NOUVELLE FORMULE - NOUVELLE FORMULE

44

JUILLET/AOÛT 2009

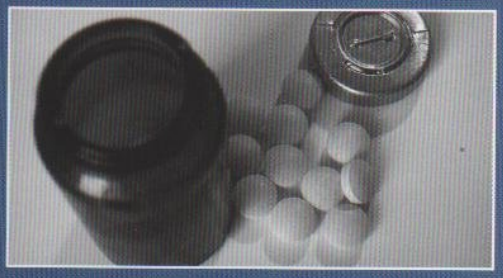
MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

SOCIÉTÉ / *SPAM*

Le business de la vente de Viagra online



DOSSIER

COMPROMISSIONS ÉLECTROMAGNÉTIQUES

Quand vos machines diffusent vos données à votre insu

- Propagation des signaux par voie hertzienne
- Utilisation de matériel « TEMPEST »
- Lutte contre les SPC

APPLICATION / *WIKI*

Travail collaboratif : La sécurité et la confidentialité avec DokuWiki

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

L 19018 - 44 - F: 8,00 € - RD

RÉSEAU / *ROUTAGE*

Sécurité d'OSPF : Attaque sur les numéros de séquences cryptographiques

SCIENCE & TECHNOLOGIE

Cryptographie par courbes elliptiques et attaque par canaux auxiliaires sur systèmes embarqués

ÉDITO : Schtroumpf Grognon

Parfois, il fait beau, les oiseaux chantent et le ciel est bleu. Et puis parfois, il y a des jours où faut pas m'ennuyer (je reste poli, merci maman). Et en ce moment, il y a des jours... tous les jours.

En ce moment, les lois autour d'Internet, c'est un peu comme les épisodes *d'Amour, gloire et beauté* : on a l'impression de voir tout le temps les mêmes.

Je ne suis pas juriste pour deux sous, mais à quoi ça sert d'empiler les lois, puis de les modifier, et ce, en permanence, alors que, déjà, les premières ne sont pas appliquées ? Voire que les anciennes sont suffisantes !

Je repense aux commentaires, sur la décision du Conseil constitutionnel à la suite de « l'acceptation à 90% (1) de la loi HADOPI » (à la différence de Madame le Ministre qui se réjouit de ce score, pour ma part, c'est de l'ironie), du conseiller spécial de notre président, Henri Guaino, mais surtout à l'analyse juridique de Maître Eolas [1].

Notre conseiller spécial dit ne pas comprendre qu'un article de la Déclaration des droits de l'Homme soit un article de Droit (oui, oui, pourtant, c'est écrit dans le titre de ladite déclaration). Il soupçonne également les révolutionnaires auteurs du texte de ne pas avoir pensé à inclure Internet en rédigeant l'article 11 (la libre communication des pensées et des opinions est un des droits les plus précieux de l'Homme). Rendons-lui justice, ils n'ont pas non plus pensé à la radio, à la télé, à la télépathie et aux autres moyens de communication à venir. Mais, est-ce grave ?

Comme le souligne avec brio Maître Eolas, cette déclaration a un caractère intemporel qui fait qu'elle traite de la liberté de communication sans tenir compte du canal employé. Et c'est ça, je trouve, qui la rend belle et puissante : elle s'abstrait du temps qui passe.

Est-ce que cette loi – peu importe la forme qu'elle prendra – sera efficace ? Tout le monde sait déjà que non, tellement elle est facile à contourner, techniquement et juridiquement. Prenons un ordinateur sur lequel tourne une machine virtuelle dans laquelle tourne le mouchard chargé de vérifier qu'on ne télécharge rien d'illégal. Pendant ce temps, à Vera Cruz, sur un autre ordinateur ou sur l'hôte, eMule voisine avec BitTorrent et quelques serveurs de news. Malheureusement, un mail d'avertissement arrive, puis une convocation. Pas de souci, ayant suivi scrupuleusement la loi, il suffit de donner les logs de la machine du mouchard pour être acquitté, car l'infaillible mouchard n'aura rien vu.

Mais pire : LOPPSI 2 (le retour), avec son filtrage du net et ses perquisitions virtuelles. Personne ne remet en cause la nécessité avancée de lutter contre la pédophilie ou le terrorisme, mais seulement les moyens de cette lutte.

Imaginons un pays merveilleux où les fournisseurs d'accès (FAI) sont tenus de bloquer l'accès à certains sites, sites déterminés par les autorités via une liste tenue secrète. Ces FAI ont l'obligation légale de le faire. En revanche, rien n'interdit d'utiliser ce même moyen de filtrage pour bloquer la VoD de chez son concurrent, puisqu'il n'y a aucun contrôle sur le filtrage.

Pour la perquisition numérique, le défi est impressionnant. Puisque nous sommes dans de la science-fiction, poursuivons. Imaginons ce qui n'arrivera jamais : les méchants découvrent ce *trojan* gouvernemental. Quelles conséquences ? Toutes les personnes qui ont cherché à analyser des Torpig et bestioles similaires le savent bien : on peut reproduire les techniques employées pour construire son propre trojan qui aura le même risque de détection, c'est-à-dire faible. Plus marrant, on peut remonter parfois sur le serveur de contrôle ou exploiter des failles pour voir alors des malwares se manger les uns les autres. Et là, je me contente des risques techniques.

Alors oui, bien sûr qu'il faut lutter contre la pédophilie et permettre aux services de police/renseignement de nous protéger. La sécurité est toujours vue comme une entrave aux libertés, tous les RSSI vous le diront. Est-ce pour autant vrai ? Alors qu'il est possible (mais pas facile) de sécuriser correctement nos réseaux sans restreindre les libertés de nos utilisateurs, pourquoi ne pourrions-nous pas reproduire cela pour notre société ? Peut-être parce que ceux qui y parviennent sont des spécialistes de la sécurité et non de l'agitation médiatique.

Au final, on place donc un contrôle chez l'utilisateur (HADOPI), un chez le FAI (LOPPSI2) et, en bonus, on a des lois jugées sur la quantité, un conseiller spécial qui devrait peut-être prendre des cours de droit, et un texte en projet sans étude d'impact : en ce moment, il y a des jours tous les jours...

Juste pour rire, j'imagine bien le mouchard gouvernemental, utilisé pour les perquisitions, bloqué par le logiciel bonne conscience d'HADOPI...

Bonnes vacances quand même, malgré l'avis de TEMPEST.

Fred Raynal

(1) On notera la pertinence du quantitatif dans ce domaine : on juge du succès de l'adoption d'une loi au poids de ce qui est validé par le Conseil constitutionnel, en faisant fi du qualitatif, c'est-à-dire ce qui faisait le cœur de la loi en question.

[1] http://www.maitre-eolas.fr/2009/06/16/1451-prix-busiris-a-henri-guaino

Rendez-vous au 4 septembre 2009 pour le n°45 !

www.miscmag.com

MISC est édité par
Les Éditions Diamond
B.P. 20142 / 67603 Sélestat Cedex
Contact commercial : contactez-nous par notre site www.ed-diamond.com
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : www.miscmag.com www.ed-diamond.com

IMPRIMÉ en Allemagne
PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036

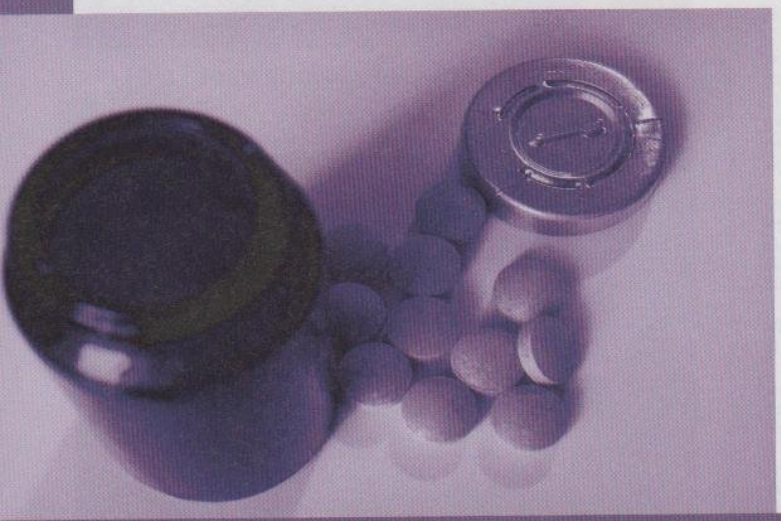
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 8 Euros

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 : contactez-nous par notre site www.ed-diamond.com
Service abonnement : abo@ed-diamond.com
Impression : VPM Druck Rastatt / 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

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 leurs 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



Guillaume Arcas
guillaume.arcas@gmail.com
David Lesperon
david.lesperon@gmail.com

ALL YOUR PILLS ARE BELONG TO US (1/2)

mots-clés : *spam / contrefaçon / cybercriminalité / écosystème / modèle économique*

« Tout est vrai. Rien n'est vrai. C'est un roman ». (Serge Bramly)

À l'instar de ceux qui opèrent dans les rues, les groupes criminels qui, comme les pédophiles néo-nazis [1] et autres psychopathes racistes [2], sont passés à l'ère du 2.0 forment un écosystème complexe. C'est ce que nous nous proposons d'effleurer dans cet article basé sur des faits réels. Afin de présenter au lecteur un ensemble cohérent et afin de ne point heurter certains protagonistes, des noms ont été modifiés ou masqués et des faits appartenant à des « affaires » différentes ont été réunis en une même histoire. Toute ressemblance avec des personnes physiques ou morales existant ou ayant brutalement cessé d'exister ne serait donc que pure coïncidence. Ou pas.

1. Introduction

« Quinze personnes ont été hospitalisées, dont trois dans un état grave, suite à l'absorption de médicaments achetés sur Internet. »

La nouvelle faisait les gros titres de la presse nationale et européenne. James Phelps avait écourté ses congés d'été pour participer à la cellule de crise mise en place par les laboratoires Lullu Sanizer. Après quelques années passées dans divers « services » gouvernementaux, il avait rejoint le géant pharmaceutique dont il dirigeait depuis trois ans le Département de lutte contre les contrefaçons (DLC). Traquer les fabricants et les revendeurs de produits contrefaits était depuis devenu son quotidien.

2. Lundi 03 août

Sur le bureau de Phelps étaient posées 5 plaquettes de Cyagra, le produit phare des laboratoires. Pour un profane – c'est-à-dire un consommateur – rien ne distinguait les comprimés : tous portaient le logo bien visible de Lullu Sanizer. Seul un œil expert aurait pu remarquer l'absence d'étiquette holographique sur les boites contenant des produits contrefaits.

« Notre problème est le suivant », expliqua le responsable du laboratoire d'analyse interne, « quatre de ces produits sont des faux, mais tous proviennent d'une de nos usines de fabrication. Ils ont été achetés sur le même site Internet, à quelques semaines d'intervalle et, pour corser l'affaire, leur analyse chimique révèle qu'ils ont une composition qui n'est aujourd'hui que légèrement différente de celle du produit original. Rien de très important, quelques milligrammes en plus ou en moins, mais cette différence peut être mortelle pour nos patients. »

« Vous dites 'aujourd'hui' ? », s'enquit Phelps.

« Exact, les premières contrefaçons ne contenaient même pas le principe actif de l'original, ni rien de véritablement actif en fait, et une simple réaction colorimétrique suffisait à les différencier de notre produit. Aujourd'hui, des techniques comme la chromatographie en phase liquide à haute performance sont nécessaires. »

Le directeur adjoint prit alors la parole et s'adressa à Phelps : « Contrairement aux cas plus classiques que vous avez eu à traiter jusqu'à présent, nous vous demandons la plus grande discrétion dans cette affaire : si un de nos sous-traitants est impliqué dans ce trafic, nous souhaiterions savoir à quel niveau et nous voulons en apprendre le plus possible sur la filière d'écoulement des produits.

Notre antenne de Singapour prendra en charge le volet industriel de cette fraude, nous vous demandons d'enquêter sur le site qui met en vente ces produits sur Internet. Vous serez aidé par Cyril Hique, un de nos experts Sécurité Internet. »

« Bonjour Monsieur Phelps. Nous avons obtenu d'une des victimes – monsieur M. – d'avoir accès à son ordinateur, en échange de la prise en charge par les laboratoires de ses frais d'hospitalisation. Cela devrait nous permettre d'en savoir plus sur cette fraude. »

Les affaires démarraient toujours à peu près de cette manière : un briefing rapide, une chemise, rouge, pour souligner l'urgence de la situation, contenant un mémo, tendue d'un mouvement de bras énergique, un ou deux éléments désignés pour l'épauler. Il aimait cet instant : il annonçait la traque.

« Cyril, je vous propose de faire un point lundi prochain sur cette affaire. »

3. Lundi 10 août

« Bonjour Monsieur Phelps. L'analyse de l'ordinateur de monsieur M. a mis en évidence des détails intéressants. Monsieur M. achète régulièrement ses produits sur le site de notre partenaire Acadian Pharmacy. Il n'a donc pas été étonné de recevoir un courriel l'informant d'une offre promotionnelle ».

Pharmacy online. Ship from Canada...save big
http://dom.ir/735

Fig. 1 : Un courriel minimaliste

Fig. 2 : La « pharmacie » en ligne utilisée par monsieur M.

CNN.com THE DAILY TOP 10

TOP 10 STORIES

#1. It's a buyer's market if you know what 'code words' to look for.

2. "Yard Work and Leather" is just one of many candle scents for men.
3. It's a buyer's market if you know what 'code words' to look for.
4. Pool Parasite Infections on the Rise
5. Boys bounce for 24 hours in world record attempt
6. Pool Parasite Infections on the Rise
7. Superheroes Get Sandy
8. Furnished Nazi bunkers surface in Denmark
9. War, Spying and Party Game Delusions
10. Half-scale replica of German tank built for paintball competition.

TOP 10 VIDEOS

#1. Russian stocks take hit as govt. looks to nationalize steel, oil companies.

2. Tropical Storm Edouard moving toward Texas coast
3. Christina Applegate treated for breast cancer
4. Police killed in west China ahead of Games
5. Bikers down to bare basics for eco demonstration
6. Olympic Sport: Blocking the Internet
7. Guinea Pigs Get Dressed ... and Eaten
8. GPS-equipped turtle stumbles upon field of marijuana in a D.C. park.
9. Olympic Sport: Blocking the Internet
10. Half-scale replica of German tank built for paintball competition.

Fig. 3 : Un exemple de pourriel du mois d'août

Ce qui frappe, c'est son esthétique pour le moins... minimaliste (Fig. 1) :

« Vous êtes en train de me dire que monsieur M. a cliqué puis acheté après avoir reçu ça ?! Notre département Marketing va être heureux de l'apprendre... »

« Il faut en effet croire que les mots *'save big'* ont fait leur effet. Le lien pointe sur la page suivante (Fig. 2) : »

« Dites-moi Cyril, nous savons qui a envoyé le spam et nous avons aussi l'adresse du site. Je vais peut-être vous paraître naïf, mais ne peut-on pas dès lors lancer des poursuites contre ces deux sources ? »

« Comme vous vous en doutez, ce n'est pas si simple. J'ai récupéré d'autres pourriels, envoyés début août par la même machine, grâce à des amis qui me font suivre les messages indésirables qu'ils reçoivent. Par chance, j'en ai retrouvé d'autres, d'un tout autre genre, comme celui-ci par exemple (Fig. 3) :

Voici les en-têtes du message (Fig. 4) :

```
Return-Path: <Navraj-atsaksru@breining.edu>
Delivered-To: michel.monpain@blueblue.com
Received: (qmail 4457 invoked from network); 5 Aug 2008
06:54 -0000
Received: from 82.64.73.194 (HELO lns-bzn-21-82-64-73-194.
adsl.proxad.net) (82.64.73.194)
 by smtp-in.blueblue.com with SMTP; 5 Aug 2008 06:54} -0000
X-mailed-to: michel.monpain@blueblue.com
X-To: cnn-dailytop10#*#michel.monpain@blueblue.com
X-job: 20080801155902.cnn-dailytop10.5866
Message-Id: <20080801155902.cnn-dailytop10@mail.cnn.com>
From: "Daily Top 10" <Navraj-atsaksru@breining.edu>
To: michel.monpain@blueblue.com
Date: Tue, 5 Aug 2008 08:53:40 +0200
Subject: CNN.com Daily Top 10
Content-type: multipart/alternative;
boundary="605umpjwo610"
MIME-version: 1.0
```

Fig. 4 : En-tête du pourriel du mois d'août

« Regardez la ligne X-job. Cet en-tête n'est pas courant. Il semble avoir été ajouté pour marquer le message. On le retrouve dans tous les spams envoyés début août (Fig. 5) : »

```
X-job: 20080801155902.cnn-dailytop10.7249
X-job: 20080801155902.cnn-dailytop10.8999
X-job: 20080801155902.cnn-dailytop10.1593
X-job: 20080801155902.cnn-dailytop10.0888
X-job: 20080801155902.cnn-dailytop10.3387
```

Fig. 5 : En-têtes « X-job » extraits de différents pourriels

« Après X-job, on dirait une date, non ? Mais, elle est antérieure à celle du message de presque quatre jours. »

« Exact, M. Phelps. Je pense qu'il s'agit, comme son nom semble l'indiquer, de l'identifiant d'un 'travail' programmé à l'avance. On retrouve la même structure : horodatage, sujet du message (cnn-dailytop10 en l'occurrence) et un numéro qui pourrait être l'identifiant de la machine qui a envoyé le spam.

Quelques jours après, les sujets des messages de cette série étaient similaires, mais ces derniers ne contenaient plus d'en-tête X-job. »

« Vous dites 'cette série', mais qu'est-ce qui vous permet de dire qu'il s'agit bien de la même ? »

« Eh bien, mis à part les titres similaires, et les pointes du même humour dans les accroches, je me suis entre autres appuyé sur les adresses source des envois. Ainsi, à l'aide des pourriels, j'ai pu établir un lien entre un sujet de message et un groupe d'envoyeurs et, si ce même groupe de machines envoie un message différent peu de temps après, il m'est possible de faire le lien entre deux campagnes de « spam » au travers des exécutants. »

« Bien vu. »

« Intéressons-nous maintenant au contenu du spam. En observant le source du message, la première chose que l'on voit est la version 'texte' de ce dernier (Fig. 6). En fait, elle a l'air normale, si on omet le sens du texte dans le contexte d'un envoi qui serait réalisé par cette chaîne de télévision :

```
Content-type: text/plain

>+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
>THE DAILY TOP 10
>from CNN.com
>Top videos and stories as of: Aug  1, 2008  3:58 PM EDT
>+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

TOP 10 VIDEOS

1. MOM PARTIES AT A CLUB
http://www.cnn.com/video/partners/email/index.html?url=/
video/crime/2008/08/05/ng.mom.pics.cnn
Nancy Grace has new photos of Casey Anthony partying at a
nightclub after her daughter Caylee vanished.

2. WOMAN'S COMPUTER SPIES ON HER
http://www.cnn.com/video/partners/email/index.html?url=/
video/tech/2008/08/05/pkg.webcam.spying.wcjb

3. TIGERS ATTACK TEEN AT ZOO
http://www.cnn.com/video/partners/email/index.html?url=/
video/us/2008/08/05/wuellner.tiger.attacks.teen.kytv
```

Fig. 6 : Partie « texte » du pourriel

« Les liens contenus dans ce code renvoient bien vers des domaines appartenant à CNN.

La partie suivante, censée être la version HTML du même message, contient, elle, des liens vers une toute autre destination » (Fig. 7) : image ci-contre.

« Tous ces liens renvoient vers une seule et même page : » (Fig. 8) :

Fig. 8 : Une page invitant à mettre à jour le lecteur Flash

```
Content-type: text/html

<html>
<head>
<title>CNN.com Daily Top 10</title>
<style type="text/css">a{text-decoration:none;}</style>
</head>
<body style="margin:0px;">

<!--Content-->

<table border="0" cellpadding="0" cellspacing="0">
<tr>
<td style="padding:10px 0px 10px 7px;" valign="top"
width="65">
<img src="http://i.a.cnn.net/cnn/linkto/email/daily.top10/
images/tz.1.gif"><br>
<img height="16" src="http://i.cnn.net/cnn/images/1.gif"></
td>
<td valign="top" style="padding:10px 0px 10px 10px;font-
family:arial;font-size:12px;">
<a href="http://www.martinkahl.com/cnnvideo.html">
<font color="#980000">
<strong>Murderer on the loose after cop bungle in Iowa</
strong>
</font></a>
<br></td>
</tr>
<tr>
<td colspan="2"><img src="http://i.a.cnn.net/cnn/linkto/
email/daily.top10/images/mid.dotted.line.gif"></td>
</tr>
</table><table cellspacing="0" cellpadding="0"
border="0"><tr>
<td style="padding:3px 0px 2px 12px;font-family:arial;font-
size:12px;">
<strong>2. </strong><a href="http://www.martinkahl.com/
cnnvideo.html">
<font color="#980000">Miley Cyrus describes her dream
man</font></a>
<img width="1" height="12" src="http://i.cnn.net/cnn/
images/1.gif"></td>
</tr>
<tr>
<td><img src="http://i.a.cnn.net/cnn/linkto/email/daily.
top10/images/mid.dotted.line.gif"></td>
</tr>
<tr>
<td style="padding:3px 0px 2px 12px;font-family:arial;font-
size:12px;">
<strong>3. </strong><a href="http://www.martinkahl.com/
cnnvideo.html">
<font color="#980000">Three children jailed for armed
holdup</font></a>
<img width="1" height="12" src="http://i.cnn.net/cnn/
images/1.gif"></td>
</tr>
```

Fig. 7 : Partie HTML du pourriel

« C'est à ce moment que le piège se referme : la prétendue version du lecteur Flash proposée au téléchargement est un cheval de Troie caché dans le fichier `flash_update.exe`. Comme vous l'aurez compris, nous avons affaire à un ordinateur infecté qui envoie tantôt des spams publicitaires, tantôt des courriels piégés. Autre point : cet ordinateur a participé à plusieurs campagnes promotionnelles – car on peut en effet parler de véritables opérations

qui n'ont rien à envier à celles de notre département Marketing. Parmi les spams envoyés depuis cette source, on trouve ainsi des liens vers des pharmacies en ligne, comme celle que nous venons de voir, mais aussi vers des sites de ventes de (fausses) montres de luxe, des casinos et sites de paris en ligne, sans oublier des sites qui vantent les mérites de produits naturels aux vertus... allongeantes ! »

« Je ne vois pas bien le lien entre tout ceci et notre affaire en particulier. Si je vous comprends bien, la machine qui a émis le courrier peut être mise hors de cause, c'est bien cela ? »

« Oui, son propriétaire n'est peut-être même pas au courant que son ordinateur est infecté. »

« Mais le site sur lequel Monsieur M. a passé commande, il est bien identifié lui. Il a un nom de domaine, il doit bien y avoir un moyen de remonter jusqu'à son propriétaire ! »

« En théorie, oui. Mais, dans notre cas, vous imaginez bien que ce n'est pas si simple. Pour commencer, il n'y a pas un domaine, mais des dizaines qui sont utilisés pour héberger tous les sites de vente en ligne. Par contre, pour chacune des catégories que je vous citais – pharmacie, montres, casinos – il semble n'y avoir qu'un nombre limité de modèles de sites, déclinés suivant des zones géographiques ciblées : on trouve ainsi des versions en anglais, en français, en allemand, utilisant les monnaies locales. Mais revenons sur les domaines. En théorie, pour qu'un domaine existe sur Internet, il doit dans un premier temps être déposé auprès d'un *registrar* qui lui-même a obtenu une délégation pour un *Top Level Domain* (.com, .fr, .net, etc.). Ces TLD sont gérés par l'ICANN. Pour certains dont le célébrissime .com, on trouve une multitude de sociétés autorisées à vendre l'enregistrement de noms de domaines. Ces sociétés fournissent un service de recherche communément appelé whois qui permet à tout internaute de récupérer des informations techniques et administratives relatives à un domaine. »

« C'est-à-dire ? »

« Eh bien, l'identité du dépositaire du nom, son adresse postale, la date de dépôt du domaine, celle de son expiration, etc. »

« Cette base peut donc nous donner le nom du propriétaire des domaines frauduleux ? »

« Oui... mais non : il n'y a malheureusement aucune norme, aucun standard qui oblige les registrars à vérifier la validité des données fournies par les déposants. Les seules données réellement validées sont celles relatives au paiement du service ! Autant dire que si le registrar accepte les paiements par des moyens peu traçables, il n'y a quasiment aucune chance de retrouver grâce au whois qui se cache derrière un domaine ! Prenons un exemple : dans la majorité des cas qui nous intéressent, les domaines utilisés ont été déposés auprès d'un seul et même registrar : SouthNames. En faisant une recherche sur cette société, j'ai appris que 90 % de tous les domaines qu'elle gère sont utilisés à des fins frauduleuses ! Les 10 % restants servent de couverture. De plus, cette société ne répond ni aux courriels, ni au téléphone.

Toutes les opérations d'achat et de gestion de noms de domaines se font en ligne. Je ne suis même pas certain que cette société ait une existence ailleurs que sur Internet. Une chose cependant est sûre : elle est plus que laxiste quant aux informations fournies par ses clients. On trouve ainsi des renseignements très fantaisistes dans leurs bases whois pour les domaines étudiés, certains sont attribués à des conseillers du président russe par exemple, d'autres sont domiciliés en Afghanistan. On n'apprend donc pas grand chose sur leurs réels propriétaires sinon qu'ils ont un certain sens de l'humour, à l'instar de celui-ci, domicilié dans une rue moscovite qui n'existe pas (encore) (Fig. 9) :

```
Registrant:
    Vano Bilbao
    92 Putina str. 342
    Moskow,MSK,RU 410992
```

Fig. 9 : Exemple de domiciliation fantaisiste d'un propriétaire de domaine

Certes, ce n'est pas le seul registrar utilisé par les fraudeurs, mais tous présentent les mêmes « avantages » de discrétion, certains parce qu'ils sont en Chine, la langue constituant donc une barrière 'naturelle'. D'autres n'hésitent pas à en faire un argument commercial et promettent à leurs clients anonymat et protection de vie privée (Fig. 10 et 11) :

Fig. 10 : Mot d'ordre de Privacy Protect, la protection de l'anonymat de ses clients

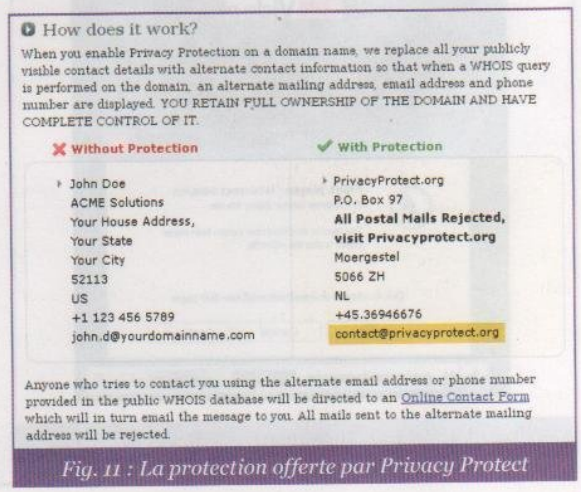


Fig. 11 : La protection offerte par Privacy Protect

La plupart de ces sociétés se contentent d'avoir quelques clients 'normaux' qui leur servent de couverture.

Aussi, il faut garder à l'esprit que les noms de domaines passent régulièrement « de main en main » via des forums non liés au *registrar* (Fig. 12), rendant plus difficile l'identification des personnes les contrôlant réellement :

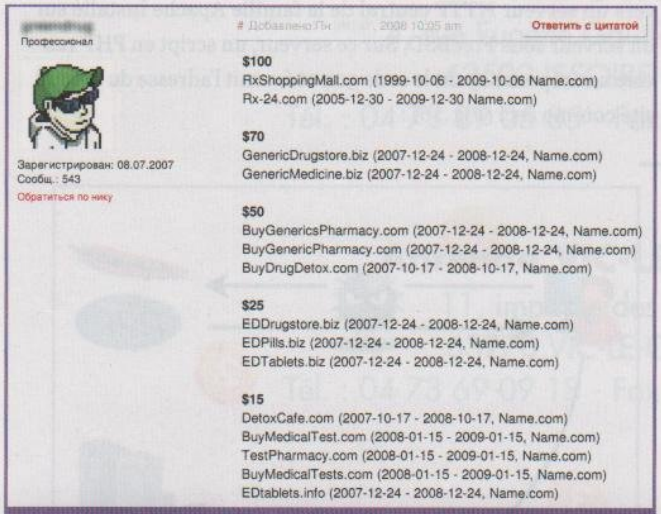


Fig. 12 : Revente de domaines de « pharmacies » sur un forum spécialisé

De la même façon que pour les *registrars*, certains hébergeurs sont moins regardants que la moyenne sur les activités de leurs clients. On trouve quelques interventions 'intéressantes' sur leurs forums (Fig. 13) : »

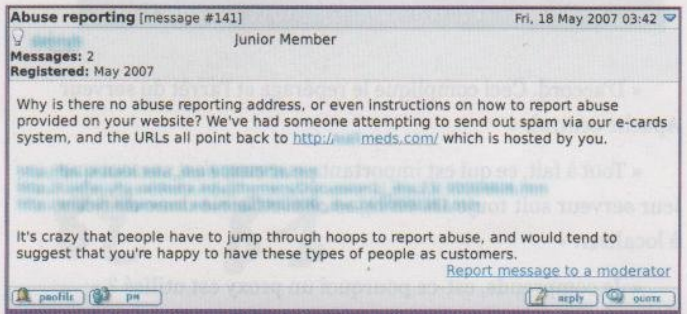


Fig. 13 : Sur le forum d'un hébergeur réputé dans certains milieux, une personne se plaint de ne pas pouvoir signaler un site de vente de médicaments dont l'adresse est envoyée en masse.

« Ces sites existent bien, eux. Il doit bien y avoir des machines qui les hébergent et donc un moyen de les localiser ! »

« Une fois encore : oui mais non. À chaque domaine sont associées plusieurs dizaines d'adresses IP différentes. La plupart du temps, il n'y a aucun lien logique entre ces machines : elles se trouvent au quatre coins du monde, dans des plages d'adresses non contiguës, et appartiennent à des FAI de toutes tailles et de toutes nationalités. Très souvent, les machines rattachées à ces IP ne sont même pas des serveurs, mais les ordinateurs de simples particuliers ou de très petites entreprises !

Prenons le cas d'un des très nombreux domaines utilisés. Voici les informations obtenues en interrogeant le DNS (Fig. 14) :

```
$ dig grv.eafor.cn
; <<>> DiG 9.4.2-P2 <<>> grv.eafor.cn
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34212
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION
;grv.eafor.cn.           IN    A

;; ANSWER SECTION:
grv.eafor.cn.       180    IN    CNAME    eafor.cn.
eafor.cn.           180    IN    A        92.100.29.179
eafor.cn.           180    IN    A        98.195.62.62
eafor.cn.           180    IN    A        190.191.12.93
eafor.cn.           180    IN    A        60.243.6.61
eafor.cn.           180    IN    A        77.89.73.82
eafor.cn.           180    IN    A        78.96.254.35
eafor.cn.           180    IN    A        81.198.194.219
eafor.cn.           180    IN    A        84.42.141.20
```

Fig. 14 : Réponse du DNS pour « grv.eafor.cn »

Première remarque : on a affaire à un enregistrement de type `catch-all`. Quelle que soit la valeur de la première partie du FQDN (*Fully Qualified Domain Name*), c'est-à-dire celle qui précède `eafor.cn`, le DNS renvoie une réponse de type `CNAME` qui elle-même référence les adresses IP qui suivent. Il est ainsi possible pour les spammeurs de 'communiquer' sur un nombre de noms virtuellement illimité. »

« Quel est l'intérêt ? Les internautes ne peuvent pas retenir autant d'adresses ! »

« Ils n'ont pas besoin de le faire : n'oubliez pas qu'ils ont juste à cliquer sur un lien... puis à acheter ! L'intérêt pour les spammeurs est ailleurs : ils peuvent utiliser le nom de leur choix, `www.blueblue.com.eabelow.cn` par exemple, sans avoir à configurer quoi que ce soit, utiliser les préfixes (ex. : 'grv' supra) pour identifier une campagne de spam, jouer sur le fait que certains administrateurs n'interdiront pas l'accès à tout un domaine par peur de bloquer l'accès à un hypothétique site légitime fréquenté par les 'VIP' de l'entreprise, les foudres du patron étant un concept plus concret à leurs yeux que l'espionnage économique (ou autre). Tous les sous-domaines d'`eafor.cn` pointeront ainsi vers le même '*pool*' d'adresses IP. Revenons aux enregistrements de type `A`. Leur TTL (*Time To Live*) est court, 180 secondes. Dans certains cas, il est même égal à 0 ce qui neutralise la mise en cache des réponses par les *resolvers* DNS. Ensuite, la magie du *Round-Robin DNS* opère : les adresses IP d'un même pool sont renvoyées dans un ordre différent.

