions.

Nous pouvons alors en déduire une grille d'analyse en considérant ces objets (ou acteurs) comme des cibles ou des moyens d'action, suivant le schéma ci-après. Et nous appellerons *guerre informatique* l'ensemble représenté par la croix verte des actions de moyens informatiques sur des cibles de tout ordre, et des actions de tout ordre sur des cibles informatiques. Cette définition est plus large que la vision commune et



médiatique, qui se focalise sur le « cœur du métier » qu'est l'attaque informatique directe de cibles informatiques (vision qui repose implicitement sur des hypothèses fortes de connexion). Elle permet de représenter et de situer tous les enchaînements d'actions imaginables (actions physiques sur des calculateurs, social engineering, manipulation des forums Internet, action indirecte sur un objet du monde physique par la subversion des calculateurs qui le contrôlent, ...), ce qui nous sera utile plus loin.

Cette définition permet aussi, pour peu que l'on raffine le découpage en y introduisant des couches particulières, d'analyser d'autres notions telles que par exemple la guerre électronique (en séparant une couche « électromagnétique » de la couche physique), ou la guerre psychologique. On objectera que les croix qui les représentent se recouvrent partiellement, mais n'est-ce pas l'ambiguïté des frontières entre domaines (génératrice de querelles de clochers) qui est ainsi explicitée ?

Pour revenir à notre propos, c'est bien la possibilité d'enchaîner des actions complexes dans ce domaine élargi qui va donner tout son potentiel à la guerre informatique, en cherchant de manière systématique à exploiter l'interconnexion des systèmes entre eux, et avec le monde réel, pour obtenir des effets maximaux. Cette évolution va nous conduire des coups d'épingle appliqués par les hackers à des cibles informatiques « en ligne », vers les actions coordonnées du cyberterrorisme, utilisant tous les moyens disponibles contre des cibles bien définies telles que les infrastructures critiques des nations modernes.

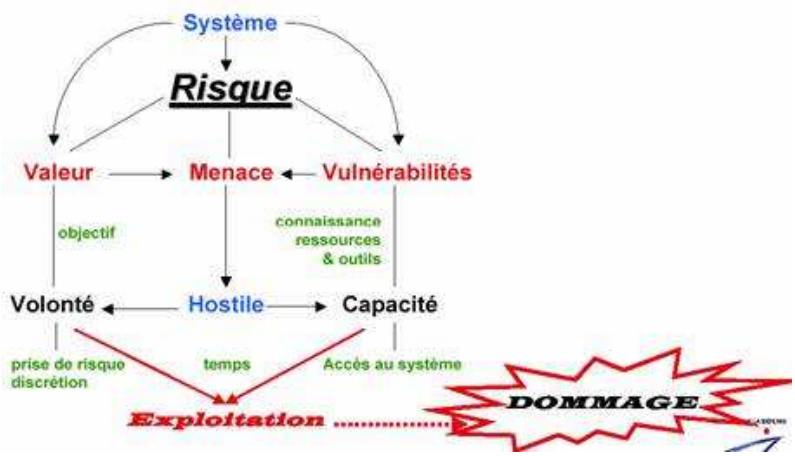
Le problème est-il réel ? Sans aucun doute. Quelle est son ampleur véritable ? Là est justement la question qui alimente aujourd'hui la réflexion.

E Comment ? Les éléments d'une attaque

Comment peut-on réaliser de telles attaques ? On observera qu'une bonne part de la réflexion est déjà conduite dans le cadre défensif de la SSI, dont il suffira de retourner le point de vue pour se mettre dans la peau de l'attaquant qui cherche à investir la place, et qui cherche plus précisément le chemin (d'attaque) qui le conduira jusqu'à sa cible finale.

Revenons à la notion classique du risque pesant sur un système informatique, dont les principales composantes sont :

- la valeur du système pour l'hostile, qui en fait un objectif potentiel, et lui donnera la volonté de l'attaquer pour peu que soient satisfaites des conditions minimales de discrétion ou de prise de risque ;
- les vulnérabilités du système, dont l'hostile peut avoir la connaissance. S'il a en outre les ressources et les outils pour les exploiter, ainsi que l'accès au système, il aura la capacité d'attaquer.