À chaque nouvelle requête, ce sont donc de nouvelles adresses qui sont retournées. Ce qui signifie qu'il n'y a pas un site qui héberge notre pharmacie, mais des dizaines. J'ai même trouvé des domaines qui utilisaient des centaines d'adresses IP différentes. Cette technique a un nom : *Fast-Flux*. L'ironie est qu'elle s'inspire de méthodes utilisées pour assurer à des sites tout à fait officiels une très haute disponibilité tout en répartissant la charge géographiquement, tout comme le fait Akamai par exemple. »

« Comment est-il possible d'installer sur toutes ces machines des sites Web identiques ? Comment les pirates qui sont derrière tout cela arrivent-ils à administrer un tel parc ? »

« C'est relativement simple. Regardez la séquence suivante, ce sont les commandes HTTP échangées entre un navigateur et le serveur vers lequel pointe le lien d'un spam (Fig. 15) :

Client	Serveur
GET / HTTP/1.1 Host: kzto.eabelow.cn	
	HTTP/1.1 302 Found Server: Apache/2.0.59 (FreeBSD) X-Powered-By: PHP/4.4.4 Location: http://pharmacheap.org Content-Length: 0 Connection: close Content-Type: text/html
GET / HTTP/1.1 Host: pharmacheap.org	
	HTTP/1.0 200 OK Server: nginx/0.6.32 Content-Type: text/html X-Powered-By: PHP/4.4.9 Set-Cookie: USID=7b228cdbbdb08f517f87f9d6cfae1244; expires=Sun, 19 Feb 2012 02:10:59 GMT; path=/ X-Cache: MISS from loadbalancer X-Cache-Lookup: MISS from loadbalancer:80 Via: 1.0 loadbalancer:80 (squid)

Fig. 15 : Échanges HTTP quand on suit le lien d'un spam

À la première requête, le client se voit répondre, par un serveur Apache sous FreeBSD, que le site demandé a changé d'emplacement et le serveur lui donne le nom du 'nouveau' site. Le client envoie ensuite une requête vers le second site et reçoit une réponse, la page d'accueil du site, d'un serveur nginx. Cette réponse est passée par un serveur mandataire Squid, ce qui nous est indiqué dans l'en-tête `Via` et par la présence des en-têtes `X-Cache` et `X-Cache-Lookup`.

Nous avons vu qu'un grand nombre d'adresses IP sont associées au premier nom de domaine et si nous interrogeons régulièrement le serveur DNS, nous verrons qu'elles changent plus ou moins régulièrement. Si, par exemple, nous demandons l'enregistrement `PTR` associé à chaque adresse IP, nous pouvons déduire des noms obtenus que ces adresses sont, pour la plupart, celles d'ordinateurs appartenant à des particuliers (ce qui peut se confirmer par d'autres méthodes). Or, la plupart des particuliers n'ont probablement pas un serveur Apache installé sur leur machine et, si c'était le cas, il ne serait pas toujours de la même version et encore moins souvent sous FreeBSD. Mon hypothèse est que ces machines auxquelles nous nous connectons en premier lieu sont des machines infectées qui servent de tunnel TCP vers un serveur HTTP central de la famille Apache installé sur un serveur sous FreeBSD. Sur ce serveur, un script en PHP renvoie une réponse ayant le code 302 et fournit l'adresse du second site comme ceci (Fig. 16) :

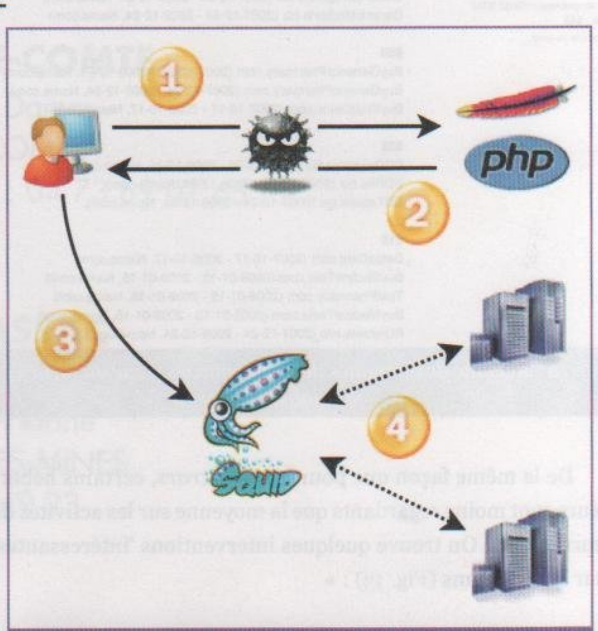


Fig. 16 : Tunnel TCP, redirection, proxy et serveurs HTTP

« D'accord. Ceci complique le repérage et l'arrêt du serveur Apache central. »

« Tout à fait, ce qui est important pour eux, c'est surtout que leur serveur soit toujours en ligne et accessible donc difficile à localiser. »

« Je comprends, est-ce pourquoi un proxy est utilisé ? »

« Je vois plusieurs raisons possibles d'utiliser un serveur mandataire dans ce cas. Il peut tout simplement s'agir d'un mécanisme de répartition de charge ou de tolérance de panne. Les machines qui hébergent réellement les sites – les *motherships* ou maisons mères – peuvent ainsi se voir attribuer les requêtes d'un client parce que les autres sont trop « chargées » ou en cours de maintenance, etc. Cela peut avoir une autre utilité : héberger un site sur un serveur sans que cela ne soit visible de l'extérieur quand on se connecte directement à celui-ci. Par exemple, imaginons un serveur hébergeant une page visible de tous, d'apparence anodine et ne comportant

aucun lien, à la racine du site et stockée à l'emplacement /var/www/index.html sur le disque. Ajoutons une seconde page moins anodine à l'emplacement /var/www/hiddenstuff/pharmacy/index.html. Si le serveur reçoit une requête GET /index.html, il renvoie une page qui semble légitime. Si le proxy reçoit une requête GET /index.html, il peut la réécrire avant de la transmettre au serveur et lui envoyer GET /hiddenstuff/pharmacy/index.html. Le serveur renverra, dans ce cas, la page d'accueil d'un site de vente de contrefaçons de médicaments, page qui semblera être à la racine d'un site pour le visiteur. Ceci peut permettre l'hébergement clandestin d'un site sur un serveur dont les journaux sont peu surveillés sans que le site d'origine soit modifié ou supprimé. Plusieurs serveurs compromis à l'avance permettent alors d'avoir une réserve de sites prêts à l'emploi qui peuvent être activés par un simple changement de configuration du serveur mandataire. Évidemment, au-delà de cela, le fait de mettre un proxy entre les visiteurs et les motherships rend l'arrêt de l'ensemble plus complexe, surtout si le mandataire est situé dans un pays qui n'est pas en 'relation suivie' avec celui où est la maison mère, etc.

Une dernière chose sur cette chaîne HTTP : lors de mes tests, j'ai pu constater qu'il y a un contrôle relativement basique du type de client utilisé pour se connecter à un site. »

« Qu'entendez-vous par là ? »

« J'ai utilisé dans un premier temps le logiciel wget pour me connecter, afin d'éviter toute infection ou attaque dirigée contre mon navigateur. Voici les réponses que j'ai obtenues (Fig. 17) :

```
$ wget -S http://ktzo.eabelow.cn
Résolution de ktzo.eabelow.cn... 212.16.134.173, 77.41.69.13,
81.198.194.219, ...
Connexion vers ktzo.eabelow.cn|212.16.134.173|:80...
connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.1 302 Found
  Server: Apache/2.0.59 (FreeBSD) PHP/4.4.4 with Suhosin-
  Patch
  X-Powered-By: PHP/4.4.4
  Location: http://pharmacheap.org
  Content-Length: 0
  Connection: close
  Content-Type: text/html
Emplacement: http://pharmacheap.org [suivant]
Résolution de pharmacheap.org... 121.127.77.27
Connexion vers pharmacheap.org|121.127.77.27|:80...
connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.0 502 Bad Gateway
  Server: squid
  Content-Type: text/html
  X-Squid-Error: ERR_ZERO_SIZE_OBJECT 0
  X-Cache: MISS from loadbalancer
  X-Cache-Lookup: MISS from loadbalancer:80
  Via: 1.0 loadbalancer:80 (squid)
```

Fig. 17 : Réponses obtenues avec un client non accepté

En clair, si le second serveur ne détecte pas le champ User-Agent dans les en-têtes HTTP de la requête qui lui est passée, il renvoie une réponse de taille nulle. Cela évite de répondre aux curieux qui ne sont pas trop insistants...

De même, si l'on invoque un préfixe quelconque comme misc.eabelow.cn, on obtient une erreur de type 404. »

« J'ai quelques difficultés à concevoir qu'un même groupe de personnes puisse contrefaire des médicaments, organiser un réseau de vente, envoyer du spam, maintenir des sites avec une architecture à forte redondance, gérer les paiements, expédier les produits, etc. alors que, nous, nous arrivons juste à avoir du mal à les attraper. »

« Il ne s'agit pas d'un seul groupe de personnes, M. Phelps. Voici un schéma avec les principaux intervenants, grâce auquel je vais vous expliquer la nature des relations entre les uns et les autres (Fig. 18) :

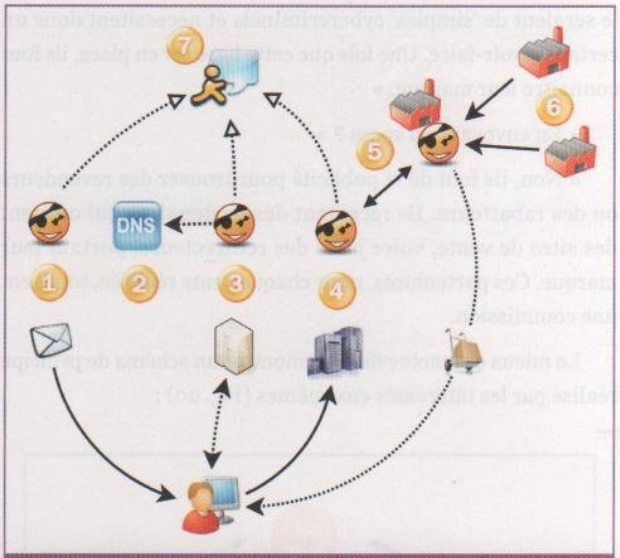


Fig. 18 : Principaux intervenants

Les spammeurs (n°1) font des campagnes de publicité et incitent les destinataires à visiter des sites plus ou moins insaisissables maintenus par des redirecteurs (n°3) qui perçoivent des commissions sur les ventes versées par les organisateurs de réseaux (n°4). Les redirecteurs font appel aux services de registrars plus ou moins scrupuleux (n°2) pour déposer les noms de domaines utilisés lors des campagnes de spam.

Les contrefacteurs (n°5) se fournissent auprès de fabricants de produits de base (n°6) comme on peut en trouver sur ce site de B2B (*Business-to-business*) (Fig. 19) et qui sont probablement, pour certains, tout ce qu'il y a de plus « légitime » :

« Nos contrefacteurs s'associent ensuite à des 'organisateurs de réseau' qui leur achètent les produits finis et organisent la vente, c'est-à-dire qu'ils créent une entreprise, lui font ouvrir des

Fig. 19 : Exemple de produits proposés sur un site de B2B

comptes bancaires, trouvent un expéditeur, vérifient la mise en place de la logistique nécessaire pour que les commandes des clients soient honorées, etc. Ces actions sont, si l'on peut dire, 'impliquantes' pour l'organisateur de réseau, elles ont des conséquences 'palpables' (ex. : pas d'expéditeur, pas de livraison). Elles demandent d'être plus 'en prise' avec 'le terrain' que ne le seraient de 'simples' cybercriminels et nécessitent donc un certain savoir-faire. Une fois que cette base est en place, ils font connaître leur marque. »

« En envoyant du spam ? »

« Non, ils font de la publicité pour trouver des revendeurs ou des rabatteurs. Ils recrutent des partenaires qui ouvrent des sites de vente, voire juste des redirecteurs, portant leur marque. Ces partenaires, pour chaque vente réalisée, touchent une commission.

Le mieux est encore de vous montrer un schéma de principe réalisé par les intéressés eux-mêmes (Fig. 20) :

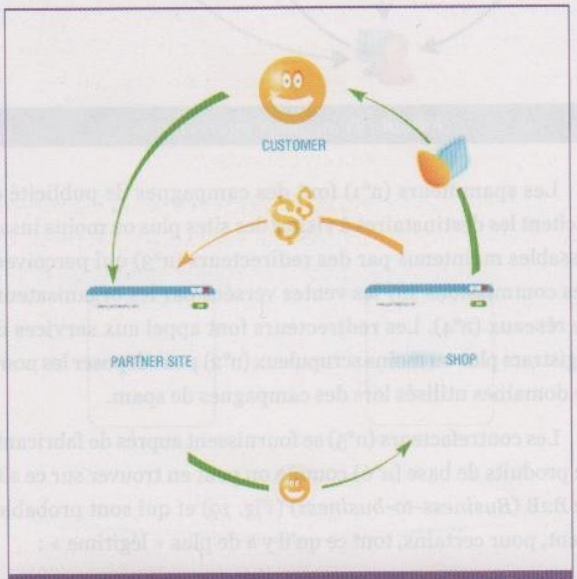


Fig. 20 : Versement de commissions sur la vente de médicaments

Comme vous le voyez, le client visite le site du partenaire, qui le renvoie vers le site du véritable vendeur, il fait des achats, reçoit – parfois – des produits proches de ceux qu'il a commandés, et le propriétaire du site de vente reverse une commission au partenaire.

Plusieurs 'organisateurs de réseau' existent et rivalisent donc d'atouts (Fig. 21) pour recruter les meilleurs partenaires :

Fig. 21 : Les avantages mis en avant pour attirer les partenaires

Des taux importants, des paiements réguliers, la possibilité de simplement diriger des clients vers un site de vente principal ou d'ouvrir son propre site, une interface de gestion simple à utiliser, des statistiques (Fig. 22), etc. Il doit être assez coûteux, ne serait-ce qu'en temps, de trouver des partenaires fiables et de confiance, donc ils s'efforcent de les fidéliser. ».

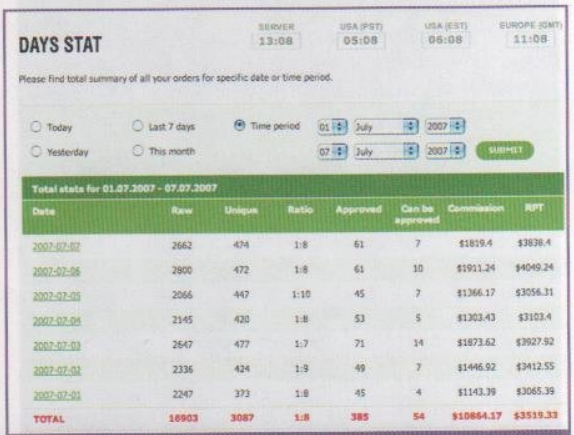


Fig. 22 : Statistiques de commissions sur ventes de médicaments, vues depuis une interface de gestion

« Vous voulez dire que ces pirates ont adopté un modèle proche de celui des franchises commerciales ? En gros, les redirecteurs sont des franchisés et les motherships des franchiseurs, ces derniers libérant les premiers des contraintes liées au développement d'une marque ou d'une enseigne. Comme le fait McDonald's avec son réseau de restaurants ou Renault avec ses concessionnaires ! Et la publicité revenant aux franchisés, ceux-ci font appel aux services d'annonceurs, à savoir les spammeurs ! »

« En quelque sorte, oui. Sur certains points le modèle a même été amélioré, ne serait-ce que parce que nos pirates sont affranchis des contraintes légales et juridiques. »

« Cela me semble beaucoup plus professionnel et organisé que ce que j'imaginais. Et, puisqu'il semble y avoir de nombreux intervenants de la fabrication à l'expédition au client : peut-on faire le lien entre eux ? »

« Pour certains d'entre eux, oui, puisque nous pouvons faire le lien entre différents sites franchisés et un franchiseur. Par exemple, voyez la capture suivante (Fig. 23) :

Fig. 23 : Sites de vente montrés en exemple par le propriétaire de la « marque »

On y voit différents sites servant d'exemple au propriétaire de la 'marque'. En cherchant un peu, on arrive à retrouver des sites plus ou moins jumeaux de ces derniers. »

« Que voulez-vous dire par 'plus ou moins jumeaux' ? »

« Regardez. Pour une liste de noms de domaine, je vais récupérer la page d'accueil, sommairement extraire les chemins vers les scripts PHP qui sont dans les attributs des balises HTML et compter le nombre de domaines qui font référence à chacun de ces scripts (pour les dix plus courants) (Fig. 24) :

```
$ for d in \
> canadian-pharmacy-4u.com \
> rxmeds4all.com \
> www.easyprescription.us \
> www.healthmens.org \
> www.midpharmacy.com \
> www.simple-medstore.com ; do
> wget -O - -q "http://${d}/" \
> | sed -E -n 's:^.*[a-z]="([^"]*/)?'\
> '([^"/]+\.php)(\?[^"]*)?".*$:\2: p' \
> | sort | uniq ; done \
> | sort | uniq -c \
> | sort -n | tail -n 10
   5 counter.php
   5 item.php
   6 all_products.php
   6 cart.php
   6 contacts.php
   6 faq.php
   6 moneyback_policy.php
   6 privacy_policy.php
   6 search.php
   6 shipping_policy.php
```

Fig. 24 : Nombre de sites, parmi six, qui font référence à un même script

Que remarquez-vous ? »

« Je dirais...que, sauf exceptions, on retrouve les mêmes scripts sur les six sites interrogés. »

« Tout à fait : mis à part le thème graphique et quelques détails, la structure de ces sites est identique. Bref, les franchiseurs fournissent des 'kits clés en main' aux franchisés et c'est un lien qui pourrait être suivi. »

« Et concernant le lien avec le spam ? »

« Nous allons y venir. J'ai récupéré quelques exemplaires du programme malveillant utilisé pour les envois de pourriels, leur analyse est en cours, je vous fais un topo la semaine prochaine sur ce que j'aurai découvert. »

À suivre... au prochain numéro (faute de place). ■

Références

[1] VAL (Philippe), « Internet, la Kommandantur libérale ».

[2] LEFEBVRE (Frédéric), « Internet, [...] les psychopathes, les violeurs, les racistes et les voleurs y ont fait leur nid. », http://www.assemblee-nationale.fr/13/cri/2008-2009/20090103.asp#P705_153094.

mtix.spc@gmail.com

LA COMPROMISSION ÉLECTROMAGNÉTIQUE

mots-clés : *TEMPEST / réglementation / OTAN / SDIP / canaux auxiliaires*

« La physique nous trahit et James Maxwell est son complice. »

C'est de cette manière que pourrait se résumer ce qui se cache derrière le concept ésotérique de « compromission électromagnétique ». Et encore, il n'a pas encore été question de « La Menace Tempest » (pas si fantôme que ça d'ailleurs)...

En termes simples, la compromission électromagnétique pourrait se définir par « la fuite fortuite d'informations lorsqu'elles sont manipulées par des machines électriques ou électroniques ».

Les paragraphes qui suivent détaillent quelques aspects de cette menace, à commencer par l'explication de l'origine physique de cette compromission, son exploitation et les moyens de s'en protéger.

Mais, il ne faut pas limiter ce concept à ce qu'historiquement on appelle « le Tempest ». En effet, cette physique qui nous trahit (ou du moins trahit nos secrets) est exploitée aux mêmes fins, mais de manière adaptée, par les hackers hardware qui s'en prennent aux cartes à puce ou aux équipements de chiffrement. Une histoire en rouge et noir...

1. Les signaux parasites compromettants

1.1 Origines des SPC

Le passage d'un courant dans un circuit électrique est la source d'un rayonnement électromagnétique, autrement dit, une onde se propageant en espace libre, par rayonnement. Ce rayonnement est appelé parasite, car s'il est capté par un autre circuit (une radio ou une centrale ABS de voiture par exemple), il peut être à l'origine d'un bruit acoustique ou d'une défaillance plus grave.

Le problème se corse lorsqu'un parasite présente une corrélation avec son signal d'origine. Il devient alors fort intéressant pour un agresseur de parcourir le chemin inverse du parasite à savoir remonter aux informations dont il est issu. On parle alors de signaux parasites compromettants, les célèbres « SPC ».

1.2 Anatomie des SPC

Les SPC sont polymorphes. Ils peuvent se révéler sous la forme de signaux très large bande et leur captation peut être (dé)favorisée sur certaines plages de fréquences par un nombre incalculable de paramètres : la géométrie des lieux, la présence d'un conducteur fortuit, la météo, les tolérances des caractéristiques des composants électroniques qui le rendront plus facilement détectable, etc.

Les SPC peuvent présenter des corrélations sur fronts, sur niveaux ou une combinaison complexe de ces deux catégories. Dans le cas d'une corrélation sur front, les SPC exhibent des pics correspondant aux fronts montants ou descendants d'un signal logique. Ces fronts, qui ont des raideurs différentes (les fronts montants sont en général

plus lents que les descendants), n'ont pas les mêmes contenus spectraux, ce qui explique qu'ils ne soient en général pas disponibles sur les mêmes plages de fréquence. Cependant, il arrive de trouver des plages de fréquences « favorables » dans lesquelles les deux types de fronts sont présents ce qui simplifiera grandement l'étape de reconstitution du signal.

Il est rare de rencontrer des SPC à bande très étroite, à moins qu'ils ne soient portés par une modulation interne à l'équipement ou issus d'une fuite intentionnelle...

1.3 Propagation des SPC

1.3.1 Hertzien

Le vecteur intuitif de propagation des SPC est la propagation hertzienne. On aura compris que les rayonnements électromagnétiques se propagent de manière incontrôlée depuis leur source (le circuit électrique ou électronique traitant l'information) vers l'extérieur de l'équipement, en jouant au passe-muraille. Savoir capter, identifier et exploiter des SPC requiert une forte somme de compétences, de savoir-faire et du matériel ad hoc. La difficulté consiste à capter de « bons signaux », à savoir ceux qui seront porteurs d'informations peu polluées par les parasites « inoffensifs » et les émissions « légales » (radio, communication sans fil,...).

1.3.2 Conductions

Les SPC sont des phénomènes sournois. Ils ne se contentent pas de se propager par voie hertzienne, transformant tout équipement électrique en émetteur radio fortuit. Ils se propagent aussi par conduction. Lorsqu'un SPC se propage, il peut induire sur les conducteurs électriques se trouvant à proximité un signal électrique portant lui-même de l'information. Deux problèmes apparaissent alors : il y a beaucoup de conducteurs fortuits autour de nous (câble d'alimentation électrique, câble réseau, câble téléphonique, tuyau de chauffage central, etc.) et, à énergie d'émission équivalente, les propagations en conduction peuvent avoir une portée bien supérieure aux propagations hertziennes. De nombreux sceptiques croient qu'étant reliés à la terre, donc à un potentiel nul, les conducteurs fortuits ne peuvent être les vecteurs de propagation de signaux parasites. Ils oublient en général que ces approximations de potentiel nul ne sont valables que pour les signaux « lents », comme les courants continus ou alternatifs 50 Hz, mais que, passé la barre de la centaine de MHz, un conducteur peut présenter une impédance selfique non négligeable (comme si une bobine apparaissait en série de la résistance), ce qui aura pour effet la transformation en une sorte de « self de choc » de notre innocent tuyau de radiateur relié à la terre. Tout devient plus simple pour les parasites haute fréquence, car la bobine qui est apparue entre le sol et le circuit de chauffage les a virtuellement déconnectés de la terre.

1.3.3 Hitchhikking

Comme expliqué dans **[NSA1 p93]**, une des plus importantes menaces est liée au couplage d'équipements radio avec les matériels de chiffrement. Dans certains cas, le signal émis par le poste est d'une telle puissance qu'il se couple avec les signaux internes de l'équipement et permet le transport modulé du signal rouge (celui qui contient l'information secrète contrairement au signal noir) à une distance très importante (éventuellement plusieurs kilomètres).

1.4 Interception des SPC

Chercher une aiguille dans une botte de foin est plus facile que de chercher des SPC. En effet, une aiguille a toujours à peu près la même forme, ce qui n'est vraiment pas le cas des SPC.

Fort heureusement, les signaux électriques doivent respecter des standards et des gabarits. Ces contraintes créent des « marquants » (fréquence d'horloge, bit unitaire, temps de montée,...) qui rendent leurs parasites plus facilement identifiables pour l'opérateur expérimenté.

1.4.1 Signal série et parallèle

104µs, c'est la période à laquelle apparaîtront les parasites d'une ligne série à 9600 bauds. Ainsi, un signal parasite dont l'autocorrélation exhiberait des pics à 104µs (et ses harmoniques) serait très certainement porteur d'informations provenant d'une ligne série.

De même, si la bande passante du signal est plus réduite, il serait possible de détecter l'enveloppe du signal, grâce aux successions des trames séparées par les bits de stop et de parité, définis dans le protocole de communication.

Toute information est utile, et c'est pour cette raison que la caractérisation préalable des signaux électriques réels (mesures des temps de montée et descente, connaissance du protocole de communication, mesure du bit unitaire,...) est un facteur de réussite important d'une bonne interception.

Les signaux parallèles ont toujours posé problème : la présence simultanée d'un nombre important de transitions synchrones rend très difficile l'extraction et l'exploitation de ce signal particulier.

1.4.2 Signal vidéo analogique

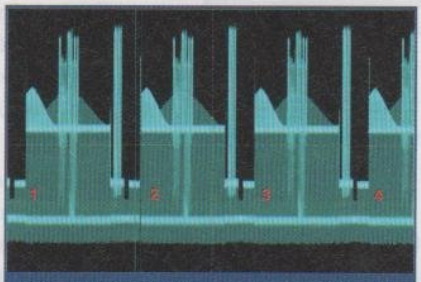


Fig. 1 : Signal vidéo analogique (Bande passante 500MHz)

Le signal vidéo VGA est une succession de motifs répétitifs, les trames, qui se répètent à la cadence de rafraîchissement de l'écran (60Hz, 75Hz, ...). Dans la capture ci-contre, les chiffres 1, 2, 3 et 4 représentent respectivement le début de trame d'images affichées à l'écran (Figure 1).

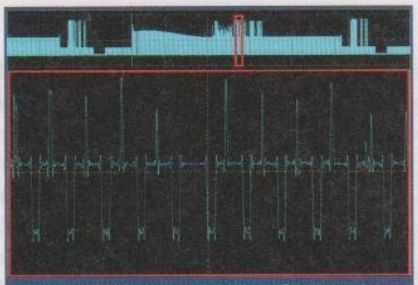


Fig. 2 : Zoom sur un signal trame (en haut) contenant des signaux ligne (en bas)

Les trames contiennent des motifs répétitifs qui correspondent au signal de balayage du spot à savoir le signal ligne. Dans la figure 2, le signal du bas (signal ligne) correspond au zoom effectué dans le rectangle rouge de la figure du haut (signal trame).

Un signal vidéo (il en faut plusieurs pour une image en couleur) représente les niveaux d'intensité d'une couleur. Dans notre cas, en basculant artificiellement ce signal à la verticale, on reconnaît très facilement un écran de login (Figure 3).

Lorsqu'il est observé à faible bande passante, c'est l'enveloppe du signal vidéo analogique qui est détecté (Figure 4).

Or, ces formes répétitives basse fréquence sont très facilement détectables à l'oreille, même noyées dans du bruit. Elles créent une « ronflette » très caractéristique qu'un opérateur aguerri a tôt fait de détecter.

Une fois ce signal identifié, il est isolé avec une bande passante adaptée (quelques MHz ou dizaine de MHz) et réinjecté sur un moniteur analogique dont on aura synthétiquement reconstitué les signaux de balayage horizontaux et verticaux.

Il n'est malheureusement pas possible de différencier les composantes RVB (ou YUV) d'un signal vidéo. Le signal obtenu représente une variation d'amplitude, car il est le plus souvent un mélange des couleurs (respectivement chrominance/luminance). L'image reconstituée par ce procédé est donc une image en noir et blanc.

Seul Tom Cruise dans *Mission Impossible* réussit à obtenir de la couleur (le titre du film aurait cependant dû nous mettre sur la piste).

La précision des générateurs utilisés lors de la synthèse des signaux de synchronisation permettra de discriminer les images de deux écrans identiques, situés dans la même pièce. Les réglages sont effectués au millionième de Hertz, ce qui permet d'être plus exigeant que la tolérance des composants électroniques utilisés dans les cartes graphiques.

1.4.3 Signal vidéo numérique

Les écrans LCD apportent-ils une solution à la fuite des signaux CRT ? Pas sûr.

Les signaux des dalles LCD sont parmi les plus rapides en temps de montée/descente que l'on puisse trouver dans un ordinateur. De fait, ils possèdent une grande richesse spectrale, ce qui tendrait à jouer en leur défaveur.

Les cartes DVI sont rarement DVI-D (*Digital*), à savoir seulement numériques. La plupart des cartes vidéo sont des DVI-I (*integrated*) dans lesquelles le signal analogique est présent... ce qui permet souvent d'effectuer des interceptions « à l'ancienne ».

1.4.5 Modulo

Les interceptions présentées ci-avant sont des illustrations classiques des problèmes liés aux SPC. Il convient néanmoins de moduler le risque qu'elles représentent. L'abondance des fuites Tempest dont il est fait mention dans les documents déclassifiés est directement liée aux technologies employées à l'époque de ces exploits. En effet, les télétypes, une des cibles favorites des chasseurs de parasites, utilisaient des relais électromécaniques à l'origine de parasites puissants et riches en spectre.

Aujourd'hui, les standards d'économies d'énergie (TCO 99 par exemple) défavorisent les attaquants car, en limitant la consommation électrique des écrans cathodiques, ces normes obligent les agresseurs à rapprocher leurs antennes . Malgré l'amélioration permanente des performances des équipements de mesure (en termes de bande passante et facteur de bruit), il semblerait que les prochaines améliorations soient à attendre du côté des post-traitements des SPC. Les interceptions de « grand-papa » ont peu d'avenir.

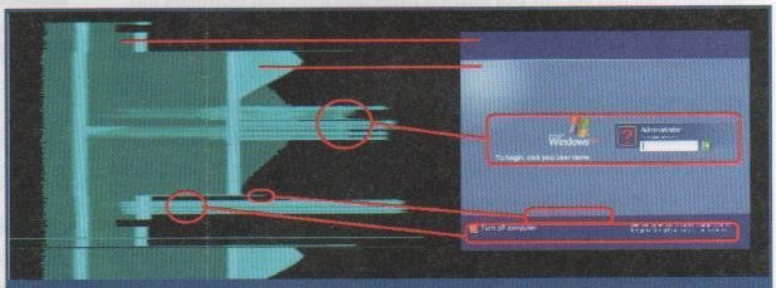


Fig. 3 : Deux trames et leur décodage

1.5 Lutte contre les SPC

1.5.1 Les 3 options de lutte contre les SPC

Après avoir pris conscience des SPC, il s'agit de s'en prémunir. À cet effet, 3 possibilités sont offertes :

- L'installation d'une cage de Faraday dans laquelle installer les équipements sensibles. Cette solution radicale a néanmoins des inconvénients : le coût, l'installation (une cage ne s'installe pas n'importe où), la maintenance (une cage s'use et a besoin d'entretien), et enfin l'ergonomie (une cage impose de travailler à longueur de journée dans une boite sans fenêtre).
- L'utilisation de matériel qualifié sur le plan Tempest. Excessivement cher (5 à 10 fois le prix d'un PC standard), cet équipement sur-blindé garantit qu'aucune information compromettante ne quittera l'ordinateur, du moins dans un périmètre de 1 mètre. Il devient alors possible de travailler presque n'importe où.

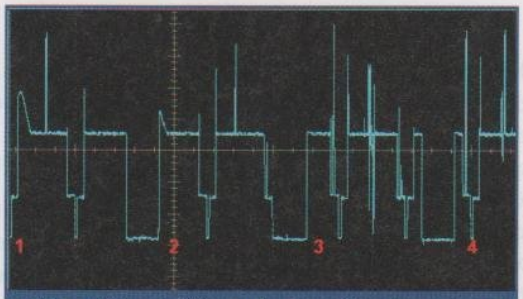


Fig. 4 : Signal vidéo analogique (Bande passante 20MHz)

L'utilisation de la démarche Tempest, à savoir, déterminer si l'utilisation d'un équipement particulier dans une pièce bien déterminée d'un bâtiment est susceptible de faire fuir de l'information au-delà de la zone de sécurité associée au bâtiment. On comprend en effet intuitivement qu'un ordinateur récent placé dans un bunker sur une base aérienne aura une probabilité d'interception plus faible qu'un vieil ordinateur placé dans un Algéco en plein centre ville. La démarche est à effectuer en 3 étapes : 1) Déterminer le niveau de classification maximum des documents à traiter. 2) Caractériser d'un point de vue électromagnétique le local dans lequel seront traitées ces informations. 3) Identifier et acheter les équipements dont le niveau de fuite électromagnétique est compatible du résultat de zonage du bâtiment.

1.5.2 Les documents réglementaires

Abordons la source des sources, la documentation de l'OTAN...

Les normes Tempest sont regroupées dans les *Secan Doctrine and Information Publications* (SDIP). Elles sont émises par SECAN (*Security and Evaluation Agency*), l'agence de sécurité et d'évaluation de l'OTAN, agence qui partage ses locaux avec la NSA à Fort Meade. Les SDIP ont remplacé en 2006 les anciennes AMSG (*Allied Military Security Guidelines*) 720, 788 et 784, en y apportant quelques évolutions. Ces documents ne sont pas publics, même si le niveau de classification des SDIP est descendu par rapport à celui des AMSG.

La SDIP n°27 concerne le zonage des équipements. Elle se décline en trois niveaux, du plus difficile à atteindre (le *level A*) au plus « laxiste » (le level C).

Cette SDIP décrit les protocoles de mesure de signaux, ainsi que les règles à appliquer pour construire un « plan de test », le document qui définit les plages de fréquences à parcourir et les niveaux d'émission à ne pas dépasser.