S'il dispose du temps nécessaire, l'attaque devient alors certaine, et la notion de menace peut alors être assimilée à une probabilité de réalisation compte tenu des éléments précités.

L'exploitation de cette définition du risque va nous permettre de mettre en évidence certaines composantes caractéristiques (information, moyens, temps, discrétion, risque, ...On peut en imaginer d'autres) du profil des agresseurs potentiels, et du profil d'attaques particulières. Reste à déterminer le chemin que suivra notre agresseur pour conduire son attaque.

Moyennant cette connaissance du système et des agresseurs probables, nous pouvons tenter d'aborder le problème de la façon suivante :

- une première analyse globale va rechercher les cibles potentielles précises. Une méthode d'exploration arborescente (« Branch and Bound ») typique de la recherche opérationnelle peut être envisagée, où l'on décompose le système dans le temps (attaquer à quelle étape du cycle de vie ?) et dans l'espace (attaquer quel

sous-système ?) pour énumérer celles-ci. On crée de la sorte un arbre d'attaque, dont les feuilles seront pondérées en termes de risque, de possibilité d'accès, de coût, ... pour l'agresseur. La mise en œuvre de cette méthode de séparation et d'évaluation (dont une ébauche a été décrite par Schneier et al.) nous conduit alors à l'analyse plus fine de l'arbre élagué en fonction des objectifs et capacités de l'agresseur ;
- cette seconde analyse locale devra déterminer précisément les étapes de la progression de l'attaque pour chacune des feuilles restantes, au rapport efficacité/coût suffisant.

Le recours au modèle en couches du monde s'avère utile à ce stade, car il permet de modéliser tout type ou enchaînement d'actions, et pas seulement informatiques stricto sensu, ouvrant ainsi la porte à l'emploi de techniques de simulation pour étudier l'ensemble des phénomènes concernés. Différents types de modèles sont envisageables en fonction de la granularité nécessaire dans la représentation et les résultats, depuis les réseaux de Petri des premières expériences, jusqu'au simulateur développé par la DGA et EADS/Sycomore (Castor).

L'approche évoquée ici représente une évolution nécessaire (et non évidente) de la SSI, qu'un véritable outillage, à l'instar d'autres disciplines techniques qui usent abondamment de la simulation, doit permettre de mieux intégrer aux phases de conception et de développement des systèmes (et sortir du syndrome de la « ligne Maginot informatique ».).

F Conclusion : SSI ou guerre informatique ?

Le monde où nous vivons aujourd'hui est dominé par l'information, information devenue vitale pour le fonctionnement de tous ses rouages. Elle est devenue l'objet de toutes les convoitises, et le levier de tous les cataclysmes, levier dont le bras est de plus en plus informatique. La menace correspondante est souvent sous-estimée, ou mal appréciée quant à son origine, ses moyens d'action, et ses conséquences.

La place tenue de nos jours par les systèmes informatiques sera donc, si elle ne l'est pas déjà, décalquée dans les termes d'une guerre dont ils seront les cibles et les moyens. Cette nouvelle forme de guerre appelle de nouvelles déclinaisons des concepts habituels de renseignement, de préparation, de conduite d'opérations, et de défense. Dans ce cadre, et si l'on veut bien considérer que le volet défensif correspond à peu près à la SSI, celle-ci devra couvrir non seulement les aspects relevant de la défense passive, c'est à dire de l'assurance de sécurité aux bases méthodologiques traditionnellement fortes, mais aussi la défense « active », dont les composantes de protection, détection, et réaction garantissent l'efficacité. La SSI devra par conséquent évoluer pour intégrer cette dimension nouvelle, en évitant si possible de trop s'interroger sur le bout par lequel sera ouvert l'œuf.

Un dernier mot pour ne pas oublier l'homme, incontournable et faillible artisan, et le temps qui ruine tout.

Index des auteurs