Les équipements « SDIP 27 level A » sont les « vrais » équipements Tempest. Le niveau d'émanation électromagnétique est difficile à atteindre et seuls des équipements « préparés » (sorties filtrées, ferrites sur câbles externes, écrans blindés) parviennent à se plier aux contraintes définies dans la norme. Ils pourront être utilisés dans les secteurs des bâtiments qui se situent en « zone 0 », c'est-à-dire dont la distance de sécurité est de 1m (en fait, à utiliser si vous ne faites pas confiance aux personnes qui se trouvent de l'autre côté du mur). Les règles de zonage des bâtiments font l'objet d'un document à part entière.

Les équipements évalués « SDIP 27 level B » sont des équipements COTS qui, de par leur construction de qualité et/ou un choix « heureux » de composants, ont un niveau d'émission suffisamment faible pour être utilisés en « zone 1 », à savoir des zones où l'atténuation des bâtiments est équivalente à l'atténuation apportée par une distance de 20m en champ libre.

Les équipements qui correspondent à la dernière catégorie, la « SDIP 27 level C », sont des équipements COTS (*Custom On The Shelf*) qui pourront être utilisés en « zone 2 », à savoir des zones où l'atténuation des bâtiments est équivalente à l'atténuation apportée par une distance de 100m en champ libre. Il est souvent question de matériel « tactique » en référence à la zone de sécurité de 100m qui devrait exister sur un théâtre d'opération, autour des équipements des centres de transmission.

SECAN maintient une liste des équipements évalués, la *Nato Recommended Product List* **[NRPL]**.

La SDIP n° 29 (ancienne AMSG 719) décrit, quant à elle, les règles d'installation des équipements évalués selon la SDIP 27. On comprend bien que l'installation d'un équipement SDIP 27 level B n'est pas autorisée près de tuyaux de chauffage reliés à une chaudière qui alimente un immeuble dont on ne connaît (contrôle) pas tous les occupants...

Par l'instruction interministérielle 300 du 20 juin 1997 **[ii300]**, la réglementation française impose la prise en compte des problèmes de compromission électromagnétique pour les documents classifiés de défense. Tout comme la réglementation européenne (qui a sa propre réglementation pour le Confidentiel UE), la réglementation française s'est longtemps inspirée de manière critique mais constructive des normes techniques de l'OTAN.

1.5.3 Durcissement

Si les performances des équipements ou des locaux ne sont pas compatibles avec le niveau de classification des informations à traiter, il est nécessaire d'en augmenter les performances, soit en sélectionnant un matériel plus performant (ce qui peut entraîner un coût prohibitif pour un passage de niveau B vers un niveau A), soit en « durcissant » les locaux. Plusieurs options existent, de la solution temporaire grâce à des tentes en tissu « faradisé » (qui ferait la joie des phobiques de l'*electrosmog* !!) **[Beam]** à la solution « long terme » grâce à des papiers-peints faradisés qui contiennent des fibres métalliques **[Dubois]**.

1.5.4 Passage en laboratoire

En France, les évaluations Tempest sont effectuées par l'État, soit par le CELAR pour les besoins des armées, soit par le laboratoire Tempest de la DCSSI pour toute demande interministérielle.

Il n'en est pas de même dans d'autres pays dans lesquels les évaluations sont effectuées par des structures privées agréées par le gouvernement.

On comprend alors que les acteurs de ce domaine soient peu nombreux et que leurs fournisseurs de matériels le soient encore moins. On citera néanmoins Rohde et Schwarz avec leur FSET **[R&S]**, un récepteur de mesure Tempest dérivé du FSEM, leur récepteur de mesure de compatibilité électromagnétique, et aussi *Dynamic Sciences*, société américaine qui exporte des récepteurs tactiques suffisamment sensibles pour les mesures normatives **[DSI]**. Sous certaines conditions, la norme (SDIP 27) autorise l'utilisation d'équipements tiers comme des récepteurs non accordables, des analyseurs de spectre (même si le FSET est en fait aussi un analyseur de spectre) ou des détecteurs à diode...

De la même manière que pour des tests de compatibilité électromagnétique, l'équipement sous test est placé en cage de Faraday pour que ses parasites y soient écoutés par un capteur (antenne, pince, cadre magnétique) en toute immunité vis-à-vis des perturbations extérieures (bruits industriels, radio FM, télévision,...).

1.5.5 Limites de l'exercice

Comme nous l'avons évoqué dans le paragraphe sur la caractérisation des signaux vidéo analogiques, la dispersion des caractéristiques des composants permet de discriminer deux écrans de même modèle, placés dans une même pièce. Le corollaire est qu'il est impossible d'affirmer que, si un écran ne rayonne pas, tous ses congénères l'imiteront. Il est alors nécessaire de procéder par échantillonnage pour vérifier l'« ergodicité » de l'hypothèse...

Une autre surprise attend les équipements de retour de maintenance. Pour de sombres raisons économiques, il est souvent plus facile de changer une carte que de réparer un composant. Cependant, il est aussi très économiquement raisonnable de ne pas utiliser des composants électroniques mono-source pour ne pas être tributaire d'un fabricant. Tolérance de caractéristiques intrinsèques des composants – composants multi-source – carte qui embarquent plusieurs composants – autant de petites dérives qui aboutissent au fait que l'équipement réparé ne possèdera plus les mêmes caractéristiques électromagnétiques qu'à l'origine. Dans la plupart des cas, ces variations sont bénignes, mais il peut se produire qu'un amplificateur « accroche » à certaines fréquences et se mette à osciller...

Enfin, que dire de l'agent de maintenance très prévenant qui saura choisir ses composants de manière à assurer le meilleur rapport signal à bruit, pour l'attaquant. Et vous l'avez vu, vous, le fil de cuivre dans le nid d'abeille de la cage de Faraday ?

2. Les à-côtés des SPC

Les quelques exemples suivants sont des notions connexes des SPC.

2.1 « Optical Tempest »

Votre écran cathodique (si vous en avez encore un) est en train de vous trahir, et dans votre dos de surcroît ! C'est ce qu'affirme Markus Kuhn dans son article de 2002 **[Kuhn]**. Son idée est que le spot d'un écran CRT ne termine pas sa course à la surface de l'écran, mais sur le mur qui lui fait face. De cette façon, il serait possible de reconstituer les images présentes à l'écran.

2.2 Intermodulation

En illuminant un équipement avec une source hyper-fréquence de forte puissance, il est possible sous certaines conditions restrictives, de faire rayonner des signaux qui ne rayonnaient pas auparavant. Le principe serait que les circuits non linéaires des équipements provoqueraient des modulations d'amplitude entre les signaux qui les traversent. Ces modulations sont ensuite captées et démodulées afin de retrouver le signal d'origine. Cette technique fort efficace sur des claviers filaires, dont les câbles et/ou les châssis métalliques doivent former de bons émetteurs/récepteurs, semble difficile à étendre à d'autres équipements. À moins qu'avec un peu d'entraînement...

Il ne faut cependant pas confondre cette « discipline » avec le dispositif acoustique utilisé par le KGB dans les années 50. Dans ce cas, il s'agissait d'un piège acoustique passif dissimulé dans un cadeau généreusement offert par les hôtes soviétiques, et qui avait la bonne idée de vibrer au diapason des conversations qui se tenaient dans l'ambassade américaine. Un puissant maser (faisceau cohérent de micro-onde) récupérait à distance par modulation ces vibrations et permettait ainsi une écoute discrète, mais non sans conséquences sur la santé des résidents de l'ambassade.

2.3 IEMN et MFP

L'impulsion électromagnétique nucléaire (IEMN) serait la conséquence de l'explosion d'une bombe nucléaire dans l'atmosphère. Cette vague électromagnétique de très forte intensité provoquerait des dysfonctionnements ou des destructions de matériels électroniques par un apport soudain d'énergie qui se dissiperait par effet joule en faisant claquer des composants.

Les micro-ondes de forte puissance sont des dispositifs capables de projeter en hyper-fréquence des puissances très importantes. Appelées armes non létales, armes à énergie dirigée ou armes sans munition, elles sont utilisées de manière affichable pour le maintien de l'ordre **[ADS1]**, mais les développements de ce domaine laissent perplexes **[ADS2]**.

Face à ce type de menace, un ordinateur Tempest résistera mieux qu'un ordinateur COTS, mais pas pendant très longtemps. En effet, le caractère Tempest d'un équipement garantit que des signaux de très faible puissance ne puissent pas quitter le blindage de l'équipement. Il n'apporte aucune protection contre les signaux de très forte puissance qui entreraient dans l'équipement (attaque *frontside*). Pour parer à ces attaques, il est nécessaire de placer des dispositifs spécifiques (éclateurs, mise à la terre,...).

3. Un peu plus près du signal

Une évaluation Tempest protège contre les fuites d'information qui pourraient se propager à l'extérieur d'un équipement. Il importe peu que l'équipement soit infesté de signaux compromettant à l'intérieur de son châssis, du moment que ces derniers ne peuvent pas en sortir.

La limite de l'exercice est le vol d'un équipement qui permettra à l'agresseur de se placer non pas à 1 mètre (comme en évaluation Tempest), mais au cœur même de l'équipement. Il n'est plus intéressé par la remontée d'un signal parasite issu d'un signal standardisé. Il peut l'obtenir directement en branchant une bretelle sur le signal intéressant. Ce qu'il recherche, ce sont les signaux parasites qui l'informeront sur l'activité calculatoire d'une carte ou d'un composant cryptographique. Ce sont soit des secrets (clefs, *passphrase*), soit des mécanismes logiciels de sécurité à contourner, en perturbant le matériel. Il va exploiter les « effets de bords » provoqués par l'implémentation physique de processus abstraits. On parle alors d'attaques à canaux auxiliaires.

De manière identique aux « attaques Tempest » dans lesquelles un agresseur écoute des signaux pour en ressortir de l'information, l'observation passive de parasites internes à un équipement est une source d'information très riche.

Que ce soit par observation de la consommation instantanée de courant avec un simple *shunt* ou par écoute du rayonnement électromagnétique en champ proche avec un mini-solénoïde, il est possible de prélever des signaux corrélés avec l'activité d'un composant électronique.

La vulnérabilité de ces systèmes vient du fait que lorsqu'un processus abstrait ou un algorithme est implémenté dans un équipement, il fait fuir des informations en offrant accès à ses calculs intermédiaires. C'est ce qui se passait, il y a quelques années, sur les implémentations naïves de l'algorithme RSA, trahies par leurs mécanismes internes d'exponentiation modulaire, le fameux « *Square & Multiply* » dont la consommation électrique instantanée permettait de lire directement les bits de la clef secrète **[S&Mult]**.

En 1999, la discipline fait un grand bond en avant avec l'article « *Differential Power Analysis* » de Paul Kocher, la célèbre DPA **[DPA]**. Cette technique présuppose la connaissance de l'algorithme attaqué afin de créer une fonction de discrimination. Cette dernière permet d'effectuer des tris statistiques judicieux aboutissant à révéler 6 bits de clef par attaque réussie.

D'autres techniques utilisent les outils des canaux auxiliaires pour « reverser » des morceaux d'algorithmes **[Scare]** & **[Fire]**.

Les contre-mesures vont bon train. À la manière des ingénieurs Tempest qui protègent les « conductions lignes noires » de leurs équipements (les câbles d'alimentation par exemple), les microélectroniciens « tempestisent » leurs puces de manière à ce que l'analyse des courants de consommation ne révèle pas de secret.

Mais, le répit est de courte durée et, à la manière des agresseurs qui s'approchent toujours plus près de la source d'information, des techniques de micro-antennes sont développées afin de venir se placer au-dessus des composants pour ne prélever que l'activité du cryptoprocesseur ou celle d'un bus mémoire **[EMA]**.

Cette mise en abyme nous rappelle que la sécurité est un perpétuel recommencement.

Conclusion

Le Tempest est mort, vive le Tempest !

La menace Tempest comme elle est décrite dans les articles déclassifiés de la NSA **[NSA1]** n'existe plus. Fort heureusement d'ailleurs, puisque l'article date de... 1973 ! De nos jours, il est rare de rencontrer des télétypes, et le niveau de consommation électrique des ordinateurs est à la baisse.

En contrepartie, cette menace, qu'il faudrait requalifier, de « compromission électromagnétique » prend de nouvelles formes :

- Elle change d'échelle en s'intéressant aux émissions électromagnétiques des puces de chiffrement et non plus à celles des équipements eux-mêmes.
- Elle s'intéresse à nos *Personal Area Network*, car, en toute insouciance, nous sommes dépendant d'une multitude d'équipements électroniques dont nous ne supposons pas la zone de couverture. À votre avis, en disposant d'équipement d'interception de type Tempest, à quelle distance pourrait-on intercepter la communication d'une oreillette de portable bluetooth placée à proximité d'un tuyau de chauffage central ? Et quid de votre réseau CPL ou WiFi ?

Bien entendu, ces communications peuvent être protégées par un protocole cryptographique et l'exploitation directe du signal intercepté n'est en général pas possible... Au fait, vous vous souvenez du *Hitchhikking* **[NSA1 p93]** ? ■

Références

Les références de cet article sont disponibles sur miscmag.com/ref44.

Martin Vuagnoux et Sylvain Pasini

ÉMANATIONS ÉLECTROMAGNÉTIQUES COMPROMETTANTES DES CLAVIERS FILAIRES ET SANS FIL

mots-clés : *claviers / signaux compromettants / TEMPEST / ondes électromagnétiques*

Les claviers d'ordinateurs sont souvent utilisés pour transmettre des informations sensibles comme des mots de passe. Puisqu'ils sont constitués de composants électroniques, les claviers émettent inévitablement des ondes électromagnétiques. Ces émanations peuvent être compromettantes en révélant par exemple quelle touche a été frappée. Dans cet article, nous décrivons une nouvelle méthode pour détecter les éventuels signaux compromettants d'appareils électroniques. Nous avons appliqué cette méthode aux claviers d'ordinateurs filaires et sans fil et nous avons découvert quatre différentes techniques qui reposent sur quatre différents signaux compromettants, permettant de recouvrer partiellement ou complètement les touches frappées à distance. Tous les claviers testés (PS2, USB, sans fil, ordinateurs portables) sont vulnérables à au moins une des quatre techniques. La meilleure attaque permet de recouvrer plus de 95 % des frappes d'un clavier à plus de 20 mètres de distance. Nous en concluons que les claviers actuellement utilisés ne sont généralement pas suffisamment protégés contre ce type d'attaque.

1. Introduction

En 2006 à Cambridge (UK), la police fait état d'un nouveau type de cambriolage. Des voleurs armés d'appareils Bluetooth tentent de détecter d'éventuels téléphones portables ou ordinateurs laissés dans les voitures des parkings. Cela leur évite de fracturer inutilement les voitures. Le protocole Bluetooth est bien évidemment prévu pour fonctionner par onde radio. La communication peut être chiffrée et authentifiée, mais l'émission d'un signal électromagnétique caractéristique rend le système traçable. Ainsi, l'onde émise par un appareil Bluetooth peut alors devenir compromettante bien qu'elle ne transporte aucune information sensible. Cet exemple met en lumière les difficultés à déterminer si un signal est compromettant ou non. Une simple lampe allumée dans un appartement indique la présence de personne(s) avec une certaine probabilité. S'agit-il de signaux compromettants ?

1.1 Historique

À la fin du 19ème siècle, l'utilisation massive du téléphone a entraîné la mise en place d'un réseau câblé dense. Dans un premier temps, ces câbles n'étaient pas torsadés. Ainsi, lorsque deux câbles proches étaient utilisés en même temps, un couplage par induction pouvait transférer les signaux d'un fil à l'autre. De cette manière, les interlocuteurs entendaient, bien malgré eux, la conversation d'autres pairs.

1.1.1 Première Guerre mondiale

La première exploitation militaire d'émanations électromagnétiques concerne également la téléphonie. Durant la première Guerre mondiale, les soldats allemands et français utilisaient un système de communication téléphonique basé sur un seul fil pour recevoir les ordres depuis le quartier général jusqu'aux tranchées les plus exposées, l'objectif étant de réduire la masse transportée par l'officier de communication. Le fil conducteur de retour était simplement une longue tige de métal, plantée profondément dans le sol. Lorsque les tranchées allemandes et françaises se trouvaient suffisamment proches et lorsque l'humidité du sol était adéquate, une boucle de masse partagée se formait entre les piques métalliques allemandes et françaises. En pratique, ce dispositif permit aux allemands d'écouter les conversations françaises. Les Français puis les Anglais ne tardèrent pas à découvrir également ce type de signaux compromettants. Rapidement, tous développèrent des postes d'écoute avancés ainsi que des contre-mesures : les consignes furent de planter simplement la pique de terre à plusieurs centaines de mètres en arrière du front.

1.1.2 Bell 131-B2

Le premier exemple d'exploitation automatisée d'ondes électromagnétiques compromettantes concerne le périphérique de chiffrement Bell 131-B2. Cet appareil était utilisé par les Américains durant la seconde Guerre mondiale pour fournir un chiffrement par flot aux téléscripteurs de l'armée Sigaba 134C. En 1943, un ingénieur des laboratoires Bell découvrit par hasard que lorsque cette machine à chiffrer passait d'un état interne à un autre, un pic apparaissait sur l'écran d'un oscilloscope distant de plusieurs mètres et non relié au système. Il en fit part à ses supérieurs qui lui demandèrent une preuve concrète du risque lié à ce signal compromettant. Plusieurs ingénieurs de la société Bell décidèrent d'enregistrer le signal émis par un Bell 131-B2 utilisé par le centre de cryptologie voisin, situé en face de la rue et distant de plusieurs dizaines de mètres. Pendant une heure, les ingénieurs capturèrent le signal émis par le périphérique de chiffrement, puis, après moins de quatre heures d'analyse, ils purent recouvrer 75% du texte clair **[TEMPEST07]**.

Cette anecdote a sûrement motivé l'élaboration du standard TEMPEST. Ce standard est encore utilisé dans sa version actuelle pour définir le rayonnement électromagnétique (entre autres) acceptable pour un appareil certifié TEMPEST. Les valeurs limites ainsi que la procédure de mesure sont toujours classifiées. Toutefois, certains appareils de mesure comme le R-1250 ou le R-1550A de Dynamic Sciences, se targuent d'être compatibles avec les seuils demandés par cette norme.

1.1.3 Projet Mogul

Les ondes électromagnétiques ne représentent pas le seul type d'émanations compromettantes. Les émissions acoustiques peuvent être étonnamment efficaces.

Le projet MOGUL a été conçu par Maurice Ewing de l'Université de Columbia à New York. Ce projet s'appuie sur une autre recherche effectuée durant la deuxième Guerre mondiale. Lorsqu'un pilote américain était abattu au-dessus de l'océan Pacifique, il ne devait en aucun cas utiliser sa radio pour émettre un message de détresse. L'ennemi aurait vite fait de le localiser à l'aide de radiogionomètres. Il avait pour consigne de jeter à la mer une sphère métallique d'une quinzaine de centimètres. Il s'agissait d'une sphère creuse, ne contenant aucun système électrique ou mécanique. Après plusieurs heures, les secours arrivaient et récupéraient le pilote.

Cette sphère, appelée bombe SOFAR (*SOund Fixing And Ranging channel*) reposait sur une recherche de Maurice Ewing concernant la vitesse de propagation du son dans l'eau. En effet, cette vitesse dépend de la pression ainsi que de la température de l'eau. À environ 1 km de profondeur, la vitesse du son est donc minimale. Ainsi, un canal sonore se forme dans lequel le son se propage horizontalement par réfraction sans atteindre le fond de l'océan, ni la surface. De cette manière, l'énergie étant relativement bien conservée, les propagations peuvent aller de l'Australie jusqu'à la côte ouest des États-Unis. Ainsi, le diamètre et l'épaisseur de la sphère étaient élaborés pour imploser à une profondeur de 1 km et créer une onde de choc dans ce canal sonore. Plusieurs hydrophones placés dans l'océan permettaient par la suite de localiser la position de l'implosion par triangulation **[SOFAR07]**.

Maurice Ewing remarqua qu'un phénomène équivalent existait dans l'atmosphère. Le projet top secret Mogul **[Muller06]** était donc de placer à haute altitude une série de microphones suspendus par des ballons. L'objectif étant de détecter l'explosion d'une éventuelle bombe atomique russe. Le projet fut mis en place en 1947, puis arrêté en 1949 après la première explosion nucléaire russe. Il s'agit donc d'une exploitation des émanations acoustiques d'une explosion nucléaire, afin de la détecter, puis de la localiser par triangulation en utilisant plusieurs microphones. Un rapport de l'armée américaine, rendu public en 1995, détermine l'origine des débris potentiels d'une soucoupe volante, découverts à Roswell le 7 juillet 1947. Il ne s'agissait que de la structure porteuse d'un microphone géant du projet Mogul suspendu à plusieurs ballons. Par ailleurs, les ballons météorologiques doivent

beaucoup au projet Mogul. Après 1949, les États-Unis utilisèrent des capteurs sismiques pour détecter les explosions nucléaires. Cette méthode s'est avérée bien plus efficace.

1.1.4 Japon

En 1962, un officier de l'armée américaine assigné à inspecter les environs d'un petit centre de cryptologie au Japon, découvrit de l'autre coté de la rue, plusieurs antennes sur le toit d'un hôpital contrôlé par le gouvernement japonais. Toutes ces antennes étaient dirigées sur l'émetteur principal de Tokyo, sauf une antenne qui ciblait le centre de cryptologie. Il s'agit de la première attaque utilisant des signaux compromettants dirigée sur les États-Unis.

1.1.5 Ambassade américaine à Moscou

En 1964, les services secrets américains découvrirent plus de 40 microphones dissimulés dans l'ambassade américaine à Moscou. Ils trouvèrent également de larges grilles de métal encastrées dans le plafond des salles accueillants les machines à chiffrer. Reliées à des fils accessibles depuis l'extérieur de l'ambassade, ces grilles étaient utilisées comme antennes pour détecter d'éventuels signaux électromagnétiques compromettants.

1.1.6 Ambassade française à Londres

Un autre exemple relatif à l'utilisation d'ondes électromagnétiques fut donné par Peter Wright, dans son livre *SpyCatcher* **[Write87]**. Ancien agent du MI5, Peter Wright y décrit les diverses opérations qu'il a effectuées durant les années 60 pour le gouvernement anglais. Il y détaille en particulier une attaque perpétrée sur l'ambassade française à Londres. Confronté à la résistance du chiffre français, le gouvernement anglais n'était pas en mesure de décrypter les communications entre l'ambassade française et Paris. Les ingénieurs du MI5 et du GCHQ cherchèrent alors à exploiter d'éventuels rayonnements compromettants. En se connectant directement sur le câble de communication, ils constatèrent qu'en plus du signal chiffré, un autre signal plus faible encodait également le texte clair. Celui-ci était vraisemblablement généré par le téléscripteur recevant le signal en clair proche de la machine à chiffrer. Notons que ce périphérique était placé dans une autre salle pour éviter ce genre de pollution de signal. Toutefois, une émanation électromagnétique compromettante passait d'une chambre à l'autre. Le MI5 fut alors en mesure de raccorder directement ce signal à un téléscripteur et put de cette manière automatiser la procédure. Pendant près de trois ans, de 1960 à 1962 le MI5 et le GCHQ ont pu lire toutes les transmissions qui passaient par l'ambassade de France, ce qui leur permis de connaître le moindre mouvement des Français pendant la tentative infructueuse des Anglais d'accéder au Marché commun. Les Affaires étrangères anglaises réceptionnaient ainsi la copie conforme des télex envoyés par le Général de Gaulle. Cette mission portait le nom de code Opération Stockade.

1.1.7 Guerre du Vietnam

La guerre du Vietnam est sans aucun doute le premier conflit où l'écoute et l'analyse d'émanations compromettantes furent massivement utilisées. La route de Hô Chi Minh qui relie le nord et le sud du Vietnam en passant par le Laos et le Cambodge fut surveillée en utilisant toute une série de nouvelles technologies pour l'époque. Sous le nom de « Igloo White » **[IglooW09, Mark95]**, les Américains développèrent la première guerre électronique à l'aide de senseurs, aussi bien acoustiques que sismiques, pour détecter la présence de soldats ou de véhicules jusqu'a 1 km. Ces senseurs, largués depuis des hélicoptères, se plantaient dans le sol et renvoyaient des informations sur différentes fréquences radio. Ils étaient capables de fonctionner pendant plus d'un mois. En 1972, un senseur d'un autre type, appelé le Corbeau noir ou plus simplement EDET III fut utilisé pour détecter certaines émanations électromagnétiques compromettantes. Les soldats américains remarquèrent que lorsqu'un camion russe, utilisé par l'armée nord-vietnamienne, était mis en marche, son démarreur électrique émettait un rayonnement très puissant détectable jusqu'à 16 kilomètres de distance. Ce senseur, planté dans le sol ou monté sur un avion Hercule C-130 pouvait alors facilement localiser le démarrage d'un camion ennemi, puis envoyer un missile sur cette position. Ce type d'attaque fut rapidement connue des nord vietnamiens, à travers des publications scientifiques de l'armée américaine. Les camions russes furent alors blindés au niveau du démarreur électrique, pour éviter ce type d'émanation électromagnétique compromettante.

1.2 Recherche académique

Au niveau des recherches académiques, l'analyse de signaux compromettants a commencé il y a environ 30 ans. Deux types de rayonnements sont analysés : les émanations proches et les émanations lointaines.

Les émanations proches sont analysées à l'aide de sondes généralement magnétiques, placées au-dessus des microprocesseurs cryptographiques ou des cartes à puce. L'objectif est généralement de découvrir une clef secrète utilisée par un algorithme.

Les émanations lointaines s'illustrent par l'analyse des signaux compromettants captés à partir d'une plus grande distance. On peut citer l'expérience de Win Van Eck, qui démontra en 1985 lors d'une émission télévisée comment lire le contenu affiché sur un écran cathodique à plusieurs dizaines de mètres. Markus Kuhn améliora cette attaque et fut capable de lire le contenu d'écrans LCD utilisant une connectique VGA ou DVI **[Kuhn03]**.

D'autres travaux récents nous donnent des preuves indiscutables de la dangerosité de ces émanations compromettantes, comme le bruit d'une touche frappée sur un clavier **[AsonovA04]**, le rayonnement d'une ligne RS-232 **[Smulders90]**, le son émis par un processeur durant le calcul d'un algorithme de chiffrement révélant une clef secrète

[TromerS06], l'analyse des rayonnements électromagnétiques des cartes accélératrices de chiffrement [GandolfiMO01] ou des FPGA [MulderBPV07]. Cette liste est bien évidemment non exhaustive.

Cet historique met en lumière le rôle du hasard dans la découverte de ces rayonnements compromettants, en particulier les émanations électromagnétiques. En effet, contrairement aux émanations compromettantes acoustiques ou visuelles, l'homme n'est pas naturellement équipé pour les détecter.

2. Acquisition de signaux compromettants

Nous passons maintenant en revue les méthodes actuellement utilisées pour découvrir ces rayonnements électromagnétiques compromettants. Nous proposons également une méthode améliorée, qui sera appliquée à un cas particulier, l'analyse des signaux compromettants électromagnétiques des claviers filaires et sans fil.

De nos jours, tout appareil électronique doit subir une série de tests afin de garantir que celui-ci n'interfère pas avec les appareils électroniques environnants. Définies par la Commision électrotechnique internationale, ces normes établissent les limites aux perturbations électromagnétiques tolérables dans un réseau électrique. Ainsi, chaque appareil électrique se doit d'être testé au niveau de sa compatibilité électromagnétique.

2.1 Types de rayonnement

On caractérise généralement deux types de perturbations. Le couplage par conduction et le couplage par rayonnement. Ce dernier se décompose en deux modes de propagation.

2.1.1 Mode différentiel

La propagation en mode différentiel se caractérise par la circulation du courant dans les boucles formées par les conducteurs du circuit (voir Figure 1). Ces boucles réagissent comme de petites antennes circulaires et rayonnent inévitablement. Il est relativement facile de se débarrasser des rayonnements en mode différentiel en blindant le système ou en réduisant la taille des boucles.

2.1.2 Mode commun

Le rayonnement en mode commun est le résultat d'une chute de tension non désirée dans le circuit. Ces chutes de tension apparaissent généralement dans le circuit de masse, par exemple lorsqu'un transistor ouvre ou ferme un circuit. Si des câbles externes sont reliés au système, ceux-ci peuvent se comporter comme des antennes excitées par ces chutes de tension (voir Figure 2). Les rayonnements en mode commun sont difficiles à prévoir et sont généralement détectés de manière empirique lors des tests de compatibilité électromagnétique.

La détection de signaux compromettants n'est qu'un sous-ensemble du domaine de la compatibilité électromagnétique. Du point de vue d'un attaquant, on peut définir deux types d'émanations compromettantes en fonction de leur origine.

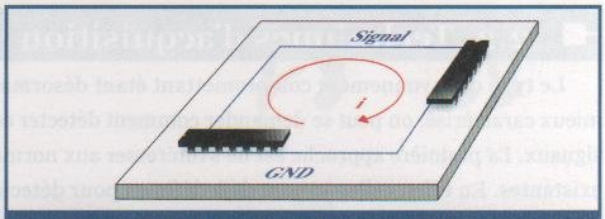


Figure 1 : Une boucle formée par un circuit peut se comporter comme une antenne circulaire et émettre un rayonnement en mode différentiel.

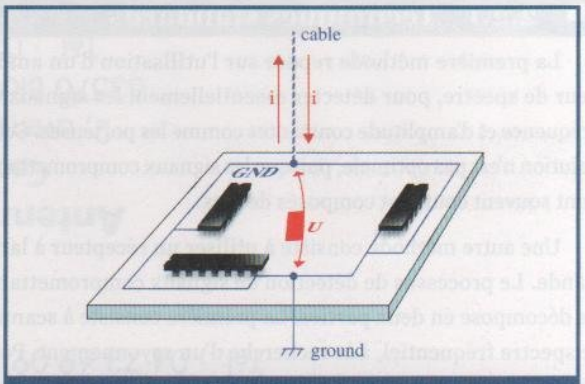


Figure 2 : Exemple de rayonnement en mode commun. La chute de tension excite le câble de connexion qui se comporte comme une antenne (dipôle).

2.1.3 Émanations directes

Les appareil électroniques numériques encodent généralement les données à l'aide d'états logiques. Le passage entre ces états se caractérise par un flanc montant ou descendant. Cette brusque transition génère inévitablement une onde électromagnétique à une fréquence maximale déterminée par la durée de la transition de flanc. Ce rayonnement est directement lié à l'état du signal transportant les données, c'est pourquoi il est appelé rayonnement direct.

2.1.4 Émanations indirectes

Les émanations électromagnétiques peuvent interagir à l'intérieur d'un circuit électronique. En particulier, certains composants induisent et modifient ces radiations. Ces émanations non désirées peuvent se manifester sous la forme

de modulations ou inter-modulations (phase, amplitude ou fréquence) ou de porteuses, par exemple générées par un signal d'horloge ainsi que ses harmoniques. De plus, un couplage non linéaire entre des signaux ou la pollution de la masse à travers l'alimentation électrique peuvent générer ces rayonnements. Ces émanations indirectes sont si difficiles à prévoir qu'elles sont généralement découvertes lors des tests de compatibilité électromagnétiques tels que FCC, CISPR, MIL-STD-461 ou encore TEMPEST/NACSIM-5000.

2.2 Techniques d'acquisition

Le type de rayonnement compromettant étant désormais mieux caractérisé, on peut se demander comment détecter ces signaux. La première approche est de s'intéresser aux normes existantes. En effet, celles-ci sont déjà définies pour détecter certains signaux compromettants. Deux techniques sont généralement utilisées pour détecter les émanations électromagnétiques compromettantes.

2.2.1 Techniques standards

La première méthode repose sur l'utilisation d'un analyseur de spectre, pour détecter essentiellement les signaux de fréquence et d'amplitude constantes comme les porteuses. Cette solution n'est pas optimale, puisque les signaux compromettants sont souvent courts et composés de pics.

Une autre méthode consiste à utiliser un récepteur à large bande. Le processus de détection de signaux compromettants se décompose en deux parties. La première consiste à scanner le spectre fréquentiel, à la recherche d'un rayonnement. Pour ce faire, le récepteur ramène le signal dans une bande de base qui est ensuite analysée. Cette bande de base caractérise la bande passante du récepteur. La deuxième étape correspond au choix éventuel de la démodulation (amplitude, fréquence, phase, etc.). En effet, certains rayonnements contiennent une information compromettante sous la forme d'une modulation. C'est le cas des attaques sur écran de Markus Kuhn **[Kuhno3]** par exemple. En pratique, des récepteurs à large bande comme le R-1250 ou le R-1550A de Dynamic Sciences sont optimaux, car ils respectent les normes secrètes TEMPEST et fournissent donc une sensibilité accrue aux signaux faibles. D'autres solutions moins coûteuses, comme le récepteur à large bande USRP (*Universal Software Radio Peripheral*) combiné au projet GNU Radio **[GNURADIO]** est capable de scanner des fréquences jusqu'à 3 GHz. L'avantage de cette solution est qu'une partie du traitement du signal s'effectue en software. De cette manière, les processus de filtrage ou de décodage peuvent rapidement être modifiés. Malheureusement, la bande passante et la sensibilité du convertisseur analogique-numérique sont plus petites.

2.2.2 Nouvelle technique

Certains signaux compromettants, directs ou indirects peuvent ne pas être détectés par les deux méthodes décrites ci-dessous. En effet, si un signal est composé de pics irréguliers ou d'une porteuse pendant un très court instant, il ne sera vraisemblablement pas détecté par un analyseur de spectre. De plus, le processus de balayage de fréquence nécessaire lorsqu'on utilise un récepteur à large bande prend un temps non négligeable. En outre, l'utilisation d'une bande de base ainsi que le processus de démodulation peuvent cacher certains signaux.

Nous proposons d'utiliser une méthode plus directe, sans modifier le signal par démodulation, ni en ramenant le signal dans une bande de base. Premièrement, nous capturons le signal directement à la sortie de l'antenne, de manière à ne pas être limité par une bande passante. Ensuite, nous calculons la transformée de Fourier de ce signal en fonction d'une fenêtre temporelle réduite (transformée de Fourier locale). Ainsi, on obtient une visualisation du signal en trois dimensions en fonction de la fréquence, du temps et de l'énergie.

Les convertisseurs analogique-numérique actuels permettent d'obtenir un taux d'échantillonnage de l'ordre du GS/s (Giga échantillon par seconde). Ainsi, une fois le signal converti en données numériques, des bibliothèques de traitement du signal, notamment GNU Radio peuvent être utilisés pour instantanément révéler d'éventuelles émanations compromettantes. Contrairement aux solutions précédentes, la transformée de Fourier locale va afficher tous les signaux dans une bande de fréquence déterminée par la moitié du taux d'échantillonnage, selon le théorème de Nyquist, qu'ils soient composés de pics ou de porteuses.

Malheureusement, il n'est pas possible de transférer cette masse de données directement sur un ordinateur. Le taux de données est trop élevé pour des protocoles comme USB 2.0, IEEE 1394, Gigabit Ethernet, Serial Ata, etc. Cependant, avec un algorithme de déclenchement de capture (*trigger*) spécifique, il est possible de n'échantillonner qu'une petite partie du signal et de stocker les données dans une mémoire à écriture rapide. Ensuite, ces données sont transférées sur un ordinateur et analysées à l'aide de bibliothèques de traitement du signal. Notons qu'aucune démodulation et aucun traitement n'est effectué sur le signal capturé. Il s'agit d'émanations brutes, limitées seulement par les caractéristiques de l'antenne.

L'avantage de cette méthode est que le cerveau humain est naturellement doué pour détecter visuellement des séquences. Ainsi, grâce à une seule capture représentée sous la forme d'une transformée de Fourier locale, il est possible de mettre en lumière un nombre important de signaux compromettants.

3. Découverte de signaux compromettants

Pour découvrir les émanations électromagnétiques compromettantes rayonnées par les claviers, nous avons premièrement disposé un clavier de type PS/2 sur une table en bois, à une hauteur de 1 mètre, dans une chambre semi-anéchoïque de 7 x 7 mètres. L'antenne fut disposée à 5 mètres du clavier. Il s'agit d'une antenne biconique typiquement utilisée lors de tests de compatibilité électromagnétique (50 ohms, VHA 9103 Dipol Balun). Comme convertisseur analogique-numérique, nous avons utilisé un oscilloscope Textronix TDS5104. En effet, ce modèle permet de capturer un signal à 5 GS/s, soit une bande passante de 2.5 GHz. Muni d'une mémoire de 1 Mpoints, cet oscilloscope permet également de définir un modèle complexe de déclenchement de mesure (trigger). De plus, il possède une interface IEEE 488 *General Purpose Interface Bus* (GPIB) qui permet d'extraire le contenu de la mémoire sur un ordinateur. Nous avons développé une application sous GNU/Linux pour récupérer ce signal, puis nous avons calculé la transformée de Fourier locale de ce signal. La figure 3 nous donne un résultat graphique en fonction de la fréquence, du temps et de l'amplitude du signal. À partir de cette image, il est possible d'extraire 4 différents signaux compromettants, permettant de recouvrer partiellement ou complètement la valeur de la touche frappée.

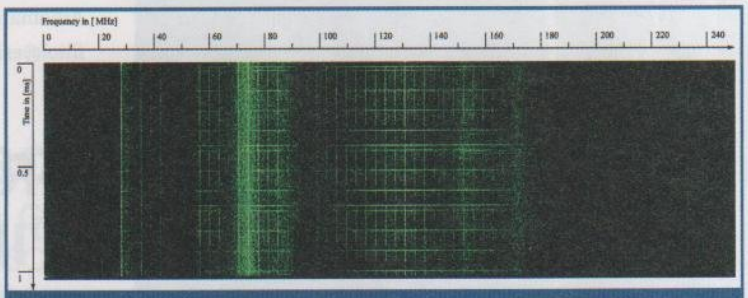


Figure 3 : Transformée de Fourier locale, calculée sur le signal brut capturé depuis une antenne biconique à 5 mètres d'un clavier PS/2 dans une chambre semi-anéchoïque.

Dans la deuxième partie de cet article, nous allons découvrir comment identifier et surtout exploiter ces quatre signaux compromettants sur des claviers de type PS/2, USB, sans-fil ou encore intégré à un ordinateur portable. Nous verrons que ces émanations sont aussi bien directes qu'indirectes. Nous verrons également comment ces émanations électromagnétiques peuvent être exploitées dans quatre environnements différents. D'une chambre semi-anéchoique à des locations en condition réelle telles qu'un appartement au cinquième étage d'un immeuble alors que l'antenne se trouve dans la cave. Nous verrons que la meilleure attaque permet de recouvrer plus de 95% du texte frappé sur un clavier d'ordinateur distant de plus de 20 mètres. ■

Références

[TEMPEST07] National Security Agency, « TEMPEST: A Signal Problem », 2007, http://www.nsa.gov/public_info/_files/cryptologic_spectrum/tempest.pdf

[SOFAR07] Historic Naval Ships Association, SOFAR, « Harbor Defense and other Sonar Systems », 2007, http://hnsa.org/doc/sonar/chap16.htm

[Muller06] MULLER (Richard A.), « Physics for Future Presidents - C10 », University of Berkeley, 2006. http://video.google.com/videoplay?docid=-2334857802602777622

[Write87] WRIGHT (Peter), *Spycatcher - The Candid Autobiography of a Senior Intelligence Officer*, William Heinemann Australia, ISBN 0-7923-7978-0, 1987.

[IglooW09] Wikipedia, the free encyclopedia, « Operation Igloo White », juin 2009, http://en.wikipedia.org/wiki/Operation_Igloo_White

[Mark95] MARK (Eduar), *Aerial Interdiction : Air Power and the Land Battle in Three American Wars*, DIANE Publishing, ISBN 0788119664, 9780788119668, 1995

[Kuhn03] KUHN (Markus G.), « Compromising Emanations : Eavesdropping Risks of Computer Displays », Technical Report UCAM-CL-TR-577, University of Cambridge, Computer Laboratory, 2003, http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-577.pdf

[AsonovA04] ASONOV (Dimitri) et AGRAWAL (Rakesh), « Keyobard Acoustic Emanations, IEEE Symposium on Security and Privacy », *IEEE Computer Society*, 2004, p. 3-11.

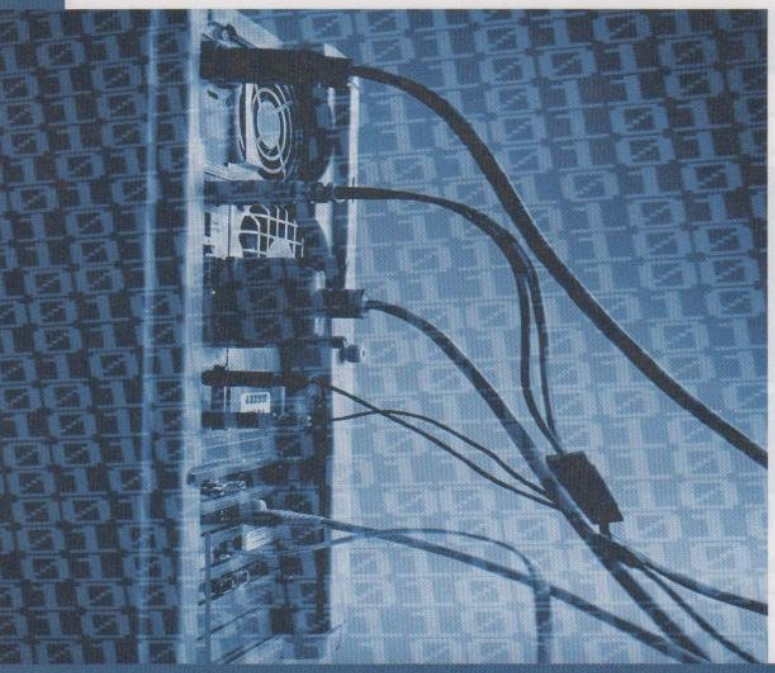
[Smulder90] SMULDERS (Peter), « The Threat of Information Theft by Reception of Electromagnetic Radiation from RS-232 Cables », *Computers and Security*, 1990, p. 53-58.

[TromerS06] SHAMIR (Adi) et TROMER (Eran), « Acoustic Cryptanalysis – On Nosy People and Noisy Machines », 2006, http://people.csail.mit.edu/tromer/acoustic/

[GandolfiMO01] GANDOLFI (Karine), MOURTEL (Christophe) et OLIVIER (Francis), « Electromagnetic Analysis: Concrete Results, Cryptographic Hardware and Embedded Systems », Lecture Notes in *Computer Sciences*, Springer, CHES 2001, p. 251-261.

[MulderBPV07] MULDER (Elke de) et BERNAORS (Siddika) et PRENEEL (Bart) et VERBAUWHEDE (Ingrid), « Differential Power and Electromagnetic Attacks on a FPGA implementation of Elliptic Curve Cryptosystems », *Computer & Electrical Engineering 33*, 2007, p. 367-382.

[GNURADIO] The GNU Software Radio, http://www.gnuradio.org



Eric Filiol
Laboratoire de virologie et de cryptologie opérationnelles (V + C)°
École Supérieure en Informatique, Électronique et Automatique – Laval
filiol@esiea.fr

ORGANISER LA FUITE D'INFORMATION D'UN POSTE ISOLÉ : MÉTHODES LOGICIELLES

mots-clés : *canaux cachés / espionnage / TEMPEST / virus / évasion de données / détournement matériel / attaque informatique*

Les articles précédents ont montré que les signaux parasites compromettants (SPC) constituent une menace très sérieuse permettant de collecter passivement des informations traitées dans un système d'information [N1]. Dans ces articles, il a également été montré que des protections existent couplant des dispositifs matériels (blindage TEMPEST, cages de Faraday [N2]), mesures techniques (zonage TEMPEST, chiffrement) et organisationnelles (règles d'emploi des matériels, procédures). Mais, cette réglementation, établie au niveau du SGDN, ne prend en compte que les attaques passives. Or, tout modèle de sécurité se doit de considérer les attaques actives. Dans le cas du Tempest, il est possible de concevoir des codes malveillants créant ou amplifiant des SPC, exploitant le fait que le logiciel pilote le matériel. Mais, un attaquant peut faire encore mieux et, dans une certaine mesure, plus facilement. De simples méthodes logicielles – typiquement un « simple » virus – permettent de faire sortir des informations d'un poste qui n'est pas en réseau. Dans cet article, nous présentons quelques-unes de ces méthodes.

Les modèles de sécurité font, de la séparation physique, une mesure radicale et définitive protégeant les systèmes contre les risques extérieurs, en particulier actifs. Dans l'esprit des utilisateurs avec une écrasante majorité, la déconnexion du réseau est la pierre philosophale de la sécurité. Cependant, toute personne ayant une bonne connaissance des techniques de captation et de traitement des SPC sait que cela n'est déjà plus vrai concernant les attaques passives. Mais, cette connaissance est malheureusement encore restreinte au monde discret de la sécurité gouvernementale et, hormis les entreprises travaillant sur du classifié de défense (et encore pas toutes), la plupart des entreprises – en particulier les petites entreprises de haute technologie – ne font pas l'objet d'une sensibilisation et d'une protection de type TEMPEST.

En revanche, dans le domaine des attaques actives, créer ou amplifier un signal de type SPC ou exploiter un canal caché pouvant y être assimilé est un domaine pratiquement inconnu ou, si certaines recherches sont menées, très peu sont en réalité publiées. Cela concerne en particulier les attaques actives mises en œuvre par des méthodes logicielles. Les attaques reposant sur le matériel, si elles sont loin d'être impossibles, impliquent plusieurs aspects qui relèvent très vite du monde du renseignement et ne peuvent être décrites ici **[N3]**.

Tout d'abord, il est essentiel de rappeler que le modèle de von Neumann – lequel décrit tous nos systèmes d'information – s'oppose au principe de l'isolation physique d'un système d'information : en effet, l'unité de traitement ne fonctionne qu'en liaison avec une unité d'entrée-sortie. L'expérience montre qu'un système n'est jamais véritablement isolé de l'extérieur, au moins ponctuellement. Imaginez ce qu'il est possible de faire avec un CD-ROM ou une clef USB faussement oubliée et avec un peu d'ingénierie sociale. Les entreprises se font prendre tous les jours à ce jeu-là. Il est toujours possible d'introduire un code malveillant dans la plupart des systèmes. Nous considérons que c'est le cas et nous traitons le challenge suivant : comment organiser la fuite de données sensibles d'un ordinateur qui n'est pas connecté à un quelconque réseau. Ce cas se généralise à celui d'un poste en réseau, mais pour lequel toutes les liaisons sont chiffrées et donc a priori inutilisable pour ce genre de techniques. La solution que nous présentons consiste à utiliser un canal caché ou quelque chose qui y ressemble... ou bien en créer un, uniquement par des méthodes logicielles.

Nous présentons quelques méthodes originales que nous avons créées et testées avec succès. Sans perte de généralités, nous avions pour objectif de faire évader une séquence d'une vingtaine d'octets représentant un login/mot de passe (donnée critique par essence). Dans le cas de données plus volumineuses (fichiers), il existe bien sûr d'autres possibilités comme le fractionnement sur plusieurs canaux, l'étalement dans le temps...

1. Travaux et résultats antérieurs

Par manque de place, nous ne rappellerons pas les quelques techniques existantes dans le domaine de la création ou l'amplification de signaux type SPC ou assimilés, ou les techniques qui s'en rapprochent. Soit elles ont été présentées dans ce dossier, soit le lecteur pourra les trouver dans la littérature de référence, soit encore elles sont détaillées dans [1].

Résumons-en les principales :

- Travaux de Van Eyck en 1985 **[2]** traitant des émissions à partir des écrans type CRT. Il s'agit des travaux les plus anciens et qui ont initié les travaux de Kuhn et Anderson.
- Travaux de Kuhn et Anderson **[3]** (écrans type CRT) et Kuhn **[4,5]** (écrans LCD). Les travaux de 1998 ont permis à ces deux auteurs de produire des fontes dont le rapport signal à bruit est tel qu'il limite naturellement les capacités d'interception et de traitement des SPC. Il est intéressant de noter qu'une approche inverse consistant au contraire à accroître ce rapport signal/bruit est de nature à amplifier ce signal. Ce genre d'attaque peut alors être mise en œuvre par un code malveillant installant (ou substituant) des fontes « malicieuses » (dans les répertoires concernés soit de l'OS, soit d'une application donnée : gestionnaire PDF ou suite Office).
- Travaux de E. Thiele avec *Tempest for Eliza* consistant à créer un signal à partir de données de telle sorte que leur visualisation sur un écran CRT provoque un signal radio AM en modulation d'amplitude **[6]** qu'il est possible de capter à plusieurs centaines de mètres.
- Travaux de Tanaka et al. **[7,8]** en 2004 et 2007 qui ont montré qu'il est possible d'intercepter à plus de 80 % les fontes TEMPEST de Kuhn et Anderson (utilisation de filtres gaussiens optimisés). Ils ont eux-mêmes proposé des voies d'amélioration des fontes TEMPEST.

Si ces approches concernent des techniques fondées sur la physique, leur intérêt tient au fait qu'elles sont réalisables par un « simple » code malveillant, et ce, sans aucune difficulté.

Toutes ces techniques reposent de manière systématique sur le concept de canal caché, concept que nous employons également.

Rappelons tout d'abord ce qu'est un canal caché. Il existe de nombreuses définitions, plus ou moins équivalentes. Nous reprenons celle du DoD US qui date de 1985 :

« Canal de communication B empruntant une part de la capacité (bande passante) d'un canal de communication A légitime ».

Ce canal B permet donc de transmettre de l'information sans la permission du propriétaire légitime du canal A, lequel n'est même pas conscient de son existence. Les données transmises sont généralement les siennes. Il est essentiel de garder à l'esprit que ces canaux existent toujours : il suffit soit de savoir où les chercher, soit de les créer soi-même. Et comme nous allons le montrer, un ordinateur contient un nombre incalculable de tels canaux (en fait pratiquement tout dans un ordinateur le permet, même un canal chiffré IPsec **[9]**). Nous en présentons deux familles :

- détourner des ressources logicielles natives en utilisant des techniques proches de la stéganographie, mais fondée sur des techniques de traitement du signal (« *Windows jingle attack* ») ;
- détourner l'activité matérielle de périphériques : mise en/hors tension d'un écran (« *visual covert channel* »), bruit d'un disque dur ou d'un ventilateur (« *audio covert channel* »)...

2. Attaque via le jingle Windows

2.1 Principe général

Cette attaque a été développée au laboratoire en 2008 avec Vincent Calmette-Vallet et Stéphane de Royer-Dupré. Elle reprend dans l'esprit les techniques de stéganographie fondées non pas sur des techniques de mathématiques discrètes exploitant un format numérique, mais sur des techniques analogiques du traitement du signal. Le principe en est simple : le code malveillant récupère le login/mot de passe et va les dissimuler dans le jingle sonore émis par Windows lors de l'ouverture de session (ou tout autre événement de ce type). Les données sont ensuite récupérées soit via un microphone dissimulé dans la pièce ou une pièce attenante (un téléphone mobile dissimulé dans un faux plafond et déclenchable à distance fait l'affaire) ou bien, à une distance plus grande au moyen d'un micro très haute sensibilité. Même captées à distance plus ou moins grande, l'expérience montre que les techniques de traitement du signal sont suffisamment performantes pour extraire les données ainsi dissimulées.

Les outils utilisés pour mettre en œuvre une telle attaque sont relativement simples et génériques (hormis le code malveillant lui-même que nous ne détaillerons pas) : un logiciel de type SCILAB pour le traitement du signal [N4], un logiciel d'acquisition audio (Audacity dans notre cas) et d'un microphone.

Les données secrètes que nous souhaitons faire sortir sont codées sur 8 bits par caractère, soit b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0. Si, pour des besoins didactiques, nous considérerons des données non chiffrées, il est bien évident que le virus lui appliquera d'abord la compression puis le chiffrement de ces données. Chaque bit est représenté (codé) par une fréquence spécifique. Si le ième bit du code ASCII du caractère ($0 \leq i \leq 7$) est égal à 1, alors la fréquence correspondante est émise, et ne sera pas émise dans le cas où le bit vaut 0.

Le jingle d'ouverture de session de Windows XP est utilisé pour dissimuler l'émission des fréquences sonores. Ce jingle dure environ cinq secondes. Une contrainte importante imposée par ce scénario d'attaque réside dans la durée d'émission d'une fréquence de codage qui doit être suffisamment longue pour être traitée dans la phase d'analyse et de décodage. Optimalement, nous utilisons une durée de 136 ms suite à nos différents tests : cette durée de base est celle qui permet la plus grande discrétion. Ainsi, une séquence de 20 caractères (136 ms/caractère) peut être dissimulée dans les cinq secondes du jingle.

Prenons par exemple la chaîne de vingt caractères suivante « login/mot de passe » et fixons arbitrairement 8 fréquences code f_0, f_1, f_2, f_3, f_4, f_5, f_6 et f_7. Le caractère « l » (en binaire 01001100) sera codé par l'émission simultanée des fréquences f_2, f_3 et f_6 pendant 136 ms, le caractère « o » (en binaire 0110111) sera codé par l'émission simultanée des fréquences f_0, f_1, f_2, f_3, f_5 et f_6 pendant 136 ms... et ainsi de suite pour l'ensemble des caractères. Au total, nous avons un temps d'émission total de 20*136 ms, soit 2,8 secondes ; ce qui est parfaitement dissimulé dans la durée totale des cinq secondes du jingle.

Notons que le choix des fréquences de codage constitue une quantité secrète (qui peut également varier de caractère en caractère – application par exemple de permutations) connue du seul virus et de l'attaquant.

Le son ainsi émis précédemment est enregistré pour être ensuite analysé. On effectue tout d'abord la transformée de Fourier générale du signal enregistré auquel on applique ensuite des filtres centrés autour des différentes fréquences de codage. On applique alors à chacun de ces résultats filtrés une transformée de Fourier inverse cette fois-ci pour retrouver la présence ou non dans le temps de la fréquence de codage.

2.1.1 Codage des informations secrètes à transmettre

2.1.1.1 Numérisation du jingle de WINDOWS

Lorsque l'on procède à la numérisation d'un fichier « wav » avec le logiciel *Scilab*, une fréquence d'échantillonnage F de 22050 Hz est imposée (dans le cas général, avec une bibliothèque en C, on peut considérer toute autre fréquence). De cette fréquence et de la durée du jingle découle un nombre de points d'échantillonnage total de 106150 pour pouvoir décrire le jingle. En conséquence, lorsque le logiciel Scilab procède à la numérisation du jingle, il crée une matrice de 2 lignes et de 106150 colonnes décrivant cette mélodie. Les 2 lignes proviennent du caractère stéréographique du signal. Dans toute la suite des expérimentations,

nous décidons de réduire le jingle à un unique vecteur de 106150 valeurs (première ligne de la matrice). À l'écoute, pour la plupart des oreilles, la différence sonore entre ce signal réduit et le signal réel (matrice complète) n'est pas perceptible.

2.1.1.2 Détermination des fréquences de codage

La figure 1 représente la transformée de Fourier générale du jingle sous le logiciel SCILAB. Elle visualise l'ensemble des fréquences composant ce jingle. L'étude approfondie de ce graphe doit nous permettre de choisir les huit fréquences de codage des données.

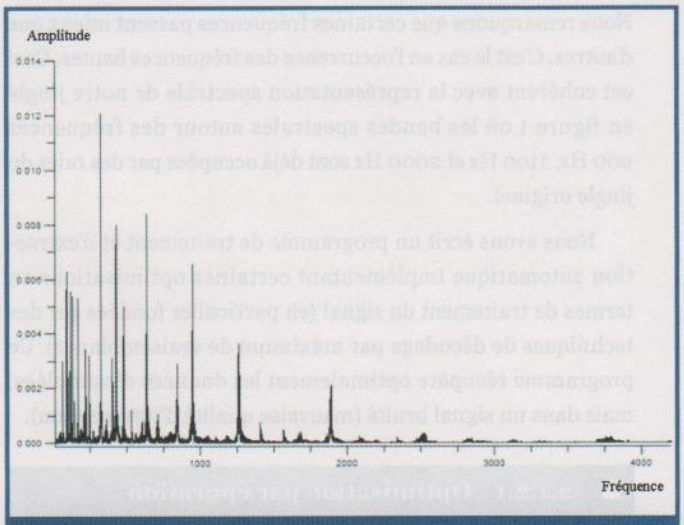


Figure 1 : Spectre fréquentiel du jingle

Comme le logiciel SCILAB impose une fréquence d'échantillonnage fixe de 22500Hz et que le théorème d'échantillonnage de Shannon [10] stipule que la fréquence d'échantillonnage doit être au minimum égale à deux fois la valeur de la fréquence maximale du spectre numérisé, une limite supérieure théorique à 11250 Hz pour le choix des fréquences de codage est imposée. D'autre part, au vu de la figure, nous observons que le choix de fréquences trop aiguës (f > 2000 Hz) représente un risque pour la discrétion du codage (absence naturelle de raies spectrales du jingle dans cette plage). À ces deux premières considérations s'ajoute également la bande passante des haut-parleurs de l'ordinateur cible et du micro qui nous imposent de travailler dans une gamme correcte (ni trop basse, ni trop haute) afin d'avoir le meilleur rendu possible des sons.

Il faut donc trouver un compromis dans le choix des fréquences entre :

- d'une part, une plage de fréquences vierges permettant de discriminer nos fréquences de codage avec les différentes raies du spectre du jingle ;
- d'autre part, une place proche d'une des raies du spectre du jingle afin de « noyer » les sons du codage dans ceux du jingle.

Notons que lorsque les fréquences choisies sont trop basses, les haut-parleurs bourdonnent à l'émission et les sons sont inexploitables. De manière optimale, les fréquences suivantes ont été considérées : 900Hz, 1100 Hz, 1350 Hz, 1500 Hz, 2000 Hz, 2250 Hz, 2700 Hz et 3000 Hz. Elles autorisent une dissimulation des données optimales notamment en termes de furtivité.

2.1.1.3 Création de la matrice des fréquences de codages du secret

Le logiciel SCILAB représente un son de fréquence f dans le domaine temporel sous la forme d'un vecteur à n colonnes. Le ième terme du vecteur s'écrit sous la forme :

$$K\cos(2\pi f.Pas.i)$$

où Pas = 1/F, K est un coefficient d'amplification et F est la fréquence d'échantillonnage.

On utilise cette propriété afin de créer une matrice appelée « signal » qui contient l'ensemble des sons à ajouter au jingle. Cette matrice possède 8 lignes (correspondant aux 8 fréquences utilisées pour coder chaque caractère) et 106150 colonnes (correspondant à la durée du jingle comme nous l'avons vu précédemment). En fait, cette matrice est la concaténation de x sous-matrices, x étant le nombre de caractères à transmettre. Chaque sous-matrice comporte 8 lignes et 3000 colonnes. Ce nombre de colonnes correspond à la durée pendant laquelle les fréquences relatives à un caractère vont être émises ou non (3000/F soit environ 136 ms). La création d'une sous-matrice de la matrice signal s'obtient en effectuant une multiplication terme à terme (opérateur $\otimes$) de deux matrices appelées « M_c » et « M_f ». « M_c » possède 8 lignes correspondant à la traduction binaire sur un octet d'un caractère ASCII et 3000 colonnes toutes identiques. « M_f », de même taille que « M_c » est une représentation SCILAB des fréquences ajoutées entre les échantillons (n-1)*3000 et n*3000, n décrivant le nième caractère à transmettre. Un exemple de création de sous-matrice est détaillé à la figure 2, page suivante.

On obtient ainsi x sous-matrices qui servent chacune à décrire un son émis sur 3000 points d'échantillonnage. La matrice signal est constituée de ces x sous-matrices et complétée avec des zéros pour être de même longueur que le vecteur jingle réduit.

1 1 1		cos (2π.f1. pas. 1) cos (2π.f1. pas. 2) ... cos (2π.f1. pas. 3000)
0 0 0		cos (2π.f2. pas. 1) cos (2π.f2. pas. 2)...cos (2π.f2. pas. 3000)
1 1 1		...
0 0 0	⊗	
1 1 1		
0 0 0		
1 1 1		...
1 1 1		cos (2π.f8. pas. 1) cos (2π.f8. pas. 2) ... cos (2π.f8. pas. 3000)

Figure 2 : Exemple : Création de la première sous-matrice de la matrice « signal »

À chaque terme « i » du vecteur jingle sont alors additionnées toutes les lignes de la colonne « i » de notre matrice signal. Nous obtenons ainsi un nouveau vecteur constituant le « jingle codé » qui n'est rien d'autre que la numérisation temporelle du jingle original auquel sont ajoutées, toutes les 136ms, huit fréquences sonores au maximum représentant le codage ascii d'un caractère composant les données à faire fuir du poste. Pour masquer optimalement cet ajout de fréquences, il convient d'utiliser un coefficient d'amplification K le plus faible possible. La lecture par le logiciel SCILAB du jingle codé permet d'apprécier subjectivement la répercussion sonore de la valeur de K sur la qualité finale du jingle. Toute la difficulté est de trouver le compromis entre une valeur de K suffisamment faible pour être imperceptible à l'oreille tout en étant suffisamment prononcée pour permettre de retrouver le secret lors de la phase d'analyse du signal.

2.1.2 Analyse du son et récupération des informations secrètes

Dans notre scénario d'attaque, un micro situé à proximité de l'ordinateur cible enregistre le son émis. Il nous faut donc réaliser un programme spécifique pour analyser ce son. Pour cela, nous utilisons une nouvelle fois le logiciel SCILAB qui permet de passer facilement du domaine spectral au domaine temporel par les outils des transformées de Fourier. L'objectif étant de faire une analyse spectrale du son et d'y appliquer des filtres pour retrouver les 8 fréquences ajoutées : on retrouve ainsi les codes ASCII correspondant aux caractères du couple login/mot de passe.

Dans un premier temps, nous validons notre approche en considérant le cas le plus défavorable : le secret est la chaîne « ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ » (bits tous à 1) nécessitant d'émettre toutes les fréquences simultanément. Nous créons donc 8 filtres « passe-bande » centrés sur les 8 fréquences de codage que nous appliquons successivement à la transformée de Fourier générale du jingle codé. Ensuite, à chaque résultat intermédiaire, nous appliquons une transformée de Fourier inverse afin de retrouver dans le domaine temporel, c'est-à-dire au cours de la lecture du jingle codé, la présence ou non de la fréquence de codage. La présence de ladite fréquence traduira la présence d'un bit égal à 1 ; son absence traduira la présence d'un bit égal à 0.

La figure 3 représente un exemple de courbes obtenues avec notre programme pour extraire le secret « ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ » avec un coefficient K de 0,01. Le caractère « ÿ » est représenté en binaire par 11111111 et le caractère « espace » est représenté en binaire par 00000100. La chaîne « ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ » doit théoriquement se présenter graphiquement pour les fréquences 0, 1, 2, 3, 4, 5, 6 et 7 par une succession de créneaux et de vides et pour la 6ème fréquence par une émission continue. Cette chaîne test permet de vérifier la pertinence des paramètres choisis pour l'expérimentation.

La lecture seule des graphes permet aisément de reconstituer le code binaire de chacun des caractères de notre secret. Nous remarquons que certaines fréquences passent mieux que d'autres. C'est le cas en l'occurrence des fréquences hautes. Ceci est cohérent avec la représentation spectrale de notre jingle en figure 1 où les bandes spectrales autour des fréquences 900 Hz, 1100 Hz et 2000 Hz sont déjà occupées par des raies du jingle originel.

Nous avons écrit un programme de traitement et d'extraction automatique implémentant certaines optimisations en termes de traitement du signal (en particulier fondées sur des techniques de décodage par maximum de vraisemblance). Ce programme récupère optimalement les données dissimulées, mais dans un signal bruité (mauvaise qualité d'interception).

2.1.2.1 Optimisation par épuration fréquentielle

Nous avons mis au point une technique permettant de discriminer encore mieux les fréquences de codage lors de l'analyse consiste à réaliser un « trou » dans la représentation spectrale du jingle afin d'y apposer nos fréquences de codage dans une zone vierge de toutes autres fréquences parasites. La figure 4 illustre l'expérience réalisée dans les mêmes conditions que précédemment (figure 3) à la différence que des trous de 60 Hz ont été réalisés dans le spectre du jingle original centrés sur les emplacements des fréquences de codage. Nous pouvons comparer cette figure avec celle du paragraphe précédent et apprécier ainsi la différence de qualité qu'apporte cette épuration fréquentielle. D'un point de vue sonore, cette technique d'épuration fréquentielle reste imperceptible. Notons que cette technique permet d'insérer une fréquence de codage dans une plage de fréquences encombrée du jingle afin de gagner en discrétion.

2.1.2.2 Validation opérationnelle

Pour jouer l'attaque en conditions réelle, nous avons utilisé des micro-espions classiques (micro Logitech, enregistreur numérique VoiceTracer 620, téléphone mobile en écoute). Nous avons ensuite considéré un secret réel constitué de la chaîne

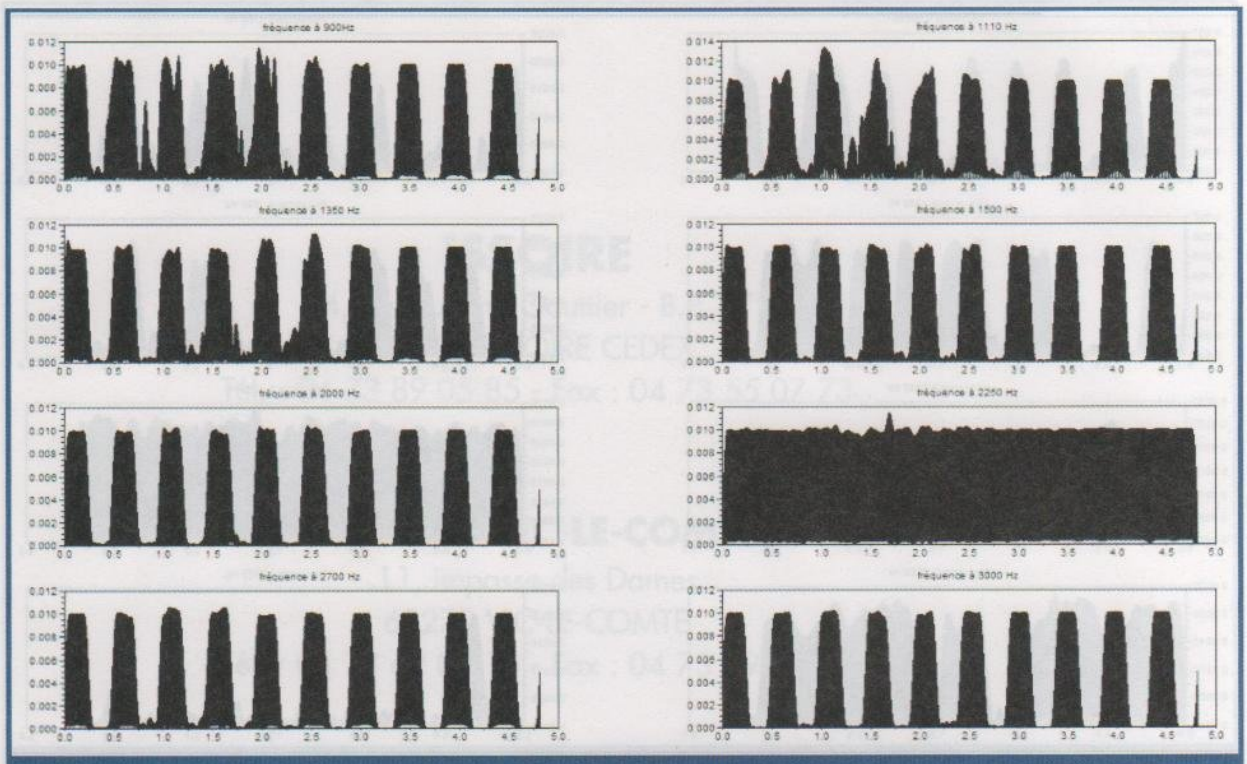


Figure 3 : Extraction théorique du secret « ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ » avec un coefficient K de 0.01

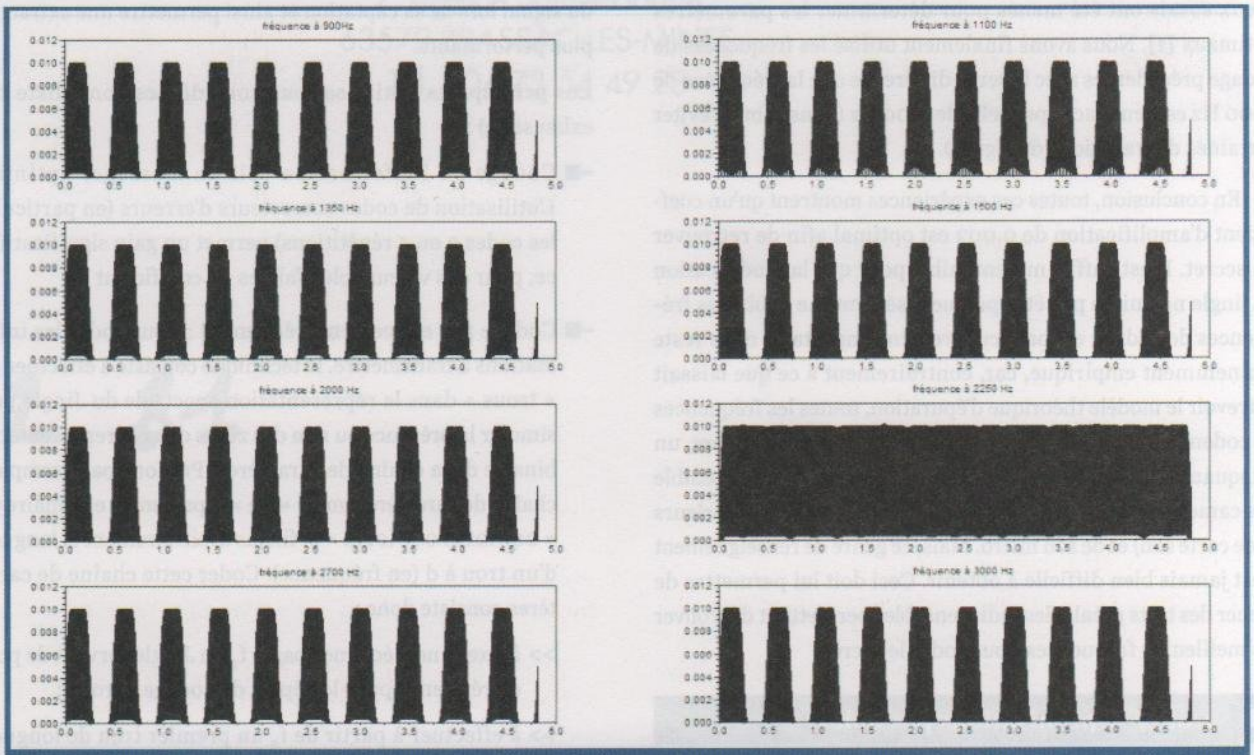


Figure 4 : Extraction théorique du secret « ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ » avec un coefficient K de 0.01 et avec la méthode d'épuration fréquentielle

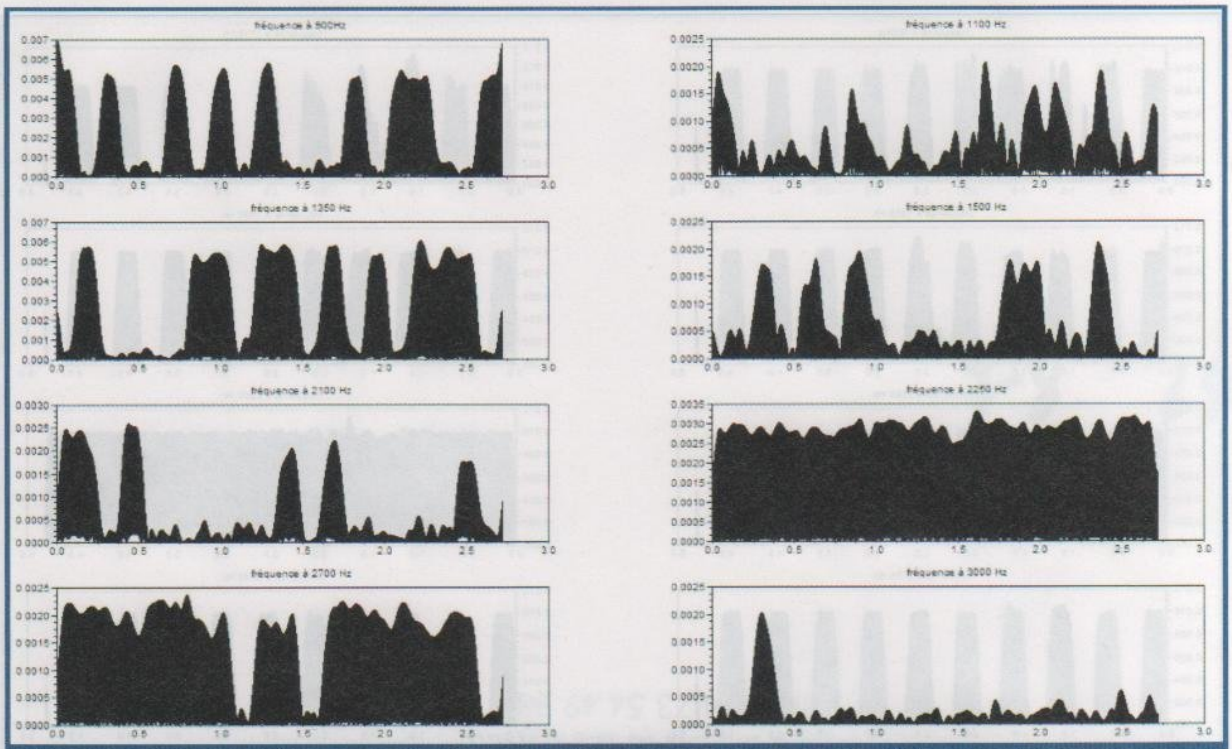


Figure 5 : Troisième expérience : extraction du secret « stéphane et vincent ! » avec un coefficient d'amplification K=0.003 et avec le remplacement de la fréquence à 2000 Hz par 2100 Hz. Le secret extrait est « stéphane et vincent ! ».

de caractères suivante : « stéphane et vincent ! ». De nombreux essais ont été menés pour déterminer les paramètres optimaux [1]. Nous avons finalement utilisé les fréquences de codage précédentes avec la seule différence que la fréquence de 2000 Hz est remplacée par celle de 2100 Hz (dans le but d'éviter certaines dégradations du signal).

En conclusion, toutes ces expériences montrent qu'un coefficient d'amplification de 0.003 est optimal afin de retrouver un secret. Il est suffisamment faible pour que la modification du jingle ne puisse pas être perçue aisément. Le choix des fréquences de codage est particulièrement important, mais reste éminemment empirique, car, contrairement à ce que laissait entrevoir le modèle théorique d'épuration, toutes les fréquences ne codent pas l'information avec la même qualité. Dès lors, un attaquant doit avoir une connaissance la plus précise possible des caractéristiques de l'ordinateur cible (type de haut-parleurs et de carte son) et de son micro. Mais, ce genre de renseignement n'est jamais bien difficile à obtenir. Ceci doit lui permettre de mener des tests préalables indispensables permettant de trouver les meilleures fréquences pour coder le secret.

2.1.2.3 Optimisations possibles

Certaines optimisations ont été mises au point et testées (ou sont en cours de validation finale). Elles permettent à la fois de rendre l'insertion des données inaudible, même pour une oreille exercée et de résister à la déformation du signal lors de sa captation et ainsi permettre une extraction plus performante.

Les principales optimisations considérées sont (liste non exhaustive) :

- Codage de l'information (bit de message) optimisé. L'utilisation de codes correcteurs d'erreurs (en particulier les codes 3 ou 5 répétitions) permet un gain significatif, et ce, pour des valeurs plus faibles du coefficient K.
- Codage par effacement fréquentiel : pour coder les informations à transmettre, la technique consiste à effectuer des « trous » dans la représentation spectrale du Jingle pour simuler la présence ou non des zéros dans la représentation binaire de la chaîne de caractères. Prenons par exemple la chaîne de caractère simple « hh » représentée en binaire par « 00010110 00010110 » et fixons arbitrairement la longueur d'un trou à d (en fréquence). Coder cette chaîne de caractères consiste donc :
  - >> à fixer une fréquence basse f_B du Jingle servant de point de référence pour le départ du codage à trous ;
  - >> à effectuer à partir de f_B un premier trou de longueur 3*d, puis de laisser les fréquences inchangées sur une longueur d, puis de refaire un autre trou d'une longueur d, et ainsi de suite pour coder l'ensemble de notre chaîne

de caractères. Les différentes expérimentations montrent que les modifications sonores sont inaudibles. Le décodage se fait en considérant des techniques de décodage classiques pour un canal à effacement techniques de modulation de phase : cela consiste à effectuer une modulation de la phase de certaines fréquences présentes dans le jingle. Par exemple, on code un bit à 0 avec un déphasage de $\pi/2$ tandis que l'on code un bit à 1 avec un déphasage de π. Cette technique est totalement imperceptible pour l'oreille humaine quelle que soit l'amplitude sonore des fréquences déphasées. En outre, en ne se cantonnant pas à deux modulations de phase, on peut augmenter considérablement le débit d'informations à extraire. Ainsi, avec une modulation de phase à 4 moments, on peut transmettre en une fois 2 bits simultanément.

Conclusion

L'attaque via le jingle Windows est facilement transposable à n'importe quel système d'exploitation et à n'importe quelle ressource nativement présente dans la machine cible. L'interception du signal sonore peut se faire à une distance relativement importante qui est en fait déterminée par la qualité du micro utilisé pour capter le signal. Un service spécialisé dispose de micros extrêmement sensibles **[N5]**. Les capacités en termes de traitement du son sont également un facteur déterminant.

Une autre piste de travail considérée actuellement consiste à émettre les données à faire sortir dans une gamme de fréquences inaudibles : infra-sons ou ultra-sons. Les difficultés techniques sont nombreuses, en particulier car les cartes audio actuelles peuvent avoir des capacités limitées (notamment au-delà des 25 KHz). Mais, la piste semble néanmoins prometteuse. À suivre...

3. Attaque par manipulation de périphériques

Nous exploitons maintenant le fait que beaucoup de périphériques soit se comportent différemment selon l'activité du système d'exploitation et/ou du processeur, soit peuvent être manipulés logiciellement pour faire varier leur comportement :

- bruit du disque dur lors des accès en lecture/écriture ;
- vitesse et donc bruit du ventilateur ;
- mise en veille, extinction, activation du moniteur ;
- mise en veille/redémarrage du système d'exploitation ;
- diodes LED de cartes réseau, de webcam, de lecteur de CD-ROM ;
- consommation électrique de courant ;
- émission de chaleur **[N6]** ;
- ouverture/fermeture du lecteur de CD-ROM ;
- ...

Un virus peut donc très facilement faire varier l'activité d'un périphérique afin de coder les données à faire sortir de la machine cible. L'interception du signal résultant (sonore, thermique, lumineux) peut alors se faire à très longue distance, en particulier la nuit **[N6]**.

Le principe général de ce type d'attaque repose sur plusieurs étapes :

- Le virus code les informations dérobées en hexadécimal (après une compression et un chiffrement éventuels).
- Le virus établit – selon une convention secrète déterminée par l'attaquant et régulièrement modifiée par le virus – une correspondance biunivoque entre les caractères hexadécimaux et des intervalles de temps.

0	1	2	3	4	5	6	7
t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
8	9	A	B	C	D	E	F
t_8	t_9	t_A	t_B	t_C	t_D	t_E	t_F

Ces intervalles de temps représentant l'intervalle entre deux changements d'activité du périphérique utilisé pour faire évader les données du poste cible. Ils doivent en particulier être choisis très soigneusement pour tenir compte de la résilience physique des périphériques (par exemple, un écran ne se rallume ou s'éteint pas immédiatement). Mais, des techniques de codes correcteurs d'erreurs peuvent être utilisées pour rendre l'attaque encore plus efficace.

Nous montrons maintenant quelques-unes des attaques que nous avons testées.

4. Attaque du type « Visual covert channel »

La première attaque est tellement simple qu'elle en est presque triviale. Il suffit d'utiliser un écran (cela marche mieux de nuit, mais, le jour, dans un bureau entre 12 : 00 et 13 : 00 quand la personne est partie déjeuner, cela marche aussi très bien) et de lui faire faire une sorte de morse lumineux (selon le codage précédent) en alternant les fonctions *power off* et *power on*. Ainsi, la séquence hexadécimale 0xCAFE sera codée de la manière suivante :

Power on – attendre t_C secondes – power off – attendre t_A secondes – power on – attendre t_F secondes – power off – attendre t_E secondes – power on – ...

Dans le cas d'un écran, nous avons utilisé les valeurs t_i permettant de faire sortir une séquence de 20 octets en moins d'une heure. Le débit est certes lent, mais il est possible de faire mieux par des techniques de codage optimisées.

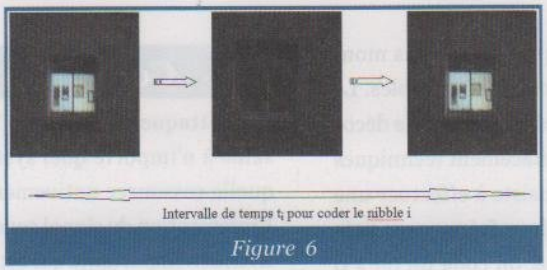


Figure 6

La figure 6 illustre le principe (photos prises lors des tests).

Pour réaliser ce type d'attaque, l'utilisation de certaines fonctions de l'API Windows est probablement l'approche la plus facile (ce n'est pas la seule), et essentiellement via la fonction `SendMessage()` **[11]**.

Il existe de nombreuses variantes possibles et de nombreuses fonctions pour les réaliser (par exemple fonction de redémarrage via la fonction `ExitWindowsEx` ; il faut juste bien veiller à ce que le virus fonctionne en modes persistant et résident).

L'usage de la commande `NirCmd` permet également un nombre incroyable de détournements de périphériques. Essayez par exemple de coder une séquence de 20 octets en faisant, selon le codage précédent, ouvrir/fermer le lecteur de CD-ROM ! C'est d'une simplicité déroutante. Le tout est de soigneusement planifier ces actions pour ne pas alerter l'utilisateur.

5. Attaque du type « Audio covert channel »

Le principe reste le même, mais considère le domaine sonore. Selon l'activité du processeur ou de l'OS, les actions menées, les périphériques font plus ou moins de bruit. Il est également possible de piloter ces périphériques de sorte à faire varier l'intensité sonore de leur activité, selon le même principe général décrit plus haut. Nous allons considérer deux types de périphériques : un disque dur classique et un ventilateur (général ou d'une carte vidéo).

5.1 Activité du disque dur

Nous avons exploité dans ce cas le fait qu'un accès en écriture ou lecture de manière intense se traduit par une augmentation du bruit du disque dur (série de clics ou craquements). Nous avons remarqué lors des expériences que ce phénomène était plus flagrant s'agissant d'une partition non système.

Le virus écrit ou lit aléatoirement des données sur le disque dur à des intervalles de temps variables pour coder les données à faire évader. Le bruit sera d'autant plus important que les distances entre les positions des *clusters* concernés seront grandes. Enfin, plus que pour un écran, la résilience d'un disque dur est plus importante.

Le lecteur pourra analyser le fichier audio réalisé montrant une différence de niveau sonore lors d'un accès en écriture **[1]**.

5.2 Activité de ventilateur

L'idée cette fois est de faire varier la vitesse d'un ventilateur (processeur ou carte graphique) de sorte à coder les données volées. L'utilisation de processus provoquant une élévation de température et donc conséquemment une variation de la vitesse du ventilateur concerné, reste assez difficile à mettre en œuvre : elle n'est pas systématique sauf dans le cas de calculs assez longs et lourds (donc le taux de transmission s'en trouver d'autant réduit), et va dépendre de plus de la température de la pièce dans laquelle se trouve l'ordinateur cible.

Il est plus intéressant donc de profiter dans certains cas des ressources permettant de piloter directement la vitesse d'un ventilateur au moyen de commandes natives au système (ou contenues dans le virus). Il est essentiel de signaler que les primitives concernées ne sont pas universellement portables et vont dépendre de telle ou telle famille de plateforme.

Citons quelques moyens pour le faire :

- Utilisation de la primitive `SetSpeed` (`CIM_FAN Class`) sous Windows (mais n'étant pas nativement implémentée dans WMI, il est nécessaire de concevoir son propre conteneur).
- Utilisation de bibliothèques dédiées, par exemple `thinkpad-acpi` sous Linux (machines IBM type *thinkpad*). Une fois le module chargé (`rmmod thinkpad_acpi && modprobe thinkpad_acpi`) - les primitives suivantes font varier la vitesse du ventilateur (voir **[12]** pour plus détails et d'autres techniques similaires) :

```
# echo level 0 > /proc/acpi/ibm/fan (fan off)
# echo level 2 > /proc/acpi/ibm/fan (low speed)
# echo level 4 > /proc/acpi/ibm/fan (medium speed)
# echo level 7 > /proc/acpi/ibm/fan (maximum speed)
# echo level auto > /proc/acpi/ibm/fan (automatic - default)
# echo level disengaged > /proc/acpi/ibm/fan (disengaged)
```

Il existe de telles bibliothèques un peu partout et pratiquement pour tous les matériels (si on ne veut pas écrire soi-même ses propres outils). Par exemple, pour les cartes NVidia, l'utilitaire *nvclock* **[13]** permet de faire varier la vitesse du ventilateur des cartes de ce constructeur.

La commande

```
$ nvclock --force --fanspeed xx
```

fixe la vitesse de rotation du ventilateur (la valeur `xx` est proportionnelle à cette vitesse, mais il est nécessaire de faire des essais pour trouver les bonnes valeurs en fonction de la carte vidéo cible).

Conclusion

Dans cet article, nous avons montré comment créer des canaux cachés permettant de manière purement logicielle (via un code malveillant) de provoquer une fuite d'information. Ces techniques sont encore plus efficaces que l'exploitation de signaux parasites compromettants. Cela montre une fois de plus que, d'une part, le fait d'isoler un ordinateur d'un réseau n'est qu'une condition nécessaire mais nullement suffisante et que, comme le précisent les textes officiels (IGI 900), la sécurité d'un système d'information repose en grande partie sur la sécurité informatique au même niveau que la lutte contre les signaux parasites compromettants. Cela doit nous rappeler, en particulier, que l'usage de la cryptologie (chiffrement des données) n'est pas une protection en soi, mais une composante parmi d'autres.

Mais ce type de techniques doit aussi nous faire réfléchir sur les normes et matériels que nous utilisons, lesquels sont fabriqués à l'étranger donc hors de notre souveraineté. À une époque où la rétro-ingénierie matérielle devient compliquée sur certains équipements, comment s'assurer que les périphériques que nous utilisons ne contiennent pas des fonctionnalités cachées favorisant de telles approches. C'est précisément ce que fait la Chine – par ailleurs grande pourvoyeuse de matériels informatiques – en concevant ses propres normes et matériels. Enfin, l'évolution des technologies doit être suivie en permanence, car chacune d'entre elles apporte son lot de possibilités ou de faiblesses en termes de rayonnements TEMPEST naturels ou provoqués : ainsi à ce jour, l'absence de risque de technologies comme le courant porteur en ligne (CPL) est loin d'avoir été établie : de quoi occuper de nombreux chercheurs ! ■

Notes

[N1] Il est important de rappeler la définition officielle de ce qu'est un système d'information (IGI 900) : « tout dispositif fonctionnant à l'électricité traitant de l'information (de sa création à sa destruction) ».

[N2] Je ne peux m'empêcher de partager avec le lecteur ce souvenir savoureux d'une réponse d'un de mes élèves qui, dans une copie, présentait, la « cage à farfadet » comme un des moyens de lutte contre les SPC.

[N3] Le lecteur se doit toutefois de réfléchir au fait que parmi ces aspects figurent : la possibilité d'introduire des trapdoors matérielles (processeurs, cartes mères, chipsets...), la difficulté voire l'impossibilité en pratique de pratiquer de la rétro-ingénierie matérielle et, surtout, le modelage des esprits par l'hégémonie de certaines normes. À ce jour, seule la norme AMSG, d'origine américaine est prise en compte. Or, il est clair que certains partis pris techniques ne sont pas pris en compte par cette norme (par exemple, la manière dont les cartes mères sont câblées) : volontairement ou non ? Au lecteur de se faire une idée !

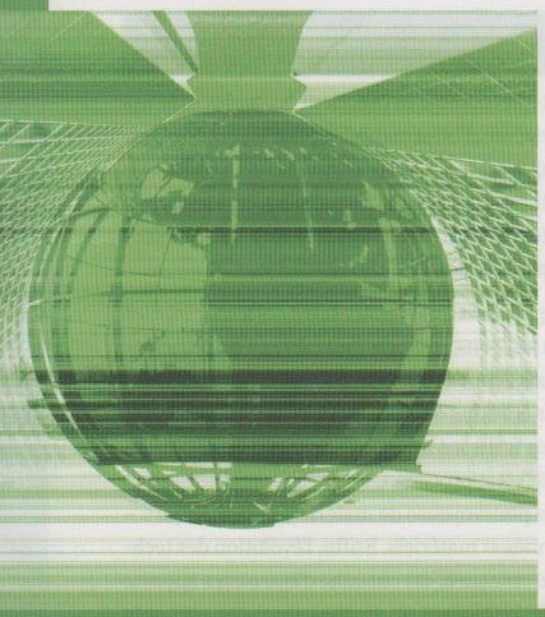
[N4] Dans le cas d'une implémentation au sein du virus, il existe des bibliothèques performantes permettant de faire du traitement du signal, l'usage de SCILAB servant juste pour la phase initiale de validation et l'aspect didactique de cet article.

[N5] Les services secrets israéliens, en particulier, ont mis au point des micros ultra-sensibles reposant sur les fibres optiques qui peuvent retransmettre des signaux sonores lorsque cette fibre est frappée par un signal laser. Une autre technique de captation de signaux sonores extrêmement faibles dans une pièce pourvue de fenêtres consiste à frapper perpendiculairement les vitres d'une de ces fenêtres d'un mince faisceau laser et à traiter mathématiquement le faisceau réfléchi, lequel contient des informations fortement corrélées au signal sonore cible.

[N6] Nous avons observé que l'activité du processeur s'accompagnait directement d'une élévation de température significative qui lui était fortement corrélée. Les techniques que nous décrivons ici sont directement transposables et l'interception avec une caméra thermique de bonne qualité peut se faire actuellement à plusieurs centaines de mètres. Ce genre de techniques peut également être transposable à des signaux lumineux nocturnes faibles en utilisant des intensificateurs de lumière (amplification d'un facteur 1000 de la lumière résiduelle). Lorsque l'on considère le nombre d'ordinateurs qui restent actifs la nuit dans les entreprises ou les bureaux, on imagine sans peine le terrain de jeu disponible pour les attaquants.

Références

Les références de cet article sont disponibles sur miscmag.com/ref44.



François Ropert
fropert@packetfault.org

ATTAQUE DES NUMÉROS DE SÉQUENCES CRYPTOGRAPHIQUES SUR OSPF

mots-clés : *réseaux / vulnérabilité / OSPF / exploitation*

Il existe plusieurs scénarii d'attaques sur le protocole de routage OSPF. Les défauts de conception, l'injection de messages OSPF par un tiers, la consommation excessive de ressources et les attaques collatérales impactant le fonctionnement du protocole. La vulnérabilité décrite dans cet article utilise un défaut de conception ainsi qu'un défaut dans l'implémentation logicielle. Si cette dernière peut être corrigée, la première ne l'est pas. Et c'est ainsi que commença le jeu du chat et de la souris où le dernier qui riposte vaincra sur le réseau. La vulnérabilité se situant en amont de la vérification du condensat MD5, toute résistance via le moyen cryptographique est futile.

1. Historique des failles OSPF

OSPF (*Open Shortest Path First*) est un protocole de routage interne à état de liens. Il existe deux familles de protocoles : l'interne et l'externe. Le premier permet d'acheminer les paquets le plus rapidement possible par le chemin le plus court. Les protocoles RIP, EIGRP et OSPF entrent dans cette catégorie. Quant au routage externe, il se distingue par sa capacité à choisir un chemin pour lequel une politique de flux a été définie et donc pas forcément le plus court. L'on peut citer le protocole BGP. OSPF utilise l'état des liens pour créer son arbre de routage. Si un lien tombe ou devient disponible, la topologie est recalculée. Aujourd'hui, il est possible de distinguer deux versions du protocole OSPF. La version 2 décrite dans la RFC2328 [1] et la version 3 décrite dans la RFC5340 [2]. La première concerne IPv4 et la seconde IPv6. L'on peut noter qu'on ne peut employer OSPFv3 pour faire du routage IPv4. Au fil des années, plusieurs vulnérabilités notables liées au protocole OSPF sont apparues.

Le premier exploit pouvant être cité est l'OoopSPF de Felix Lindner. C'est un dépassement de tampon concernant la gestion des voisins lorsque plus de 255 entrées sont ajoutées. Ensuite est apparu Nemesis, un outil écrit en 2000 qui exécute des attaques génériques sur les protocoles de routage.

Le module OSPF permet, notamment, l'injection de paquets de type *link state advertisement* manipulant la table topologique OSPF. Cette dernière est utilisée par un routeur pour remplir sa table de routage globale. Cependant, si, pour une destination donnée, OSPF n'est pas sélectionné pour indiquer le chemin le plus court, alors l'attaque aura l'effet d'un coup d'épée dans l'eau. L'outil permet aussi d'interagir avec l'élection du DR (*Designated Router*) et BDR (*Backup Designated Router*) en environnement commuté. Le DR est un routeur qui se charge de récupérer les mises à jour topologiques déclenchées par un routeur OSPF et de pousser les mises à jour vers les autres routeurs appartenant au même segment Ethernet. Le BDR prend la relève au cas où le DR tombe. Pour ce type de manipulation, il est plus intéressant de placer un rogue routeur octroyant plus de contrôle à l'attaquant. Toutefois, la finalité de cette dernière attaque est peu intéressante. D'autres vulnérabilités touchent l'implémentation logicielle OSPF dans l'IOS et ont été trouvées « par hasard » chez des clients de Cisco. Ceci arrive relativement fréquemment dans des environnements très spécifiques aussi bien en configuration logicielle que matérielle menant à des comportements atypiques comme celui décrit dans le bulletin cisco-sa-20080326-queue [3].

Dans la majorité des cas précités, excepté la dernière faille qui reste unique en son genre, le *palliatif* le plus efficace est l'activation de l'authentification par MD5 pour les envois de paquets OSPF. Si le condensat (auquel aucun *salt* n'est appliqué) est récupéré avec un analyseur de trames, il peut ensuite être attaqué « offline » avec un dictionnaire ou par brute force.

L'attaque découverte et décrite par l'auteur de cet article impacte deux classes de vulnérabilité : une attaque triviale liée à la conception de la version 2 du protocole OSPF et un problème d'implémentation logicielle qui a été reproduit sous forme de « *proof of concept* » sur du matériel embarquant les logiciels Cisco IOS et Cisco ASA. Le tableau 1 décrit un récapitulatif de l'actualité sécurité d'OSPF.

Événement	Description
RFC OSPF version 2 (1998)	Draft OSPF version 2 en 1997 et ratifié en avril 1998
Nemesis (2000)	Outil d'injection de paquets pour protocoles réseau
OoopSPF (2002)	Exploitation d'un *heap overflow* dans OSPF et exécution de shellcode
IETF RPSEC OSPF (2006)	Draft IETF sur la sécurité du protocole OSPF
OspfAsh (2007)	Outil d'écoute et d'injection de paquets OSPF
MPLS VPN OSPF DoS (2008)	Déni de services dans le traitement de l'option OSPF *sham-link* en environnement MPLS VPN sous réserve d'utilisation d'une carte SUP32 ou SUP720
Manipulation des numéros de séquences (2008)	Attaque décrite dans cet article. Mise à jour des RFC par le groupe IETF RPSEC

Tableau 1

2. Vulnérabilité de conception

Afin de comprendre où se situe l'attaque, le tableau 2 décrit les étapes de l'établissement d'une session OSPF entre deux routeurs sur un réseau Ethernet.

Il y a un changement d'état lorsque deux équipements s'échangent certaines données au travers du protocole OSPF (numéro 89). Il y a un en-tête OSPF générique à tout échange. Celui-ci contient, notamment, la version du protocole OSPF utilisée, le router-id qui est un numéro identifiant le routeur au sein d'OSPF sous la forme d'une adresse IP et un numéro de séquence d'authentification correspondant généralement à une authentification de type MD5 si celle-ci a été activée. À la suite de cet en-tête, se trouvent les informations qui seront traitées localement et immédiatement par le processus OSPF de chaque routeur dès lors qu'une nouvelle connexion est créée comme les voisins actifs (impact sur les états DOWN, INIT, 2WAY) ou les mises à jour topologiques (EXSTART, EXCHANGE, LOADING et FULL).

Etat	Description
DOWN	Aucun message HELLO n'a été envoyé
INIT	Un message HELLO a été envoyé, mais le routeur n'a pas encore accusé réception
2WAY	La communication est bidirectionnelle
EXSTART	Établissement de sessions entre les routeurs et le DR/BDR
EXCHANGE	Échange des DBD (*DataBase Description*). Un DBD contient les en-têtes correspondant à chaque paquet LSA qui vont être envoyés lors de la prochaine étape
LOADING	Envoi des paquets LSA de mise à jour topologique
FULL	Les deux routeurs sont synchronisés. Les tables topologiques sur chacun des routeurs sont identiques.

Tableau 2

Ci-après, un exemple d'en-tête OSPF pour lequel l'authentification MD5 est présente.

```
No.   Time    Source        Destination      Protocol Info
   1 0.000000  192.168.0.100     224.0.0.5       OSPF   Hello Packet

Frame 1 (98 bytes on wire, 98 bytes captured)
Ethernet II, Src: Cisco_26:3f:20 (00:b0:64:26:3f:20), Dst: IPv4mcast_00:00:05
(01:00:5e:00:00:05)
Internet Protocol, Src: 192.168.0.100 (192.168.0.100), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
  OSPF Header
    OSPF Version: 2
    Message Type: Hello Packet (1)
    Packet Length: 48
    Source OSPF Router: 192.168.0.100 (192.168.0.100)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0x0000 (none)
    Auth Type: Cryptographic
    Auth Key ID: 1
    Auth Data Length: 16
    Auth Crypto Sequence Number: 0x2b9542ad
    Auth Data: 038473959C37C62A7B60D1128212B81E
```

Bien que le champ Auth Data soit positionné juste après le numéro de séquence par la majorité des analyseurs de trame dans le but de faciliter la lecture du paquet par un humain, en réalité, celui-ci est ajouté à la fin du paquet après avoir calculé le hash MD5 de la clé pré-partagée concaténée au contenu du paquet OSPF comprenant l'en-tête (donc le numéro de séquence) et la charge utile (messages HELLO, LSA et DBD). L'on voit bien ce comportement en lisant le code disponible dans le projet libre Quagga :

```
1. /* Generate a digest for the entire packet + our secret key. */
2. md5_init_ctx (&ctx);
3. md5_process_bytes (ibuf, ntohs (ospfh->length), &ctx);
4. md5_process_bytes (auth_key, OSPF_AUTH_MD5_SIZE, &ctx);
5. md5_finish_ctx (&ctx, digest);

6. /* Append md5 digest to the end of the stream. */
7. oldputp = stream_get_putp (op->s);
8. stream_set_putp (op->s, ntohs (ospfh->length));
9. stream_put (op->s, digest, OSPF_AUTH_MD5_SIZE);
10. stream_set_putp (op->s, oldputp);

11. /* We do *NOT* increment the OSPF header length. */
12. op->length = ntohs (ospfh->length) + OSPF_AUTH_MD5_SIZE;
```

Le champ Message Type nous indique que l'en-tête suivant qui n'est pas affiché dans la capture est un paquet HELLO. Ce type de message est transmis en permanence à intervalle régulier via *multicast* ou en point à point.

Le numéro de séquence d'authentification qui est en fait un *timestamp* UNIX est positionné ici à 0x2b9542ad et incrémenté de 10 (0xA) entre chaque émission de message HELLO. La valeur d'intervalle de temps entre chaque HELLO se configure sur chaque routeur et doit être obligatoirement identique pour qu'une relation OSPF s'établisse entre deux routeurs. Ceci afin d'éviter des inconsistances dans la table de voisinage OSPF.

Un paquet OSPF doit remplir une condition pour être accepté : le numéro de séquence doit être supérieur au numéro de séquence précédent. Voici comment est géré le numéro de séquence par le logiciel libre Quagga :

```
1. nbr = ospf_nbr_lookup_by_routerid (oi->nbrs, &ospfh->router_id);
2. if (nbr && ntohl(nbr->crypt_seqnum) > ntohl(ospfh->u.crypt.
   crypt_seqnum))
3.  {
4.    zlog_warn ("interface %s: ospf_check_md5 bad sequence %d
      (expect %d)",
5.                    IF_NAME (oi),
6.                    ntohl(ospfh->u.crypt.crypt_seqnum),
7.                    ntohl(nbr->crypt_seqnum));
8.    return 0;
9.  }
10.// Le Digest MD5 est comparé ici (code non affiché)
11. if (nbr)
12.   nbr->crypt_seqnum = ospfh->u.crypt.crypt_seqnum;
13. return 1;
```

En lisant la ligne 2 du code, il est identifié qu'un équipement réseau prendra logiquement en compte un paquet dont le numéro de séquence est supérieur à la séquence actuelle pour un router-id particulier.

L'auteur de l'article va maintenant exploiter la connaissance de cette faiblesse théorique sur les équipements de type Cisco IOS et Cisco ASA.

3. Vulnérabilité d'implémentation

En ayant connaissance du code précédent, lorsqu'un paquet OSPF est réceptionné, le test du numéro de séquence échouera si le numéro de séquence est identique. Or, certaines versions IOS ainsi que toutes les versions de l'ASA ne rejettent pas cette condition. L'auteur n'a pas accès au code source Cisco, mais il est possible, au regard de tests, de déduire que, dans le code de l'IOS, est utilisé un signe supérieur ou égal au lieu d'un signe supérieur. Premier problème, le paquet sera accepté si l'on effectue un simple rejeu de paquet. Le second problème est la fenêtre d'attaque. En effet, dans les spécifications d'OSPF, un message HELLO est envoyé toutes les 10 secondes sur le segment Ethernet en considérant un comportement par défaut. Il serait donc possible de rejouer un paquet seulement pendant les dix prochaines secondes. Or, ce n'est pas le comportement observé sur l'IOS et l'ASA. Un paquet rejoué avec le même numéro de séquence pendant plus de dix secondes n'est pas invalidé, car le routeur utilise sa propre horloge pour déduire le prochain numéro de séquence et non la valeur du dernier numéro de séquence envoyée par la machine attaquante. Donc, tant que l'attaque continue et que le timestamp générant le numéro de

séquence du routeur n'est pas dépassé, celui-ci a le contrôle sur la table topologique OSPF du routeur attaqué et identifié par son router-id.

Pour la démonstration, l'auteur a choisi de supprimer toutes les routes d'un routeur voisin reçues par l'équipement cible.

Un paquet HELLO est généré par la machine attaquante avec comme adresse source l'équipement qui n'est pas la cible de l'attaque. Dans ce paquet, le champ Active Neighbor recense les routeurs voisins OSPF. L'attaquant fait croire au routeur cible qui va recevoir le paquet par le Multicast que son voisin n'a pas détecté sa présence. L'équipement cible ne se voyant pas lui-même dans les paquets émis par ses routeurs voisins, cela signifie que ces derniers ne voient pas le routeur cible. Alors, le routeur cible modifie l'état de la session OSPF avec son voisin, car il interprète que la communication n'est pas bidirectionnelle. En relisant le tableau 2, la connexion passe de l'état 2WAY à INIT. Les routes précédemment propagées via les LSA par le routeur victime ne le sont plus dans cet état. Si, pour une route annoncée, OSPF était le protocole sélectionné chez les voisins du routeur cible et que c'était le seul chemin connu (la destination n'est pas annoncée par d'autres routeurs sur le réseau) pour cette route, alors la table de routage sera mise à jour et la route ne sera plus annoncée.

De plus, comme dit précédemment, le numéro de séquence utilisé dans l'en-tête OSPF est un *timestamp* UNIX. Aussi, celui-ci a une dépendance totale envers l'horloge du routeur. Ainsi, la modification de celle-ci a pour conséquence le contrôle complet du processus de vérification du numéro de séquence par l'attaquant. Si un routeur n'a jamais été mis à l'heure manuellement (via la commande set clock) ou automatiquement par NTP (qui n'a jamais rencontré un équipement Cisco avec l'horloge en 1993...) et si celui-ci est redémarré, alors le timestamp sera positionné dans le passé. Concrètement, il suffit pour un pirate de capturer un paquet OSPF de type HELLO ne contenant pas le routeur victime dans les Active Neighbor. Par exemple, une capture est réalisée à 0x00004242. L'équipement cible est redémarré. Le compteur de numéro de séquence est réinitialisé à 0x00000000. L'attaquant possède maintenant une fenêtre d'attaque de presque 5 heures. Le premier paquet de l'attaque est rejouable dans cette période de temps et sera valide indéfiniment au bon vouloir du pirate. Il est aussi important de noter que lorsque le numéro de séquence dépasse 0xFFFFFFFF, celui-ci est réinitialisé à 0x00000000. La conséquence du passage du numéro de séquence à 0 est que pendant quelques secondes (l'intervalle de temps correspondant à l'émission de quatre messages HELLO), les routeurs n'accepteront plus les paquets LSA. Il faut alors attendre que chaque routeur réinitialise la séquence de découverte du routeur voisin OSPF.

L'auteur de l'article a développé un code « *proof en concept* » en Python mettant en pratique les informations vues précédemment. Voici l'état de la table de voisinage avant attaque :

```
192.168.0.101#show ip ospf neighbor
Neighbor ID Pri State Dead Time Address Interface
192.168.0.100 1 FULL/DROTHER 00:00:31 192.168.0.100 Ethernet0
192.168.0.1 1 FULL/DR 00:00:34 192.168.0.1 Ethernet0
```

Le mot-clé full indique que l'équipement 192.168.0.101 a échangé ses routes avec 192.168.0.100 et 192.168.0.1. Le Neighbor ID correspond au router-ID. Le Dead Time indique en secondes le temps avant de considérer un routeur voisin comme non participatif à l'arbre OSPF. Lorsque le compteur est décrémenté à 0 ou quatre fois l'intervalle de temps d'envoi d'un paquet HELLO, le routeur concerné est retiré de la table de voisinage. En condition normale d'utilisation, et, avec une configuration par défaut, un paquet HELLO est envoyé toutes les 10 secondes. Ce qui fait que le compteur ne descend pas sous les 30 secondes. Lorsque l'attaque est lancée, le résultat de la commande show ip ospf neighbor est la suivante :

```
192.168.0.101#show ip ospf neighbor
Neighbor ID Pri State Dead Time Address Interface
192.168.0.100 1 INIT/DROTHER 00:00:39 192.168.0.100 Ethernet0
192.168.0.1 1 FULL/DR 00:00:35 192.168.0.1 Ethernet0
```

L'état du voisin 192.168.0.100 est plongé dans l'état INIT. La table topologique est recalculée par le routeur 192.168.0.101 et supprime les routes apprises par l'équipement 192.168.0.100. L'impact CPU et mémoire sur l'équipement est très faible pendant l'attaque. Un show tech donne peu d'indices. À moins de sniffer les paquets entrant sur l'équipement, il est très difficile d'effectuer un premier diagnostic sur un routeur en proie à cette attaque. Si la vulnérabilité semble être très localisée, ce n'est que la partie émergée de l'iceberg.

4. Impact sur le réseau et palliatifs

OSPF est un protocole de routage interne, il peut être découpé en aires. Chaque aire est une feuille de l'arbre OSPF. De par sa conception, le blocage d'un routeur de type *Area Border Router* (ABR) coupera l'accès à une aire par le reste des autres aires OSPF. L'aire 0 est l'aire de *backbone*. Lorsqu'une aire veut communiquer avec une autre aire, la communication doit passer par l'aire 0 en utilisant un ou plusieurs *virtual-links* jusqu'à ce que l'ABR ayant une connexion dans l'aire 0 soit joignable. Si toutes les connexions vers l'aire 0 sont bloquées, c'est tout le réseau interne de l'entreprise qui est bloqué. Exception faite lorsqu'un flux est initié depuis une aire de transit vers une aire qui lui est directement raccordée et donc n'a pas besoin de

transiter par l'aire de *backbone*. La fonctionnalité *OSPF area transit capability*, laquelle est activée par défaut sur les routeurs Cisco, permet à deux aires non-backbone de communiquer directement sans remonter par l'aire 0. L'attaque peut aussi avoir un effet collatéral néfaste si BGP est utilisé. En effet, ce dernier peut employer OSPF dans sa partie iBGP (c'est-à-dire dans le réseau interne) pour établir une communication. Si le seul moyen de joindre un voisin iBGP est OSPF, alors les sessions BGP seront coupées et votre réseau ne sera plus utilisé comme aire de transit. Pour reprendre l'attaque précédente, la topologie du réseau de test est représentée par la figure 1 et, ensuite, l'état de la table de routage avant attaque.

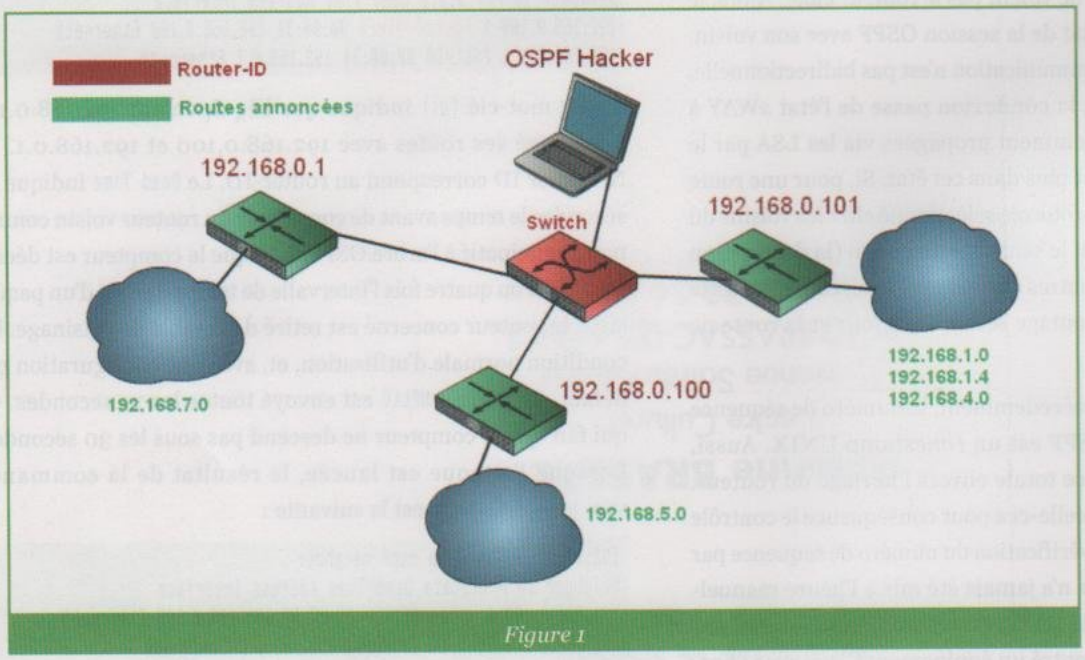


Figure 1

```
192.168.0.100#show ip route
     192.168.4.0/32 is subnetted, 1 subnets
O       192.168.4.1 [110/11] via 192.168.0.101, 00:01:48, Ethernet0/0
C    192.168.5.0/24 is directly connected, Loopback5
C    192.168.0.0/24 is directly connected, Ethernet0/0
     192.168.1.0/32 is subnetted, 2 subnets
O       192.168.1.1 [110/11] via 192.168.0.101, 00:01:48, Ethernet0/0
O       192.168.1.5 [110/11] via 192.168.0.101, 00:01:48, Ethernet0/0
     192.168.7.0/32 is subnetted, 1 subnets
O       192.168.7.1 [110/11] via 192.168.0.1, 00:01:45, Ethernet0/0
```

Les entrées débutant par O indiquent une route apprise par OSPF. Ci-après, le contenu de la table de routage global lorsque l'attaque est lancée. Les routes apprises via le voisin 192.168.0.101 sont supprimées :

```
192.168.0.100#show ip route
C    192.168.5.0/24 is directly connected, Loopback5
C    192.168.0.0/24 is directly connected, Ethernet0/0
     192.168.7.0/32 is subnetted, 1 subnets
O       192.168.7.1 [110/11] via 192.168.0.1, 00:01:45, Ethernet0/0
```

Concernant le problème de rejeu de paquet, un correctif est disponible dans la prochaine version du pare-feu ASA et est déjà disponible en version INTERIM (Bêta). Le bug est référencé sous l'identifiant CSCsv21224 [4], mais nécessite un accès Cisco CCO pour être consulté.

Le palliatif classique est l'utilisation d'infrastructure ACL. Pour rappel, l'attaque peut être menée avec ou sans protection cryptographique, que ce soit MD5 ou SHA. Néanmoins, l'auteur recommande de paramétrer un roulement automatique des clés pré-partagées et d'utiliser différentes clés dans les différentes aires de l'arbre OSPF lorsque la protection cryptographique est déployée. L'application du roulement des clés aura pour effet l'invalidation temporaire de l'attaque jusqu'à ce que l'attaquant détecte le changement de clé et réadapte l'attaque. Une autre méthode pour invalider temporairement l'attaque consiste à changer le router-id du routeur attaqué. Le but de cette opération est d'invalider au niveau OSPF la relation de voisinage identifiée par le router-id (Colonne neighbor-ID retournée par la commande `show ip ospf neighbor` dont un exemple est décrit précédemment dans cet article). En modifiant le router-id du routeur, une nouvelle entrée se créera dans la table de voisinage alors que l'ancienne entrée est toujours attaquée, mais ignorée par les deux routeurs incriminés. D'autres bonnes pratiques existent pour réduire la surface d'attaque et sont facilement déployables. Par exemple, la désactivation d'OSPF sur les liens non OSPF. Annoncer le strict nécessaire sur le réseau et éviter ainsi de rendre joignable les blocs de *loopback*, DMZ ou d'infrastructure. Pour pallier une erreur humaine, un agrégat de sous-réseaux identifiés peuvent être routés en bordure vers nullo afin d'éviter les fuites. Pour d'autres idées de sécurisation de routeurs Cisco, une carte heuristique des meilleures pratiques est disponible sur le blog [5] de l'auteur.

Conclusion

Suite à la réalisation d'une preuve de concept concernant les vulnérabilités décrites dans cet article, le groupe RPSEC de l'IETF a mis à jour le document « *Issues with existing Cryptographic Protection Methods for Routing Protocols* » et le brouillon « *OSPF HMAC-SHA Cryptographic Authentication* », pour un usage futur et conjoint de SHA et OSPF.

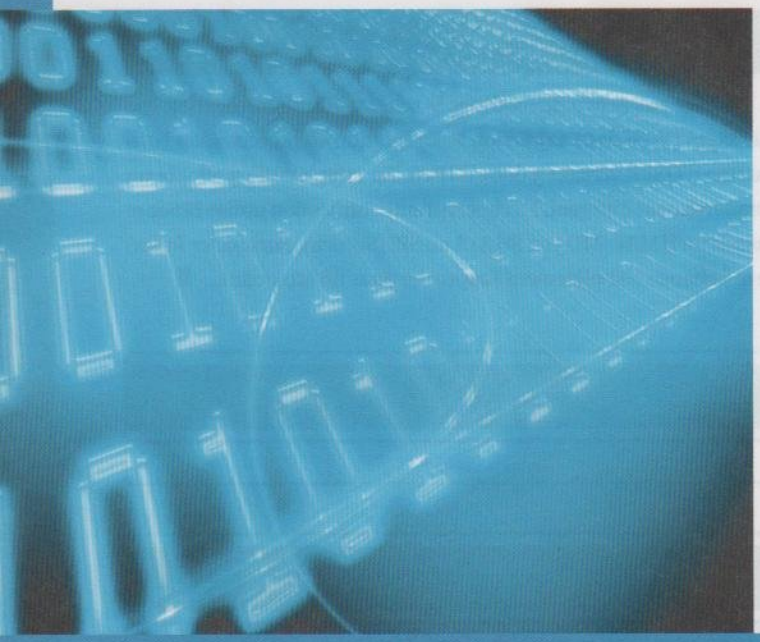
Il n'est pas prévu de modifier le schéma d'authentification des messages OSPF dans sa version 2. La solution au problème est l'utilisation d'OSPF version 3 donc d'IPv6. En effet, OSPFv3 adopte un comportement différent. La couche d'authentification utilise IPSec et associe un HMAC-MD5 ou SHA1 à un SPI (*Security Parameter Index*) ce qui change complètement l'approche pour un attaquant. ■

Remerciements

Merci à Cédric Foll pour la relecture ainsi qu'au PSIRT Cisco.

Liens

[1] http://www.ietf.org/rfc/rfc2328.txt
[2] http://www.ietf.org/rfc/rfc5340.txt
[3] http://www.cisco.com/warp/public/707/cisco-sa-20080326-queue.shtml
[4] http://tools.cisco.com/Support/BugToolKit/search/getBugDetails.do?method=fetchBugDetails&bugId=CSCsv21224
[5] http://blog.packetfault.org/archives/2008/11/23/carte_heuristique_securite_cisco_ios/index.html



Rémi Gacogne – rgacogne@coredump.fr
Vincent Rasneur – vrasneur@free.fr

DES ÉCHANGES SOAP PROPRES ET SANS BAVURE : SIGNATURE ET CHIFFREMENT

mots-clés : *Web Services / SOAP / WS-Security / savon / signature / chiffrement*

Cet article constitue une suite au dossier sur les web services du numéro précédent. Nous l'avons vu, de plus en plus d'entreprises se tournent vers la mise en œuvre de web services, aussi bien pour la fourniture de services internes que pour offrir une API publique à leurs partenaires et clients. Il devient donc indispensable de se pencher sur la sécurisation de ce qui devient chaque jour un outil un peu plus central et critique.

Reprenons les bases de ce qui fait la sécurité d'un système : l'intégrité, la confidentialité et enfin la disponibilité. Les deux premiers points vont essentiellement concerner les échanges de messages entre le client et le serveur de Web Services, alors que le dernier s'applique surtout à la protection du serveur.

Tous les exemples de code fournis dans cette partie sont écrits en langage C et utilisent les bibliothèques `libxml2` **[LIBXML2]** et `xmlsec` **[XMLSEC]**. Les exemples XML sont conformes à SOAP 1.2 **[SOAP12]** et WS-Security 1.1 **[WSS]**. La norme WS-Security permet d'inclure les clés dans les messages SOAP. Néanmoins, pour ne pas compliquer excessivement les exemples, nous considérerons dans cet article que les clés sont déjà connues a priori des différents acteurs.

Les lecteurs sont invités à se reporter à l'article WS-Security du numéro précédent pour des précisions sur les aspects théoriques de cette norme.

1. Signature

Nous allons nous concentrer sur l'intégrité des données transitant entre le client et le serveur, de manière à s'assurer que ces données n'ont été ni altérées ni modifiées, intentionnellement ou non. Pour cela, nous allons nous appuyer dans un premier temps sur la norme XML Signature, qui permet la signature numérique de tout ou partie d'un arbre XML. Nous verrons ensuite son utilisation dans la norme WS-Security. Le système de signature mis en œuvre repose sur un principe de cryptographie asymétrique et présente l'avantage de garantir la non-répudiation du message.

1.1 XML Signature

Le fonctionnement de XML Signature [XMLDSIGN] est relativement simple. Il faut tout d'abord décider quel nœud de l'arbre XML nous souhaitons signer, puis les algorithmes à utiliser. Nous devons ensuite décider si nous allons utiliser une signature enveloppante, enveloppée ou détachée. Une signature enveloppante contient en elle-même les données qu'elle signe. Une signature enveloppée est, à l'opposé, contenue dans ces données. Enfin, une signature détachée se trouve à l'extérieur de l'arbre XML qu'elle signe. D'un point de vue pratique, il est recommandé de toujours indiquer explicitement quelles sont les parties du message qui doivent être signées. Cela permet notamment à des intermédiaires de modifier certaines parties du message (typiquement les en-têtes SOAP) sans rendre la signature caduque.

Nous allons voir ce fonctionnement pas à pas au travers d'un code C, que nous commenterons au fur et à mesure. Globalement, le code suivant prend une requête SOAP sous forme d'un document XML présent sur disque ainsi qu'une clé RSA, et affiche une requête modifiée contenant une signature de ce document.

La requête SOAP initiale est relativement simple :

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <PrintMe>
      Hello, World!
    </PrintMe>
  </env:Body>
</env:Envelope>
```

Le code permettant de signer la requête est le suivant.

```
static xmlNode * get_signature_template(xmlDoc * doc)
{
    xmlNode * result = NULL;
    xmlNode * sign_node = xmlSecTmplSignatureCreate(doc,
            xmlSecTransformExclC14NId, xmlSecTransformRsaSha512Id,
            NULL);
    if (sign_node != NULL)
    {
        xmlNode * ref_node = xmlSecTmplSignatureAddReference(s
        ign_node,
          xmlSecTransformSha512Id, NULL, BAD_CAST "#myBody",
          NULL);
        if(ref_node != NULL)
        {
            if(xmlSecTmplReferenceAddTransform(ref_node,
              xmlSecTransformExclC14NId) != NULL)
            {
                xmlNode * key_info_node =
                xmlSecTmplSignatureEnsureKeyInfo(sign_node,
                NULL);
                if(key_info_node != NULL)
                {
                    if(xmlSecTmplKeyInfoAddKeyName(key_info_
                    node, NULL) != NULL)
                    {
                        result = sign_node;
                    }
                }
            }
        }
        if (result == NULL)
        {
            xmlFreeNode(sign_node), sign_node = NULL;
        }
    }
    return result;
}

static xmlSecDSigCtxPtr get_sig_ctx(char const * const key_file)
{
    xmlSecDSigCtxPtr result = NULL;
    xmlSecDSigCtxPtr dsigCtx = xmlSecDSigCtxCreate(NULL);
    if(dsigCtx != NULL)
    {
        dsigCtx->signKey = xmlSecCryptoAppKeyLoad(key_file,
          xmlSecKeyDataFormatPem, NULL, NULL, NULL);
        if(dsigCtx->signKey != NULL)
        {
            if(xmlSecKeySetName(dsigCtx->signKey,
             BAD_CAST key_file) >= 0)
            {
                result = dsigCtx;
            }
        }
        if (result == NULL)
        {
            xmlSecDSigCtxDestroy(dsigCtx), dsigCtx = NULL;
        }
    }
    return result;
}

static xmlDoc * sign_file(char const * const xml_file,
      char const * const key_file)
{
    xmlDoc * result = NULL;
    if (xml_file != NULL && key_file != NULL)
    {
        xmlDoc * doc = xmlParseFile(xml_file);
        if (doc != NULL)
        {
            xmlNode * root_node = xmlDocGetRootElement(doc);
            if (root_node != NULL)
            {
                xmlNode *body_node = xpath_get_one_node(doc,
                 "/*[local-name()='Envelope']/*[local-
                 name()='Body']");
                if(body_node != NULL)
                {
                    xmlAttr * const root_node_id = add_wsu_
                    id_to_node(doc,
                     body_node, "myBody");
                    xmlNode * signature = get_signature_
                    template(doc);
                    if (root_node_id != NULL && signature !=
                    NULL)
                    {
                        xmlSecDSigCtxPtr sig_ctx = get_sig_
                        ctx(key_file);
                        if (sig_ctx != NULL)
                        {
                            xmlNode * sec_node = get_
                            security_node(doc,
                             root_node);
                            if (sec_node != NULL)
                            {
```

```
                                    xmlSetTreeDoc(signature,
                                    doc);
                                    xmlAddChild(sec_node,
                                    signature);
                                    if(xmlSecDSigCtxSign(sig_
                                    ctx, signature) >= 0)
                                    {
                                        result = doc;
                                    }
                                    xmlSecDSigCtxDestroy(sig_
                                    ctx), sig_ctx = NULL;
                                }
                            }
                        }
                    }
                }
                if (result == NULL)
                {
                    xmlFreeDoc(doc), doc = NULL;
                }
            }
        }
        return result;
}
```

Le début du code consiste essentiellement en quelques appels d'initialisation des bibliothèques `libxml2` et `xmlsec`. Ne nous y attardons pas et sautons directement à la fonction `sign_file()`, qui réalise la signature. La première tâche de cette fonction est de charger en mémoire notre requête SOAP sous la forme d'un arbre XML par l'appel à la fonction `xmlParseFile()`, puis d'en récupérer l'élément racine, nommé `Envelope` dans notre cas. Nous utilisons alors la fonction `xpath_get_one_node()` et une expression Xpath pour récupérer le nœud `Body`, auquel nous ajoutons l'identifiant `MyBody` à l'aide de la fonction `add_wsu_id_to_node()`. Cet identifiant nous permettra ensuite de retrouver ce nœud et d'indiquer qu'il est signé. Vient ensuite l'appel à la fonction `get_signature_template()` qui est chargée de la construction d'un *template* de signature, et sur laquelle nous allons nous attarder.

Un template de signature dans `xmlsec` indique tout simplement la composition de l'arborescence de la signature que nous allons ajouter dans l'arbre XML final. Premièrement, nous allons créer l'élément de plus haut niveau Signature à l'aide de la fonction `xmlSecTmplSignatureCreate()`. Cette fonction crée également un élément `SignedInfo`. Dans le cas d'une signature enveloppante, l'élément `Signature` serait également composé d'un ou plusieurs éléments `Object`. Ces derniers contiennent les données à signer. Un dernier élément est présent dans la signature : l'élément optionnel `KeyInfo` qui indique quelle clé a été utilisée pour signer le message, dans le cas où plusieurs clés sont disponibles.

Nous indiquons plusieurs paramètres à `xmlSecTmplSignatureCreate()`, le premier étant le document XML que nous allons signer, puis la méthode de canonisation qui sera appliquée à `SignedInfo`. Nous utilisons ici une canonisation exclusive (cf. partie suivante) nommée `xml-exc-c14n`. Nous précisons ensuite les algorithmes utilisés pour générer la signature proprement dite, respectivement RSA et SHA2-512 pour le chiffrement et pour le hash. Le dernier paramètre permet d'indiquer un identifiant à ajouter au nœud `Signature` ainsi créé, que nous laissons vide pour l'instant, car nous n'en avons pas l'utilité.

Le nouveau nœud `Signature` ainsi obtenu est passé ensuite en premier paramètre à la fonction `xmlSecTmplSignatureAddReference()`. Cette fonction ajoute un élément nommé `Reference` à l'élément `SignedInfo` précédemment créé. Ce nœud `Reference` indique quel nœud est signé dans le document courant, en le référençant par son identifiant passé en troisième paramètre. Nous laissons cet identifiant vide, car nous allons signer l'intégralité du document. Les paramètres suivants permettent de spécifier l'URI et le type du nœud `Reference`. Nous indiquons donc que le nœud que nous allons signer contient l'identifiant `MyBody`. Le second paramètre indique, quant à lui, l'algorithme de hash à utiliser pour générer une empreinte, mais nous y reviendrons.

L'appel à `xmlSecTmplReferenceAddTransform()` nous permet maintenant d'ajouter une méthode de transformation qui sera appliquée au nœud que nous souhaitons signer. La transformation est appliquée avant le calcul de la signature. Nous demandons tout simplement d'appliquer la méthode de canonisation vue précédemment afin de signer une version canonique de notre nœud.

Dernière étape pour finaliser notre template de signature, nous ajoutons les éléments optionnels `KeyInfo` et `KeyName`, que nous évoquions précédemment et qui permettent d'identifier la clé utilisée pour la signature.

Nous en avons fini avec cette fonction, et nous allons passer très rapidement sur la fonction `get_sig_ctx()`, qui se contente de charger une clé RSA privée (appel à `xmlSecCryptoAppKeyLoad()`) et d'y associer un identifiant. Ce sera ici le nom du fichier contenant ladite clé (appel à `xmlSecKeySetName()`).

Un appel à `xmlSetTreeDoc()` permet de vérifier que la signature est bien associée au document XML courant. Nous ajoutons alors le nœud `Signature` (qui ne contient pas encore la signature à proprement parler) dans l'arbre XML par l'appel à `xmlAddChild()`.

Nous calculons alors la signature définitive par l'appel à la fonction `xmlSecDSigCtxSign()`. Détaillons le travail effectué par cette fonction :

- Le nœud choisi est soumis aux différentes transformations définies plus tôt. Dans notre exemple, une canonisation exclusive est appliquée.
- Cette version est alors passée à travers l'algorithme de hash choisi lors de l'appel à `xmlSecTmplSignatureAddReference()` pour obtenir une empreinte de la version canonique du nœud. Cette empreinte, après un encodage implicite en base64, est ensuite intégrée (nœud `DigestValue`) dans l'élément `Reference` correspondant.

- L'élément SignedInfo ainsi mis à jour est alors canonisé par la méthode passée en second argument à la méthode xmlSecTmplSignatureCreate(), ici canonisation exclusive.
- Cet version canonique de SignedInfo est soumise à l'algorithme de signature spécifié lors de l'appel à xmlSecTmplSignatureCreate(), obtenant ainsi la valeur SignatureValue qui, après avoir été encodée implicitement en base64, est placée dans l'élément Signature.

En réalité, nous ajoutons le nœud contenant la signature avant de générer la signature, mais le nœud est mis à jour dynamiquement. Enfin, l'appel à xmlDocDump() affiche notre requête SOAP avec sa nouvelle signature.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
   200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="myBody">
    <PrintMe>
      Hello, World!
    </PrintMe>
  </env:Body>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
 exc-c14n#"/>
<SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
 more#rsa-sha512"/>
<Reference URI="#myBody">
<Transforms>
<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
<DigestValue>[...]</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>[...]</SignatureValue>
<KeyInfo>
<KeyName>privkey.pem</KeyName>
</KeyInfo>
</Signature></env:Envelope>
```

On retrouve bien la requête SOAP originale, à laquelle a été ajouté un nœud Signature. Dans ce nœud, nous voyons les informations qui avaient été ajoutées dans le template de signature, ainsi que la valeur du hash du document et la signature elle-même.

Le destinataire du message reçoit donc un arbre XML contenant à la fois les données attendues, la signature, les algorithmes et les noms des clés utilisés pour signer le document.

Celui-ci réalise alors l'opération inverse, à quelques détails près. Tout d'abord, il doit identifier dans le message un élément Signature, puis extraire la partie SignedInfo et l'algorithme de canonisation à lui appliquer. Ce dernier est précisé dans CanonicalizationMethod. Il canonise donc la partie SignedInfo, et en extrait la ou les Reference, une pour chaque objet signé. On obtient ainsi les transformations à appliquer à l'objet signé (Transforms) et l'algorithme de hash. Une fois les transformations appliquées et l'empreinte obtenue, on peut la comparer à celle reçue dans le champ DigestValue de notre référence. Si elle est identique, il ne nous reste plus qu'à vérifier que la signature est bien signée par la bonne clé privée. Sinon, le message est corrompu et doit être ignoré.

Pour finir, on détermine quel couple de clés a été utilisé par le biais du champ KeyInfo, dans le cas où plusieurs couples seraient disponibles, et on extrait l'algorithme de signature du champ SignatureMethod. On déchiffre alors la valeur présente dans SignatureValue à l'aide de la clé publique correspondant à la clé privée (référencée par KeyInfo). Le résultat doit être identique à la forme canonique de SignedInfo. Dans ce cas, nous avons validé l'intégrité et la non-répudiation du message reçu.

1.2 Vérification

Maintenant que nous avons un code valide permettant la signature d'une requête SOAP (ou d'une réponse d'ailleurs), voyons comment écrire du code permettant de vérifier cette signature. Le code suivant ressemble beaucoup au code précédent. Il prend également une requête SOAP sous forme de fichier et une clé, publique cette fois. Nous ne reviendrons pas sur les initialisations de bibliothèques qui n'ont pas changées.

```
static xmlSecDSigCtxPtr get_sig_ctx(char const * const key_file)
{
    xmlSecDSigCtxPtr result = NULL;
    xmlSecDSigCtxPtr dsigCtx = xmlSecDSigCtxCreate(NULL);
    if(dsigCtx != NULL)
    {
        dsigCtx->signKey = xmlSecCryptoAppKeyLoad(key_file,
          xmlSecKeyDataFormatPem, NULL, NULL, NULL);
        if(dsigCtx->signKey != NULL)
        {
            if(xmlSecKeySetName(dsigCtx->signKey, BAD_CAST key_file) >= 0)
            {
                result = dsigCtx;
            }
        }
    }
    return result;
}

static bool verify_file(char const * const xml_file,
    char const * const key_file)
{
    bool result = false;
    if (xml_file != NULL && key_file != NULL)
    {
        xmlDoc * doc = xmlParseFile(xml_file);
        if (doc != NULL)
        {
            xmlNode * root_node = xmlDocGetRootElement(doc);
            if (root_node != NULL)
            {
                xmlChar const * ids[] = {BAD_CAST "Id", NULL};
                xmlSecAddIDs(doc, root_node, ids);
                xmlNode * sign_node = xmlSecFindNode(root_node,
                xmlSecNodeSignature, xmlSecDSigNs);
                if(sign_node != NULL)
                {
                    xmlSecDSigCtxPtr sig_ctx = get_sig_ctx(key_file);
                    if (sig_ctx != NULL)
                    {
                        if(xmlSecDSigCtxVerify(sig_ctx, sign_node)
                        >= 0)
```

```
                        {
                            result =
                        (sig_ctx->status ==
                        xmlSecDSigStatusSucceeded);
                        }
                        xmlSecDSigCtxDestroy(sig_ctx), sig_
                        ctx = NULL;
                    }
                }
            }
            xmlFreeDoc(doc), doc = NULL;
        }
    }
    return result;
}
```

La fonction `verify_file()` prend les mêmes paramètres que la fonction `sign_file()` vue précédemment. Il récupère de la même manière le nœud principal de l'arbre XML composant notre requête SOAP. Le code se différencie ensuite du code précédent, puisqu'il cherche à localiser dans l'arbre XML un nœud contenant une signature (appel à `xmlSecFindNode()`). En revanche, la récupération du contexte signature est exactement la même que dans le code de signature, mais charge cette fois la clé publique au lieu de la clé privée. On arrive à la fin, puisque l'appel à `xmlSecDSigCtxVerify()` permet de vérifier si la signature trouvée précédemment correspond au document qui nous est présenté et à la bonne clé publique.

1.3 Signature WS-S

Voyons maintenant comment la norme WS-Security intègre XML Signature pour assurer l'intégrité et la non-répudiation des messages SOAP transmis.

Pour commencer, la norme WS-Security recommande l'ajout d'un bloc `Security` dans les en-têtes SOAP de la requête concerné. C'est dans ce bloc que doit être ajoutée la signature, dont tous les éléments `Reference` doivent se rapporter à des nœuds présents dans l'enveloppe SOAP. Ensuite, la norme précise que la canonisation utilisée doit être, dans la mesure du possible, une canonisation exclusive, et que les *tokens* de sécurité, que nous n'utiliserons pas ici, doivent être placés avant la signature pour pouvoir être lus avant d'être utilisés. Enfin, la signature utilisée ne doit être ni enveloppante ni enveloppée afin d'éviter des problèmes de sécurité, et les éléments à signer doivent être désignés explicitement.

Nous allons donc adapter notre code en fonction de ces recommandations. Si nous comparons le nouveau code avec la version précédente, nous remarquons rapidement la modification suivante dans la fonction `get_signature_template()` :

```
- xmlNode * ref_node = xmlSecTmplSignatureAddReference(sign_node,
    xmlSecTransformSha512Id, NULL, NULL, NULL);
+ xmlNode * ref_node = xmlSecTmplSignatureAddReference(sign_node,
    xmlSecTransformSha512Id, NULL, BAD_CAST "#myBody", NULL);
```

Nous voyons que, cette fois, l'identifiant du nœud à signer a bien été précisé.

Enfin, nous remarquons, juste après l'appel à `get_sig_ctx()`, un nouvel appel :

```
xmlNode * sec_node = get_security_node(doc, root_node);
```

La fonction `get_security_node()` qui n'existait pas jusqu'à présent recherche le nœud `Security` dans les *headers* SOAP, et le crée s'il n'existe pas. Nous remarquons plus loin que la signature n'est plus ajoutée directement à la racine du document, mais bien à ce nœud `Security`, comme voulu par la norme :

```
-                       xmlAddChild(root_node, signature);
+                               xmlAddChild(sec_node, signature);
```

Comme nous n'utilisons pas les tokens de sécurité et que nous avions déjà mis en place une canonisation exclusive, ces quelques changements suffisent pour être conformes avec la partie signature de la norme WS-Security.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
     wss-wssecurity-secext-1.0.xsd">
  <env:Header> <!-- L'enveloppe SOAP s'est enrichie d'un champ Header
-->
    <wsse:Security> <!-- L'en-tête spécifique à WS-Security -->
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              <!-- La méthode de canonisation appliquée à SignedInfo-->
          <SignatureMethod
              Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-
              sha512"/>
              <!-- L'algorithme de signature -->
          <Reference URI="#myRoot">
              <!-- La référence indique que c'est
              l'élément d'identifiant myRoot qui est signé -->
            <Transforms>
              <Transform
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                  <!-- La canonisation appliquée aux données -->
            </Transforms>
            <DigestMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
                <!-- L'algorithme de hash
                utilisée pour la génération de l'empreinte -->
            <DigestValue>[...]</DigestValue>
              <!-- La valeur de l'empreinte -->
          </Reference>
        </SignedInfo>
        <SignatureValue>[...]</SignatureValue>
            <!-- La valeur de la signature -->
        <KeyInfo>
          <KeyName>privkey.pem</KeyName>
            <!-- L'identification de la clé -->
        </KeyInfo>
      </Signature>
    </wsse:Security>
  </env:Header>
  <env:Body
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401\
                -wss-wssecurity-utility-1.0.xsd"
       wsu:Id="myRoot"> <!-- L'élément signé -->
    <PrintMe>
      Hello, World!
    </PrintMe>
  </env:Body>
</env:Envelope>
```

Le mécanisme de signature XML que nous venons de voir semble parfaitement adapté aux échanges SOAP, et apparaît comme une solution efficace au problème de contrôle d'intégrité que nous cherchions à résoudre. Bien que cela soit globalement vrai, il faut toutefois introduire une nuance importante. Diverses extensions SOAP existent pour étendre les possibilités de ce protocole déjà riche, apportant fatalement une complexité accrue. Parmi ces extensions, le concept de SOAP Attachements permet d'ajouter des fichiers à une requête SOAP en englobant à la fois la requête et les fichiers qui y sont joints dans un document de type MIME. Bien que les possibilités offertes par cette extension soient indéniables, le document résultant n'est plus un document XML et ne peut donc être directement signé à l'aide de XML Signature. Différentes possibilités existent pour pallier ce problème, que ce soit en ne signant que la requête SOAP elle-même à l'intérieur du document MIME ou bien en se tournant vers des solutions de signature qui ne sont pas dépendantes du format du document.

2. Canonisation

Deux arbres XML `<a attr1="1" attr2="2"><b/></a>` et `<a attr2='2' attr1='1'><b></b></a>` contiennent la même information, mais ont des structures différentes : l'ordre des attributs `"attr1"` et `"attr2"` est inversée, des guillemets simples ou doubles sont utilisés, l'élément vide `"b"` est représenté par 1 ou 2 tags. Si l'on applique un algorithme de hash sur ces 2 arbres, le résultat va être différent.

La vérification d'intégrité d'arbres XML doit donc passer par une étape intermédiaire : la canonisation. Les algorithmes de canonisation XML utilisés dans XML Signature ont été créés par le même groupe d'étude au sein du W3C.

Les algorithmes du W3C commencent à diverger lors de la gestion des *namespaces* XML. Prenons un autre exemple, un message SOAP :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <QuoteResponse xmlns="http://www.example.com/xmlns/quotes">
      <Result xsi:type="xsd:float">12.34</Result>
    </QuoteResponse>
  </env:Body>
</env:Envelope>
```

Nous avons vu que les signatures XML peuvent porter sur une partie d'un message. Si l'on ne veut signer que l'élément `Result`, un algorithme de canonisation sans gestion des namespaces peut ici entraîner une faille de sécurité : un attaquant peut changer la déclaration des namespaces `"xsi"` et `"xsd"` dans les éléments englobant `Result` sans pour autant changer le résultat de la vérification d'intégrité, et donc de la signature. Il faut ainsi récupérer les namespaces des éléments parents.

Il y a deux façons de procéder :

- la canonisation inclusive, ou tout simplement canonisation XML : depuis les nœuds ancêtres, copier toutes les déclarations et les attributs du namespace `"xml"` (`"xml:lang"` par exemple)
- la canonisation exclusive : ne copier que les déclarations de namespace visiblement utiles pour l'élément à canoniser.

Les canonisations inclusive et exclusive sont aussi divisées en deux sortes : celles qui gardent les commentaires XML (`<!-- commentaire -->`), et celles qui les enlèvent.

La canonisation inclusive est le plus souvent impossible à utiliser dans des messages SOAP : l'ajout d'un namespace (par exemple) dans l'enveloppe SOAP signifiera le rajout du même namespace dans l'élément à canoniser.

La norme WS-Security recommande donc d'utiliser la canonisation exclusive. Mais, cette canonisation pose encore un autre problème pour notre exemple. La notion de copie en fonction de la visibilité des namespaces ne permet pas de détecter tous les namespaces vraiment utilisés : `<Result xsi:type="xsd:float">12.34</Result>` le namespace dont le préfixe est `"xsd"` est stocké dans une chaîne de caractères. Il ne va pas être inclus.

L'utilisation de chaînes de caractères contenant des préfixes de namespaces est assez courante lorsqu'une expression XPath est utilisée comme valeur pour un attribut. Il ne faut donc pas considérer que ce problème n'arrive jamais.

Bien sûr, la solution existe. Un tag spécifique, nommé `inclusiveNamespaces`, a été ajouté dans la canonisation exclusive. Ce tag permet de forcer l'ajout de déclarations de namespace. L'inclusion de `"xsd"` se fait comme ceci :

```
<InclusiveNamespaces xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"
 PrefixList="xsd"/>
```

La liste des préfixes est séparée par des espaces, et peut contenir `"#default"` pour désigner le namespace par défaut.

2.1 Utilisation de la canonisation dans xmlsec

La signature d'un arbre XML utilise la canonisation à 2 endroits. Pour l'élément `SignedInfo`, l'algorithme de canonisation est indiqué lors de la création du template de signature :

```
xmlNode * sign_node = xmlSecTmplSignatureCreate(doc,
xmlSecTransformExclC14NId, xmlSecTransformRsaSha512Id, NULL);
```

Pour l'élément à signer, l'algorithme de canonisation fait partie des transformations appliquées avant de calculer son hash :

```
        xmlNode *transform_node = xmlSecTmplReferenceAddTransform
(ref_node, xmlSecTransformExclC14NId);
```

Si besoin est, un tag `inclusiveNamespaces` est rajouté. Nous le rajoutons dans l'élément `Transform` que nous venons de créer pour la canonisation [1] :

```
                if(xmlSecTmplTransformAddC14NInclNamespaces
(transform_node, BAD_CAST "xsd") == 0)
```

Essai rapide des fonctions de canonisation

Les fonctions de canonisation XML font partie de `libxml2`. Elles devraient être intégrées uniquement à `xmlsec`, mais ont été décrétées trop généralistes pour ne servir que pour la signature XML.

Vous pouvez tester directement les fonctions de canonisation inclusives et exclusives via l'outil `xmllint` (fourni avec `libxml2`). La syntaxe est :

- `xmllint --c14n document.xml` pour la canonisation inclusive ;
- `xmllint --exc-c14n document.xml` pour la canonisation exclusive.

Cependant, `xmllint` ne permet de canoniser que des documents complets et en gardant les commentaires.

3. Chiffrement

Pour parler de la confidentialité des données transitant via des services web, nous allons nous intéresser à une autre norme du W3C : XML encryption [XMLENC].

3.1 XML Encryption

Les problématiques de base restent les mêmes que pour XML signature :

- Le chiffrement est surtout applicable aux données XML (même si rien n'interdit de chiffrer une vidéo avec XML encryption).
- Granularité très fine : chiffrement sur une partie d'un document XML.
- Inclusion des métadonnées : algorithmes de chiffrement, moyens de localisation de la clé de chiffrement.

L'utilisation de XML encryption par WS-Security sera ensuite détaillée.

Le but de XML encryption est de chiffrer un arbre XML ou une partie de cet arbre. Le résultat de ce chiffrement est un élément `EncryptedData` qui remplacera les données à chiffrer dans l'arbre. Cet élément contient bien sûr également d'autres données : notamment l'algorithme de chiffrement et des informations sur la clé (nom de la clé ou propriétaire par exemple).

De la même manière que pour la signature XML, nous allons voir le fonctionnement en détail du chiffrement via un exemple en langage C. Il prend une requête SOAP sous forme de document XML présent sur le disque et affiche une requête modifiée dont un élément est chiffré. La clé AES est également stockée en local pour cet exemple.

```
static xmlNode * get_encryption_template(void)
{
    xmlNode * result = NULL;
    xmlNode * enc_node = xmlSecTmplEncDataCreate(NULL,
        xmlSecTransformAes256CbcId, NULL, xmlSecTypeEncElement,
        NULL, NULL);
    if (enc_node != NULL)
    {
        if (xmlSecTmplEncDataEnsureCipherValue(enc_node) != NULL)
        {
            xmlNode * key_info_node =
              xmlSecTmplEncDataEnsureKeyInfo(enc_node, NULL);
            if(key_info_node != NULL)
            {
                if(xmlSecTmplKeyInfoAddKeyName(key_info_node,
                NULL) != NULL)
                {
                    result = enc_node;
                }
            }
        }
        if (result == NULL)
        {
            xmlFreeNode(enc_node), enc_node = NULL;
        }
    }
    return result;
}

static xmlSecEncCtxPtr get_enc_ctx(char const * const key_file)
{
    xmlSecEncCtxPtr result = NULL;
    xmlSecEncCtxPtr enc_ctx = xmlSecEncCtxCreate(NULL);
    if(enc_ctx != NULL)
    {
        enc_ctx->encKey =
        xmlSecKeyReadBinaryFile(xmlSecKeyDataAesId, key_file);
        if(enc_ctx->encKey != NULL)
        {
            if(xmlSecKeySetName(enc_ctx->encKey,
              BAD_CAST "CN=Achille Touillette, C=FR") >= 0)
            {
                result = enc_ctx;
            }
```

```
        }
        if (result == NULL)
        {
            xmlSecEncCtxDestroy(enc_ctx), enc_ctx = NULL;
        }
    }
    return result;
}

static xmlDoc * encrypt_file(char const * const xml_file,
    char const * const key_file)
{
    xmlDoc * result = NULL;
    if (xml_file != NULL && key_file != NULL)
    {
        xmlDoc * doc = xmlParseFile(xml_file);
        if (doc != NULL)
        {
            xmlNode * root_node = xmlDocGetRootElement(doc);
            if (root_node != NULL)
            {
                xmlNode * printme_node = xpath_get_one_node(doc,
                 "/*[local-name()='Envelope']/\
                  *[local-name()='Body']/*[local-name()='PrintMe']");
                if (printme_node != NULL)
                {
                    xmlNode * template = get_encryption_template();
                    if (template != NULL)
                    {
                        xmlSecEncCtxPtr enc_ctx = get_enc_
                        ctx(key_file);
                        if (enc_ctx != NULL)
                        {
                            xmlSetTreeDoc(template, doc);
                            if (xmlSecEncCtxXmlEncrypt(enc_ctx,
                            template, printme_node) >= 0)
                            {
                                result = doc;
                            }
                            xmlSecEncCtxDestroy(enc_ctx), enc_ctx
                            = NULL;
                        }
                    }
                }
            }
            if (result == NULL)
            {
                xmlFreeDoc(doc), doc = NULL;
            }
        }
    }
    return result;
}
```

La requête SOAP initiale est la même que pour la signature. Nous allons chiffrer son élément `PrintMe`.

Nous avons les mêmes appels d'initialisation à `libxml2` et `xmlsec` au départ, suivis d'une fonction `encrypt_file()` qui réalise les mêmes actions que la signature. Le premier réel changement dans le code consiste à construire un template de chiffrement au lieu d'un template de signature, dans la fonction `get_encryption_template()`.

```
    xmlNode * enc_node = xmlSecTmplEncDataCreate(NULL,
xmlSecTransformAes256CbcId, NULL, xmlSecTypeEncElement, NULL, NULL);
```

Ce template contient l'arborescence de l'élément chiffré. L'appel à `xmlSecTmplEncDataCreate()` crée d'abord le nœud `EncryptedData` qui va remplacer l'élément à chiffrer et va ensuite le peupler.

Le premier paramètre que prend cette fonction représente le document à chiffrer. Nous n'en avons pas besoin ici. Vient ensuite l'algorithme de chiffrement proprement dit, ici AES-256 en mode CBC, ajouté dans le nœud fils `EncryptionMethod`. Les autres arguments sont les éventuels attributs à ajouter dans le nœud `EncryptedData`, ici un identifiant optionnel qui sert à localiser l'élément chiffré (à `NULL`, car nous n'en avons pas l'utilité pour l'instant), et le type de chiffrement XML que nous effectuons. `xmlSecTypeEncElement` signifie que nous chiffrons un élément XML complet. La fonction `xmlSecTmplEncDataCreate()` va aussi créer le nœud `CipherData` qui contiendra les données chiffrées.

Il reste encore des nœuds fils à créer. La fonction suivante ajoute le nœud `CipherValue`, qui contiendra les données chiffrées transformées en base64. L'ajout des nœuds `KeyInfo` et `KeyName` est effectué via l'appel aux mêmes fonctions que lors de la signature d'une requête SOAP : `KeyInfo` et `KeyName` sont définis dans le namespace de la signature XML.

La fonction `get_enc_ctx()` se contente de lire la clé symétrique AES et de lui associer un identifiant via la même fonction que pour la signature XML (c'est-à-dire `xmlSecKeySetName()`).

L'appel à `xmlSetTreeDoc()` permet, comme pour la signature, d'associer l'élément `EncryptedData` au document XML actuel. Cet élément va remplacer l'original. Nous n'avons par conséquent pas d'appel à `xmlAddChild()` comme cela était le cas auparavant.

Le chiffrement XML proprement dit est effectué par la fonction `xmlSecEncCtxXmlEncrypt()` dont le principe n'est pas très compliqué :

1. Étape 1 : L'algorithme de chiffrement est déterminé selon le choix qui a été effectué dans le template de chiffrement.
2. Étape 2 : Si le chiffrement porte sur un élément XML, le nœud à chiffrer ainsi que tous ses descendants sont lus et soumis au chiffrement défini à l'étape 1. Si seulement le contenu texte d'un élément XML est chiffré (cas que nous n'utilisons pas ici), uniquement ce contenu est lu et chiffré.
3. Étape 3 : Le résultat est codé en base 64 et est mis dans le contenu de l'élément `CipherValue`.
4. Étape 4 : Le nouveau nœud `EncryptedData` contenant toutes ses informations nécessaires est ajouté dans l'arbre. Dans le cas d'un chiffrement d'élément XML, `EncryptedData` remplace l'ancien nœud. Dans celui d'un chiffrement de contenu, l'ancien contenu est supprimé de l'arbre et remplacé par `EncryptedData`.

`xmlDocDump()` nous affiche la requête SOAP dont l'élément `PrintMe` a été chiffré :

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
          Type="http://www.w3.org/2001/04/xmlenc#Element">
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/
 xmlenc#aes256-cbc"/>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
<KeyName>CN=Achille Touillette, C=FR</KeyName>
</KeyInfo>
<CipherData>
<CipherValue>uotlf0+6ExuHWJNEUFyqlKxx4zZJoFfq70/
 aPJMKy9VraS4KOQoMqJUDHtgszwxr
6CGWgUzZqdP8vcT2cYWYTQ==</CipherValue>
</CipherData>
</EncryptedData>
  </env:Body>
</env:Envelope>
```

Nous voyons que le nœud `PrintMe` a été remplacé par `EncryptedData`. Ce nouveau nœud contient les métadonnées (algorithme de chiffrement, type de chiffrement XML, propriétaire de la clé,...) ainsi que l'ancien nœud chiffré et codé en base64 dans `CipherValue`.

3.2 Déchiffrement

L'opération de déchiffrement est facile à faire, voyons son fonctionnement grâce à un exemple de code C. Comme précédemment, il prend une requête SOAP chiffrée en entrée et une clé AES.

```
static xmlDoc * decrypt_file(char const * const xml_file,
   char const * const key_file)
{
    xmlDoc * result = NULL;
    if (xml_file != NULL && key_file != NULL)
    {
        xmlDoc * doc = xmlParseFile(xml_file);
        if (doc != NULL)
        {
            xmlNode * root_node = xmlDocGetRootElement(doc);
            if (root_node != NULL)
            {
                xmlNode * crypto_node = xmlSecFindNode(root_node,
                xmlSecNodeEncryptedData, xmlSecEncNs);
                if (crypto_node != NULL)
                {
                    xmlSecEncCtxPtr enc_ctx = get_enc_ctx(key_file);
                    if (enc_ctx != NULL)
                    {
                        xmlSetTreeDoc(crypto_node, doc);

                        if(xmlSecEncCtxDecrypt(enc_ctx, crypto_node)
                        &gt;= 0)
                        {
                            result = doc;
                        }

                        xmlSecEncCtxDestroy(enc_ctx), enc_ctx = NULL;
                    }
                }
            }
            if (result == NULL)
            {
                xmlFreeDoc(doc), doc = NULL;
            }
        }
    }
    return result;
}
```

La fonction `decrypt_file()` commence par localiser le nœud chiffré par un appel à `xmlSecFindNode()`. Ensuite, un contexte de chiffrement est créé, exactement le même que celui utilisé lors du chiffrement. La fonction `xmlSecEncCtxDecrypt()` va se charger du décodage base64 et du déchiffrement des données contenues dans `CipherValue`. Elle va ensuite tout simplement remplacer le nœud `EncryptedData` par l'ancien nœud.

3.3 Chiffrement WS-S

Pour que l'exemple ci-dessus soit conforme à WS-Security, il nous faut insérer des informations concernant la localisation des nœuds `EncryptedData`. Nous devons savoir aussi si les données chiffrées l'ont été par des clés différentes. Ces informations sont ajoutées dans le bloc `Security` de la partie en-tête d'un message SOAP. Les données chiffrées sont identifiées par la valeur de l'attribut Id de l'élément `EncryptedData` (absent de l'exemple, mais que nous allons ajouter).

Le nœud `Security` peut contenir deux éléments différents, suivant la façon dont la clé a été fournie :

- `ReferenceList` si la clé est un secret partagé entre l'émetteur du message, le destinataire et les éventuels intermédiaires.
- `EncryptedKey` si la clé est ajoutée au message. Cette clé est évidemment chiffrée aussi, cette fois-ci asymétriquement par la clé publique du destinataire. Le nœud `EncryptedKey` contient lui-même un élément `ReferenceList`.

WS-Security fait également mention de la manière d'ajouter ces 2 éléments dans le bloc `Security` : ils doivent être à chaque fois mis au début. La raison est simple : le déchiffrement doit se dérouler correctement si des données chiffrées sont de nouveau chiffrées.

Comparons les différences entre le code XML Encryption et celui prenant en charge WS-Security. Nous ne reviendrons pas sur l'ajout du bloc `Security` au message qui se fait de la même façon que pour la signature.

L'identifiant des données chiffrées est mis dans la création du template de chiffrement :

```
-    xmlNode * enc_node = xmlSecTmplEncDataCreate(NULL,
     xmlSecTransformAes256CbcId, NULL, xmlSecTypeEncElement, NULL, NULL);
+    xmlNode * enc_node = xmlSecTmplEncDataCreate(NULL,
     xmlSecTransformAes256CbcId, BAD_CAST "myPrintMe",
     xmlSecTypeEncElement, NULL, NULL);
```

Et nous utilisons l'élément `ReferenceList` pour localiser les données :

```
if(xmlSecTmplReferenceListAddDataReference(sec_node, BAD_CAST
"#myPrintMe") != NULL)
```

Ces modifications suffisent à rendre l'exemple conforme à WS-Security. Voici le message SOAP avec les changements :

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
    wss-wssecurity-secext-1.0.xsd">
```

```
<env:Header>  <!-- L'enveloppe SOAP s'est enrichie d'un champ Header -->
  <wsse:Security>  <!-- L'en-tête spécifique à WS-Security -->
    <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
          <!-- Les données chiffrées peuvent être localisées ici -->
      <DataReference URI="#myPrintMe"/>
          <!-- L'élément chiffré a l'identifiant myPrintMe -->
    </ReferenceList>
  </wsse:Security>
</env:Header>
<env:Body>
  <EncryptedData
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        Id="myPrintMe"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
         <!-- l'élément chiffré, le déchiffrement donnera
          un élément XML -->
    <EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
           <-- algorithme de chiffrement -->
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <KeyName>CN=Achille Touillette, C=FR</KeyName>
        <!-- propriétaire de la clé -->
      </KeyInfo>
      <CipherData>
        <CipherValue>xgoPbQv4AMt5Rvc5ebe2FlzjJ06XK07tv1E5JTkuh6PmAnXavaOn7
        TifyDZ/xStL
La8IrGRKBa94/9JTPLb+Eg==</CipherValue> <!-- valeur des données chiffrées -->
      </CipherData>
    </EncryptedData>
  </env:Body>
</env:Envelope>
```

Après avoir parcouru les possibilités de chiffrement d'un message SOAP, nous pouvons terminer en discutant des limitations de la norme WS-Security :

- Le concept de clé de session, générée dynamiquement et chiffrant une série de messages, ne fait pas partie de WS-Security. Le fait que le protocole SOAP soit sans état interne (*stateless*) n'est pas étranger à cette limitation. Cela oblige à inclure les clés générées dans chaque message. Une norme, WS-SecureConversation [WSC], permet de résoudre ce problème en établissant un « contexte de sécurité » s'étalant sur plusieurs messages.
- La norme WS-Security pose également des problèmes d'interopérabilité : comment s'assurer que les différents acteurs peuvent comprendre les mêmes algorithmes de chiffrement, qu'un élément défini sera toujours chiffré,... ? (Ce problème se pose également pour la signature). La réponse est dans une autre norme, WS-SecurityPolicy [WSSP], qui définit un mécanisme d'assertions liées à WS-Security.

Conclusion

Nous venons de voir deux mécanismes permettant de mettre en œuvre la protection des échanges entre un client et un serveur SOAP, couvrant ainsi les deux premiers points évoqués en introduction – la confidentialité et l'intégrité. Pour ce qui est du troisième point, il pourra faire l'objet d'un autre article consacré à la sécurité des serveurs SOAP afin d'assurer la disponibilité du service.

Les codes complets et plus robustes des différents programmes évoqués dans cet article sont présents à l'adresse suivante : **http://www.coredump.fr/static/misc/ws-security/.** ■

Note

[1] `libxml2` utilise UTF-8 comme codage interne pour ses chaînes de caractères. L'utilisation de `BAD_CAST` permet de lui indiquer que la chaîne que nous lui passons est en UTF-8.

Bibliographie & Références

KANNEGANTI (R.), CHODAVARAPU (P.), SOA Security, Manning Publications.

[LIBXML2] libxml2 : http://www.xmlsoft.org

[XMLSEC] XMLSec : http://www.aleksey.com/xmlsec/

[WSS] WS-Security : http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[XMLENC] XML Encryption : http://www.w3.org/TR/xmlenc-core/

[XMLDSIGN] XML Signature : http://www.w3.org/TR/xmldsig-core/

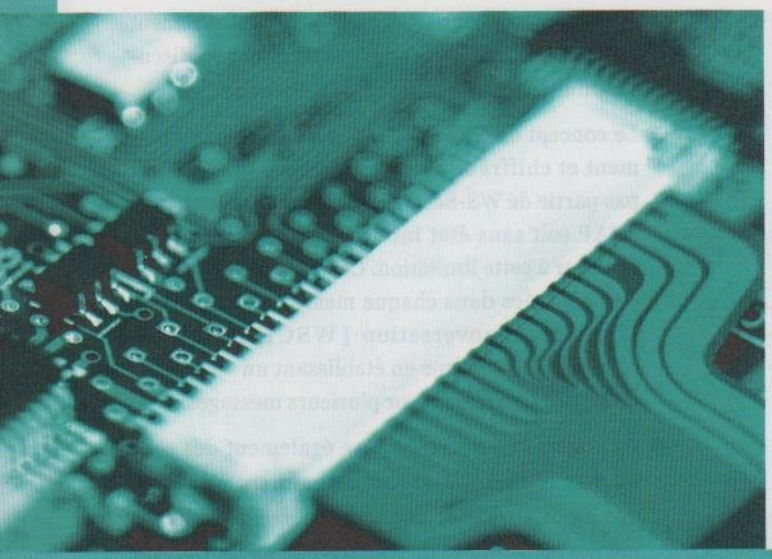
[CANONXML] Canonical XML : http://www.w3.org/TR/xml-c14n

[EXCLXML] Exclusive XML Canonicalization : http://www.w3.org/TR/xml-exc-c14n/

[WSC] WS-SecureConversation : http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx

[WSSP] WS-SecurityPolicy : http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx

[SOAP12] SOAP 1.2 : http://www.w3.org/TR/soap12-part1/



Vincent Verneuil
vincent.verneuil@laposte.net

COURBES ELLIPTIQUES ET ATTAQUES PAR CANAUX AUXILIAIRES

mots-clés : *courbes elliptiques / carte à puce / attaques / canaux auxiliaires / contre-mesures*

L'embarqué est un monde dans lequel les contraintes de puissance de calcul et d'espace mémoire sont particulièrement sensibles. Ces raisons ont conduit les acteurs de ce domaine, en particulier ceux de la carte à puce, à se tourner vers des cryptosystèmes utilisant les courbes elliptiques qui s'avèrent très avantageux en termes de rapport sécurité/longueur de clefs. Néanmoins, l'univers de l'embarqué est connu pour être vulnérable à une classe d'attaques particulières : les attaques par canaux auxiliaires, sujet présenté il y a quelques mois dans Misc n°41. Nous verrons dans cet article comment ces attaques s'appliquent en cryptographie aux courbes elliptiques et les contre-mesures qui ont été mises au point.

Avant-propos

Cet article a pour objet l'utilisation en pratique des courbes elliptiques dans le cadre de la sécurité de l'information et des systèmes embarqués. L'approche retenue n'est ni celle de la rigueur mathématique, ni celle de l'exhaustivité. Dans un souci de concision et d'accessibilité, nous aurons en particulier recours à de nombreux raccourcis mathématiques et emploierons certains abus de langage (tels que l'assimilation d'une classe d'équivalence modulo n à l'entier qui la représente naturellement). Nous renvoyons le lecteur intéressé par les dessous mathématiques de ce sujet passionnant à l'ouvrage de référence [CF05].

1. Introduction aux courbes elliptiques

Le sujet des courbes elliptiques ayant déjà été largement présenté dans *Misc* (en particulier dans le n°19 : « Cryptographie et courbes elliptiques » et le n°39 : « Courbes elliptiques et cryptographie : factorisation de grands nombres » [Phi08]), nous nous contentons ici d'un bref rappel des outils mathématiques mis en jeu et nous focalisons sur l'utilisation en pratique de cette cryptographie.

Rappelons tout d'abord que l'intérêt porté aux courbes elliptiques provient de la structure de groupe existant sur l'ensemble des points de celles-ci, structure connue depuis le XIXème siècle et très prisée aujourd'hui en cryptographie. Trois applications majeures et indépendantes leur ont été trouvées dans des domaines liés à la cryptologie : en 1985, Koblitz et Miller proposent chacun des cryptosystèmes asymétriques reposant sur les courbes elliptiques. La même année, Lenstra met au point la méthode de factorisation de grands nombres ECM (le sujet de l'article [Phio8]) et, en 1986, Atkin et Morain conçoivent le test de primalité ECPP qui reste à ce jour l'algorithme le plus efficace dans ce domaine.

1.1 Loi de groupe

Nous nous intéressons dans le cadre de cet article au seul cas des courbes elliptiques sur un corps premier F_p, c'est-à-dire (en simplifiant) l'ensemble des nombres modulo un grand nombre premier p.

Une courbe elliptique sur un tel corps peut être décrite comme l'ensemble des points (x,y) satisfaisant l'équation $(E): y^2=x^3+ax+b \bmod p$ (avec $4a^3+27b^2 \neq 0 \bmod p$), auquel on inclut un point supplémentaire O qui n'a pas de représentation en coordonnées (x,y) du plan euclidien et que l'on appelle « point à l'infini ». Notons ici que, sous cette forme, une courbe elliptique est entièrement définie par les trois paramètres p, a et b. Les courbes utilisées en cryptographie sont bien sûr standardisées, le standard le plus connu étant celui du NIST [FIPS] qui a défini cinq courbes elliptiques sur des corps premiers dont les paramètres font respectivement 192, 224, 256, 320 et 521 (!) bits.

Pour la définition rigoureuse d'un groupe, nous renvoyons le lecteur aux articles précédents ou à *Wikipédia*. Rappelons ici simplement que cette structure implique l'existence d'une loi de composition interne dans l'ensemble des points de la courbe telle que nous l'avons décrite, c'est-à-dire une opération qui, à deux points de l'ensemble, associe toujours un troisième point de l'ensemble. De plus, cette loi possède les propriétés d'associativité et de commutativité. Enfin, le point O y joue le rôle d'élément neutre.

Cette loi, dite « sécante-tangente », est bien connue et régulièrement illustrée sur l'ensemble des nombres réels (voir [Phio8]). Notée comme une addition, elle se comporte différemment selon que les deux points en entrée sont égaux ou non : on parle plus particulièrement de doublement si c'est le cas et d'addition sinon. En effet, le calcul des coordonnées (x_R, y_R) où R est le résultat de l'addition de deux points P et Q est différent selon que $P \neq Q$ ou bien que $P=Q$:

- si $P \neq Q$ alors $x_R = m^2-x_P-x_Q$ et $y_R = m(x_P-x_R)-y_P$, avec $m = (y_Q-y_P)/(x_Q-x_P)$
- si $P=Q$ alors $x_R = m^2-2x_P$ et $y_R = m(x_P-x_R)-y_P$, avec $m = (3x_P^2+a)/(2y_P)$
- de plus, comme O est l'élément neutre du groupe, $P+O=P$ pour tout point P de la courbe.

La définition de cette loi de groupe nous amène alors à définir l'opération de **multiplication scalaire**. On appelle multiplication du point P par le scalaire k et on note $k \cdot P$ le résultat de la somme $P+P+...+P$ (k fois). D'un point de vue algorithmique, le calcul d'une multiplication scalaire est un problème similaire à celui du calcul d'une exponentiation.

1.2 Problème du logarithme discret

Le problème du logarithme discret sur le groupe de points d'une courbe elliptique est fondamental pour nous, puisqu'il s'agit du problème « difficile » sur lequel repose la cryptographie des courbes elliptiques. Ce problème s'exprime comme suit : étant donné une courbe E sur un corps F_p, étant donné un point P sur cette courbe générateur du groupe [1] et un autre point Q de la courbe, comment trouver le plus petit entier d tel que $Q=d \cdot P$?

La théorie des groupes nous permet d'affirmer que d est strictement inférieur au nombre de points sur la courbe et, de plus, que le nombre n d'éléments du groupe de points d'une courbe elliptique sur un corps F_p est de l'ordre de p [2]. Pour se prémunir d'une recherche exhaustive sur les points de la courbe, il est donc nécessaire dans ce cadre-ci de prendre p au moins de l'ordre de 2^{80}, cette borne étant généralement admise comme limite à la force brute.

Bien sûr, la recherche mathématique a mis au point des méthodes plus efficaces pour la résolution de ce problème. Néanmoins, l'algorithme le plus efficace à ce jour pour extraire dans le cas général le logarithme discret d'un point d'une courbe elliptique est la méthode ρ de Pollard conçue dans les années 70 dans le cadre du problème de la factorisation des entiers. Cette méthode est de complexité exponentielle : $O(\sqrt{n})$. Pour obtenir un niveau de sécurité de l'ordre de 2^{80} opérations, il est alors nécessaire de choisir p de l'ordre de 2^{160}.

La recherche mathématique a peu avancé sur cette question depuis 30 ans, alors que, dans le domaine de la factorisation des entiers, par exemple, des algorithmes de complexité sous-exponentielle en temps ont été mis au point. C'est pour cette raison que la taille des clefs cryptographiques du système RSA dépasse de loin celle des cryptosystèmes fondés sur les courbes elliptiques à niveau de sécurité équivalent. C'est cet avantage qui motive l'intérêt croissant porté à la cryptographie sur les courbes elliptiques. Le tableau 1, page suivante, fournit les estimations habituellement proposées des tailles de clef RSA et pour les courbes elliptiques équivalentes en termes de sécurité. Notons toutefois que ces estimations diffèrent suivant les sources [BK].

Tailles de clefs en cryptographie symétrique	80	112	128	192	256
Courbes elliptiques	160	224	256	384	512
RSA	1024	2048	3072	8192	15360

Tableau 1 : Estimation des tailles de clefs (en bits) nécessaires aux cryptosystèmes reposant sur les courbes elliptiques et RSA pour obtenir des niveaux de sécurité équivalents à ceux de la cryptographie symétrique.

2. Les protocoles ECDSA et ECDH

Nous nous intéressons dans le cadre de cet article aux protocoles ECDSA (*Elliptic Curve Digital Signature Algorithm* [ANSI05]) et ECDH (*Elliptic Curve Diffie-Hellman* [ANSI01]). Le premier est une transposition dans le cadre des courbes elliptiques du protocole de signature bien connu DSA (*Digital Signature Algorithm* [FIPS]). De même, le second est une adaptation du protocole de partage de secret de Diffie et Hellman [DH76], initialement conçu sur un groupe multiplicatif F_p^*, au groupe des points d'une courbe elliptique.

2.1 Protocole de signature ECDSA

Les paramètres publics requis par le schéma de signature ECDSA sont les suivants : une courbe elliptique $E(F_p)$, le nombre de points n sur cette courbe, un point P générateur du groupe des points et une fonction de hachage H. Un utilisateur crée alors sa clef privée en tirant aléatoirement un entier d dans $[1,n-1]$ et en déduit sa clef publique constituée du point $Q=d\cdot P$.

La signature d'un message m est un couple (r,s) calculé comme suit :

1. Tirer aléatoirement k dans $[1, n-1]$
2. $P_1 \leftarrow k\cdot P$
3. $r \leftarrow x_{P_1} \bmod n$
4. Si $r=0$ revenir à l'étape 1
5. $s \leftarrow k^{-1}(H(m)+dr) \bmod n$
6. Si $s=0$ revenir à l'étape 1
7. Retourner (r,s)

La vérification de cette signature doit alors être effectuée selon la méthode suivante :

1. Si r ou $s \notin [1,n-1]$, retourner `Non valide`
2. $s' \leftarrow s^{-1} \bmod n$
3. $P_1 \leftarrow (s'H(m))\cdot P + (s'r)\cdot Q$
4. Si $P_1 = O$, retourner `Non valide`
5. Si $x_{P_1} \neq r \bmod n$, retourner `Non valide`
6. Retourner `Valide`

L'algorithme de vérification est donné ici à titre indicatif, puisqu'il ne constitue bien évidemment pas une cible potentielle pour les attaques par canaux auxiliaires : toutes les entrées du calcul sont publiques, ce qui est souhaitable si l'on veut que quiconque puisse vérifier la signature. Examinons par contre l'algorithme de génération : s'il est clair que la multiplication modulaire dr ne doit laisser fuir aucune information sur la clef secrète d, il est peut-être moins immédiat de voir que la valeur aléatoire k doit, elle aussi, faire l'objet de toutes les attentions. En effet, la connaissance de k, associée à la signature (r,s) correspondante, révèle directement la clef secrète : $d=(sk-H(m))/r$ (ceci en vertu des calculs effectués à l'étape 5). La multiplication scalaire $k\cdot P$ ne doit donc pas laisser fuir d'information sur k.

2.2 Protocole de partage de secret ECDH

Le protocole ECDH permet à deux entités A et B de s'accorder sur une clef secrète en communiquant sur un canal non sûr étant donné une courbe elliptique $E(F_p)$ et un point P de cette courbe générateur du groupe des points.

Le protocole se déroule comme suit :

- A tire aléatoirement un entier a dans $[1,n-1]$ et calcule $P_a=a\cdot P$
- B tire aléatoirement un entier b dans $[1,n-1]$ et calcule $P_b=b\cdot P$
- A envoie P_a à B et B envoie P_b à A
- A et B calculent alors le secret partagé $P_{ab}=b\cdot P_a=a\cdot P_b$

Un espion surveillant le canal de communication peut capturer les valeurs des points P_a et P_b lors de l'échange, mais, s'il veut en déduire P_{ab}, il se trouve confronté au problème du logarithme discret pour extraire a de P_a ou b de P_b.

Ce protocole étant parfaitement symétrique, on considère sans perte de généralité que A représente le composant embarqué et que la place de B peut être usurpée par un attaquant (par exemple si une carte à puce se retrouve entre de mauvaises mains). Du point de vue des attaques par canaux auxiliaires, les deux multiplications scalaires $a\cdot P$ et $a\cdot P_b$ doivent être sécurisées puisqu'elles font intervenir le paramètre secret a. Néanmoins, si le secret a est propre au composant, il suffit de calculer et

stocker à l'avance (hors ligne) le point P_a dans un environnement sûr, ce qui supprime la menace d'une attaque sur ce calcul. Dans ce cas, c'est donc la multiplication scalaire $a \cdot P_b$ qui doit être sécurisée.

2.3 Contextes de sécurité des protocoles

Nous avons identifié les calculs manipulant des informations sensibles pour les deux protocoles étudiés. Il est maintenant nécessaire de regarder de plus près les possibilités qui s'offrent à un attaquant dans chaque cas afin de préciser les contextes de sécurité précis de l'ECDSA et de l'ECDH.

Notons en premier lieu que la multiplication scalaire sensible du protocole ECDSA fait intervenir un scalaire aléatoire. Un attaquant doit donc parvenir à récolter assez d'information sur ce scalaire pour mener à bien son attaque en une seule exécution, puisque les exécutions suivantes feront intervenir de nouveaux scalaires aléatoires. D'autre part, le point multiplié est fixe (il est spécifié par le standard adopté), impossible donc pour l'attaquant de jouer sur le choix de ce point.

À l'inverse, dans le cas du protocole ECDH, c'est B – un attaquant potentiel – qui fournit le point P_b mis en jeu dans la multiplication scalaire sensible $a \cdot P_b$. De plus, le protocole est ainsi implémenté que le composant A autorise la répétition de l'opération finale autant de fois que souhaité. En d'autres termes, un attaquant a la possibilité de demander à une carte à puce d'effectuer autant de multiplications scalaires $a \cdot P_{b_i}$, avec le même scalaire secret a et des points P_{b_i} choisis par ses soins, qu'il lui est nécessaire pour mener son attaque.

Les contextes sécuritaires des protocoles ECDSA et ECDH sont donc bien différents dans le cadre des attaques par canaux auxiliaires. En particulier, le protocole de signature présente une surface d'attaque bien plus restreinte que le protocole de partage de secret. Ceci nous permettra ensuite d'identifier les attaques applicables à chacun de ces protocoles.

3. Rappels sur les attaques par canaux auxiliaires

Les attaques par canaux auxiliaires (parfois aussi appelés « canaux cachés ») ont été présentées en détail dans l'article [DP09]. Nous nous contentons donc ici d'en rappeler les points principaux.

Les attaques par canaux auxiliaires sont presque exclusivement réservées au monde de l'embarqué. Elles tirent parti des interactions entre un module et son milieu extérieur lorsque celui-ci est amené à manipuler des données sensibles. Ces canaux d'interaction sont classiquement le temps d'exécution (*timing attacks*), la consommation électrique (*power analysis* ou PA) et le rayonnement électromagnétique (*electro-magnetic analysis* ou EMA). Ces deux derniers nous intéressent plus particulièrement dans le cadre de cet article.

Les fuites qui empruntent ces canaux sont de deux ordres : les fuites logicielles et les fuites matérielles. Les fuites logicielles sont le fruit du passage des algorithmes dans des branches conditionnelles différentes, ce qui conduit à l'exécution d'opérations différentes par le module embarqué, lesquelles sont trahies par des profils de consommation ou de rayonnement différents (voir Figures 1 et 2). Les fuites matérielles résultent,

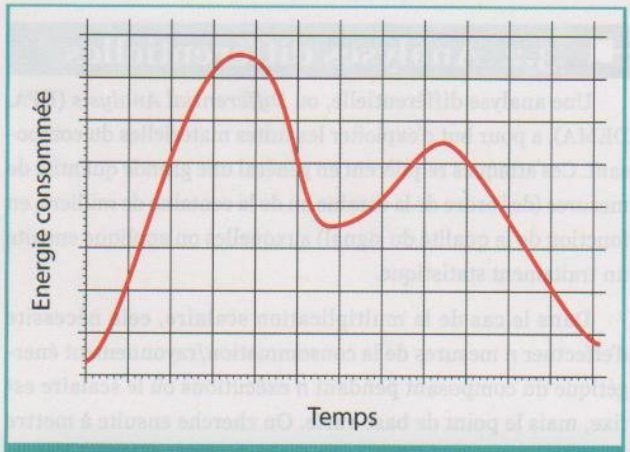


Figure 1 : Profil de consommation d'énergie du composant C pendant l'opération O1

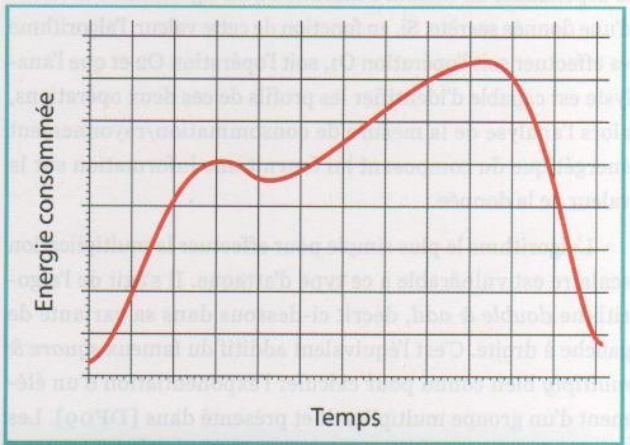


Figure 2 : Profil de consommation d'énergie du composant C pendant l'opération O2

elles, de la subtile différence d'énergie nécessaire à la manipulation d'une donnée par le calculateur en fonction de sa valeur (voir Figure 3). On considère généralement que l'énergie nécessaire à un micro-processeur pour placer une donnée dans un registre est fonction de la distance de Hamming (le nombre de bits différents) entre la valeur de cette donnée et la valeur précédente du registre. Ce modèle est parfois simplifié en ne tenant compte que du poids de Hamming (nombre de bits à 1) de la valeur de la donnée.

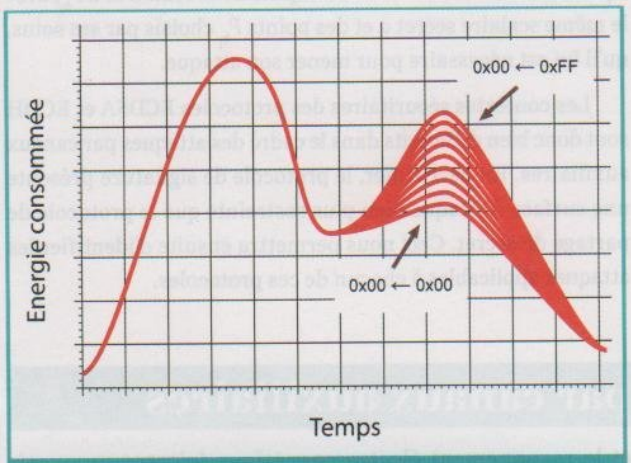


Figure 3 : Profils de consommation d'énergie du composant C pendant l'opération O1 où un registre d'un octet prend toutes les valeurs possibles. Les 9 courbes correspondent aux 9 distances de Hamming possibles.

3.1 Analyses élémentaires

Une attaque est dite « élémentaire » si une unique mesure est suffisante pour fournir l'information souhaitée. On parle alors de *Simple Analysis* et donc de SPA ou SEMA. En général, une analyse élémentaire exploite une fuite logicielle résultant de la dépendance du chemin d'exécution d'un algorithme à la valeur d'une donnée secrète. Si, en fonction de cette valeur, l'algorithme va effectuer soit l'opération O1, soit l'opération O2 et que l'analyste est capable d'identifier les profils de ces deux opérations, alors l'analyse de la mesure de consommation/rayonnement énergétique du composant lui fournit une information sur la valeur de la donnée.

L'algorithme le plus simple pour effectuer la multiplication scalaire est vulnérable à ce type d'attaque. Il s'agit de l'algorithme *double & add*, décrit ci-dessous dans sa variante de gauche à droite. C'est l'équivalent additif du fameux *square & multiply* bien connu pour calculer l'exponentiation d'un élément d'un groupe multiplicatif et présenté dans [DP09]. Les entrées de l'algorithme sont la courbe elliptique $E(F_p)$, un point P de celle-ci et un scalaire $k=(k_{l-1}...k_1k_0)_2$ de taille l bits. L'algorithme retourne $k \cdot P$.

1. $Q \leftarrow O$
2. Pour i de l-1 à 0 faire
   a. $Q \leftarrow 2Q$
   b. Si k_i=1 alors
      $Q \leftarrow Q+P$
3. Retourner Q

Les doublements et additions de points présents aux étapes 2.a. et 2.b. de l'algorithme correspondent aux deux formes prises par la loi de groupe. Ces deux opérations sont effectuées à l'aide de formules différentes comme nous l'avons vu en 1.1. Leurs profils de consommation énergétique sont donc différents et un attaquant peut directement lire les bits du scalaire : un doublement non suivi d'une addition correspond à un 0 et un doublement suivi d'une addition correspond à un 1 (voir Figure 4). Ce type d'attaque est toujours applicable puisque qu'il ne requiert qu'une mesure et ne fait aucune hypothèse sur le point de base.

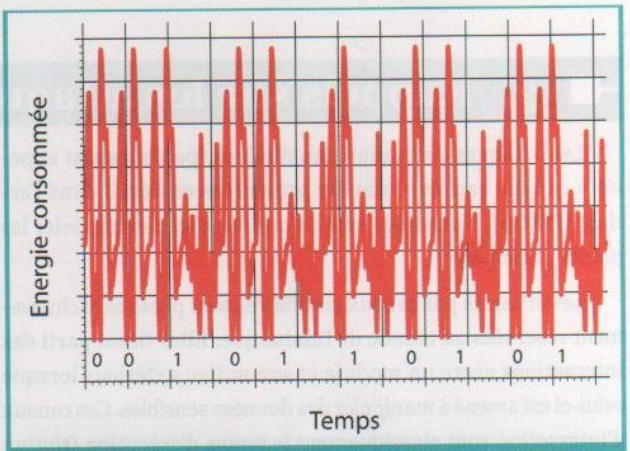


Figure 4 : Profil de consommation d'énergie du composant C pendant l'exécution de l'algorithme double & add

3.2 Analyses différentielles

Une analyse différentielle, ou *Differential Analysis* (DPA, DEMA), a pour but d'exploiter les fuites matérielles du composant. Ces attaques requièrent en général une grande quantité de mesures (de l'ordre de la dizaine ou de la centaine de milliers, en fonction de la qualité du signal) auxquelles on applique ensuite un traitement statistique.

Dans le cas de la multiplication scalaire, cela nécessite d'effectuer n mesures de la consommation/rayonnement énergétique du composant pendant n exécutions où le scalaire est fixe, mais le point de base varie. On cherche ensuite à mettre en évidence une corrélation entre les valeurs intermédiaires manipulées par le composant, dont dépendent les courbes obtenues, et la valeur du scalaire secret sur laquelle on effectue

des hypothèses. Les contraintes de ces attaques empêchent de l'appliquer à la multiplication scalaire de la signature ECDSA (scalaire aléatoire, point de base fixe). En revanche, il est possible d'attaquer de cette manière la deuxième multiplication scalaire du protocole ECDH.

Notons $C_1, C_2,..., C_n$ les courbes obtenues pour les multiplications scalaires $k \cdot P_1, k \cdot P_2,..., k \cdot P_n$ avec l'algorithme *double & add* (ou *double & add always*, présenté plus loin). Pour découvrir le deuxième bit à gauche de scalaire k_{l-2} (le premier bit à gauche vaut trivialement 1), on fait une hypothèse sur celui-ci : par exemple sa valeur est 1. Observons que les points respectifs $3P_1, 3P_2,..., 3P_n$ sont manipulés par l'algorithme à chaque exécution si notre hypothèse est vraie, mais ne le sont pas dans le cas contraire. Définissons de plus une fonction de décision $d:E(F_p) \rightarrow \{0,1\}$, c'est-à-dire une fonction booléenne qui classe les points de la courbe en deux sous-ensembles selon un critère arbitraire. Il est par exemple possible de classer les points en fonction de la valeur du premier bit de leur représentation. Ceci nous conduit à distinguer deux sous-ensembles de courbes de mesures : celles pour lesquelles $d(3P_i)=0$ et celles pour lesquelles $d(3P_i)=1$.

Si l'hypothèse sur le bit de scalaire est correcte, il doit exister une corrélation entre les deux sous-ensembles de courbes que nous avons établis. En effet, nous avons vu qu'il existe une fuite des composants en fonction des valeurs des données manipulées. Or, nous avons précisément créé ces sous-ensembles pour qu'à un instant t l'un ne contienne que des courbes où c'est un 0 qui est manipulé et l'autre que des courbes où c'est un 1. La position exacte de cet instant n'est pas primordiale : le moyen le plus simple de révéler cette corrélation est de calculer la courbe suivante $\langle C_i/d(3P_i)=0\rangle - \langle C_i/d(3P_i)=1\rangle$, c'est-à-dire la différence des moyennes des courbes de chaque sous-ensemble. Si cette nouvelle courbe laisse apparaître un ou plusieurs pics – alors appelés « pics de corrélation » – comme sur la figure 5, c'est que le point $3P_i$ a effectivement été manipulé et qu'il l'a été aux instants précis indiqués par ces pics. Sinon, la courbe ne montre que du bruit et l'hypothèse sur le bit de scalaire est fausse : celui-ci vaut 0.

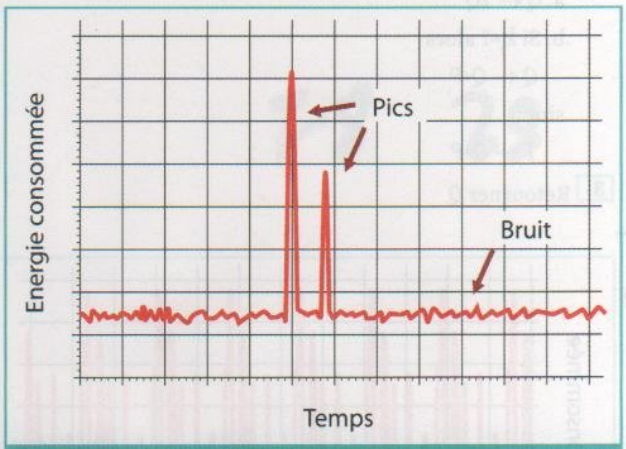


Figure 5 : Différence des moyennes des courbes des deux sous-ensembles laissant apparaître des pics de corrélation

De cette manière, les bits du scalaire sont découverts les uns après les autres. Précisons toutefois que cette attaque est plus difficile à mettre en œuvre qu'une analyse élémentaire. D'autre part, des attaques différentielles plus raffinées ont été mises au point qui font intervenir des outils statistiques plus fins ou exploitent des dépendances temporelles entre les données manipulées (*Higher Order Differential Analysis* ou HODPA/HODEMA).

4. Contre-mesures classiques

Nous abordons dans cette partie les contre-mesures logicielles applicables pour se prémunir des attaques vues ci-dessus. D'autres mesures participent à la sécurité globale de la carte à puce : des contre-mesures matérielles, électroniques... Nous nous intéressons ici néanmoins à la sécurisation des algorithmes indépendamment de celle des autres couches, selon le principe de sécurité en profondeur.

4.1 Vis-à-vis des analyses élémentaires

Il existe principalement trois méthodes pour protéger une implémentation de la multiplication scalaire des attaques SPA/SEMA :

- choisir un algorithme régulier ;
- utiliser une formule unifiée ;
- appliquer le principe d'atomicité.

Ces trois méthodes ont pour but de rendre inexploitable la connaissance de la succession des opérations effectuées par l'algorithme à un attaquant qui tenterait une analyse élémentaire. Comme nous allons le voir, chacune agit à un niveau algébrique différent : du plus grossier au plus fin.

4.1.1 Algorithmes réguliers

La première méthode consiste à modifier un algorithme pour rendre sa structure régulière et indépendante du scalaire ou à choisir un algorithme possédant cette propriété. Il est ainsi

immédiat de transformer l'algorithme *double & add* (présenté en 3.1) en *double & add always* décrit ci-après. La transformation consiste à insérer une addition factice lorsque le bit courant est un 0. Le profil de consommation/rayonnement énergétique est alors illustré en figure 6.

1. $Q, T \leftarrow O$
2. Pour i de l-1 à 0 faire
   a. $Q \leftarrow 2Q$
   b. Si k_i=1 alors
      $Q \leftarrow Q+P$
   sinon
      $T \leftarrow Q+P$
3. Retourner Q

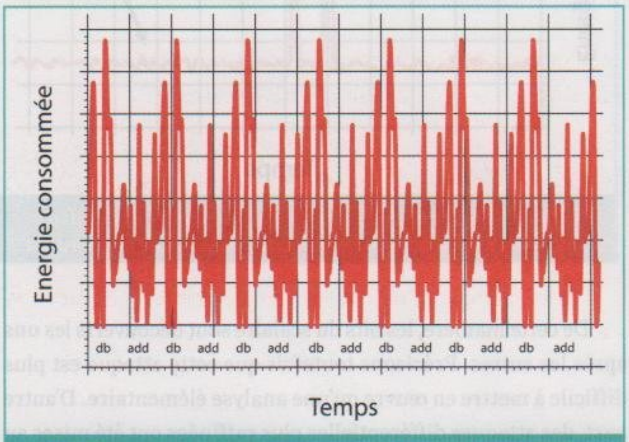


Figure 6 : Profil de consommation d'énergie du composant C pendant l'exécution de l'algorithme double & add always

L'inconvénient de l'algorithme *double & add always* est bien sûr la dégradation des performances qu'il induit : le nombre d'additions exécutées par la multiplication scalaire est en moyenne doublé. Un autre algorithme régulier utilisable pour se prémunir des analyses élémentaires à moindre frais qu'en appliquant *double & add* est l'algorithme *Montgomery ladder* (voir [FGKS02]).

4.1.2 Formule unifiée

La deuxième contre-mesure consiste à utiliser une formule unifiée d'addition et de doublement trouvée en 2002 par Brier et Joye dans [BJ02]. Dès lors qu'une formule unique permet d'effectuer addition et doublement de point, ces deux opérations deviennent indiscernables l'une de l'autre par analyse élémentaire comme l'illustre la figure 7. En théorie, on s'affranchit aussi du surcoût de temps d'exécution qu'implique le choix d'un algorithme régulier comme *double & add always*. En pratique, il s'avère malheureusement que cette formule est tellement coûteuse – en termes de nombre d'opérations sur le corps de base F_p – que les performances d'une telle implémentation sont finalement moins bonnes que celle d'une implémentation de l'algorithme *double & add always*.

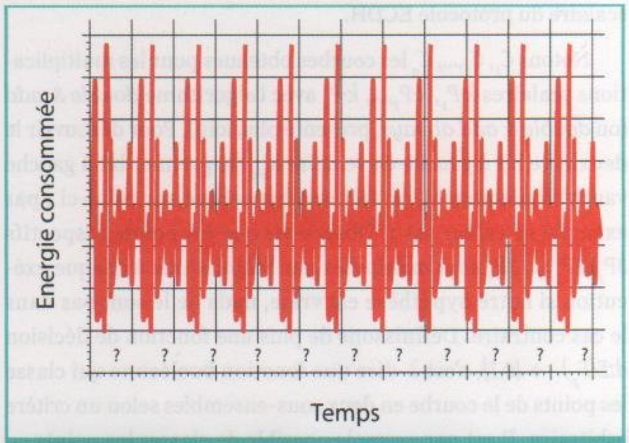


Figure 7 : Profil de consommation d'énergie du composant C effectuant une multiplication scalaire avec une formule unifiée d'addition de points

4.1.3 Atomicité

La dernière méthode de protection consiste à appliquer le principe d'algorithme régulier à l'échelle de l'opération d'addition ou de doublement. Ce concept d'atomicité a été introduit par Chevallier-Mames, Ciet et Joye dans [CCJ03] (voir encadré). Dans le cadre de la multiplication scalaire, cela revient à exprimer le doublement et l'addition de points comme une succession de blocs atomiques (des blocs d'opérations sur le corps de base F_p). L'analyse élémentaire ne révèle alors qu'une succession ininterrompue de ces blocs atomiques (voir Figure 8) et il est impossible pour un attaquant de déterminer où commence et où finit une opération. Les doublements et les additions s'en trouvent donc masqués. La dégradation des performances causée par cette méthode est bien inférieure comparée aux deux premières. Cette protection peut de plus être appliquée dans tous les cas et quelles que soient les formules d'addition/doublement (voir l'encadré de la page suivante).

4.2 Vis-à-vis des analyses différentielles

Nous présentons ici les contre-mesures les plus connues, présentées par Coron en 1999 dans [Cor99]. Ces astuces algébriques tirent profit des propriétés d'un groupe ou de la représentation des points utilisée pour les calculs pour *randomiser* (masquer) les valeurs des données manipulées par le composant. De cette manière, ces valeurs sont différentes à chaque exécution de la multiplication scalaire et les attaques différentielles simples sont inefficaces. Nous ne traiterons pas des attaques mises au point plus récemment (voir par exemple [FV03] et [Gou03]).

Illustrons le principe d'atomicité dans le cadre plus simple d'une exponentiation RSA, notée $a^k \bmod n$, réalisée par la méthode *square & multiply* : de manière équivalente à *double & add*, cette méthode (algorithme 1) parcourt successivement les bits de l'exposant et effectue une élévation au carré modulaire suivie d'une multiplication quand le bit est 1 ou une simple élévation au carré si ce bit est 0. Protéger l'implémentation de cet algorithme par atomicité revient à toujours utiliser l'opération de multiplication modulaire, que l'on effectue une multiplication normale ou une élévation au carré (algorithme 2).

Algorithme 1 :

1. $r \leftarrow 1$
2. Pour i de l-1 à 0 faire
   a. $r \leftarrow \mathbf{r^2} \bmod n$
   b. Si k_i=1 alors
      $r \leftarrow r \times a \bmod n$
3. Retourner r

Algorithme 2 :

1. $r \leftarrow 1$
2. Pour i de l-1 à 0 faire
   a. $r \leftarrow \mathbf{r \times r} \bmod n$
   b. Si k_i=1 alors
      $r \leftarrow r \times a \bmod n$
3. Retourner r

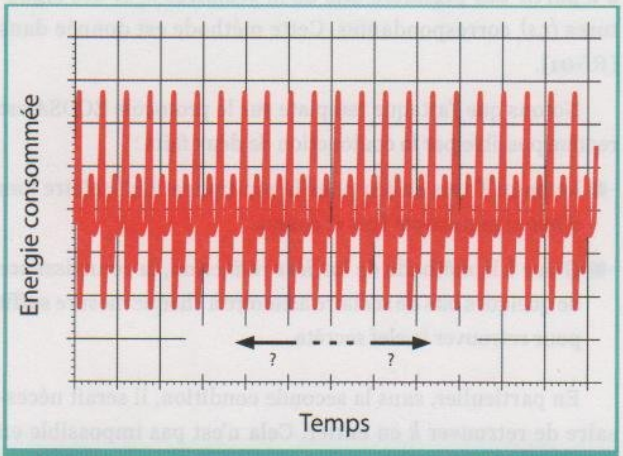


Figure 8 : Profil de consommation d'énergie du composant C effectuant une multiplication scalaire lorsque le principe d'atomicité est mis en oeuvre

Il est possible d'appliquer ces contre-mesures ensemble ou séparément en fonction des contraintes de l'environnement et des exigences sécuritaires.

4.2.1 Masquage du scalaire

Dans un groupe additif fini G de n éléments et d'élément neutre e, il existe la propriété fondamentale suivante : pour tout $x \in G$ on a $nx=e$.

La première contre-mesure de Coron consiste à tirer parti de cette propriété algébrique pour appliquer un masquage au scalaire. Pour la mettre en œuvre, il suffit de choisir un aléa r de taille l_r bits et d'effectuer la multiplication scalaire $k \cdot P = (rn + k) \cdot P$ où n est le nombre de points de la courbe elliptique.

Cette contre-mesure est très simple à mettre en place et s'avère efficace. Du choix de l_r dépend le compromis sécurité/surcoût en termes de temps d'exécution engendré : il faut ajouter l_r à la taille de clef (typiquement 160 à 521 bits) pour évaluer le temps d'exécution de la multiplication scalaire.

4.2.2 Masquage du point de base

La seconde contre-mesure de Coron repose, elle, sur l'idée de masquer le point de base pendant la multiplication scalaire.

Cette contre-mesure consiste à calculer la multiplication scalaire $k \cdot (P+R)$, où R est un point généré pseudo-aléatoirement, puis en soustrayant $S = k \cdot R$ au résultat. Cette approche nécessite naïvement deux multiplications scalaires au lieu d'une, ce qui constitue un surcoût inacceptable pour de nombreuses applications. Il est néanmoins possible d'appliquer efficacement cette technique si k est fixé en utilisant la méthode suivante : à l'initialisation du système le point R est généré et le point $S = k \cdot R$ est calculé. Lors des multiplications scalaires suivantes, les points R et S sont mis à jour en tirant α dans $\{0,1\}$ et en effectuant $R \leftarrow (-1)^\alpha 2R$ et $S \leftarrow (-1)^\alpha 2S$. R est ainsi différent à chaque exécution et le surcoût engendré n'est que de deux additions et deux doublements de point.

Bien qu'intéressante sur le papier, cette technique s'avère rarement applicable, en particulier à cause de la nécessité pour k d'être fixe.

4.2.3 Masquage des coordonnées projectives du point de base

Lorsqu'une multiplication scalaire est effectuée, les calculs en interne n'utilisent en général pas la représentation affine naturelle (x, y) des points, mais une autre forme appelée « représentation projective » notée $(X:Y:Z)$. Cette représentation possède sur la représentation affine l'avantage de produire des formules d'addition et de doublement de points ne nécessitant pas d'inversion modulaire, opération très coûteuse. Deux types de

représentation projective sont envisageables, mais nous ne considérerons ici que la plus simple d'entre elles : la représentation projective homogène. Cette représentation, comme sa variante jacobienne, a la particularité d'être redondante : en coordonnées projectives homogènes, si λ désigne un entier non nul, alors $(X:Y:Z)$ et $(\lambda X:\lambda Y:\lambda Z)$ sont deux représentants du même point.

La troisième contre-mesure de Coron consiste à exploiter la propriété énoncée ci-dessus. Avant chaque multiplication scalaire $k \cdot P$ avec $P = (X:Y:Z)$, un aléa r est tiré et on donne en entrée de l'algorithme le nouveau représentant de P $(rX:rY:rZ)$. Les valeurs manipulées ensuite par l'algorithme sont ainsi *randomisées*.

Cette mesure est efficace et peu coûteuse (seulement 2 multiplications modulaires, car en pratique $Z=1$). Elle est donc généralement appliquée si les coordonnées projectives homogènes ou jacobiennes (la contre-mesure s'adapte) sont choisies.

5. Cas particulier de l'attaque template sur ECDSA

Nous présentons dans cette dernière partie une attaque spécifique au protocole ECDSA. Comme nous allons le voir, cette attaque tire parti des fuites matérielles du composant ; elle est néanmoins classée parmi les analyses élémentaires, ceci parce que chaque mesure donne à l'attaquant quelques bits de scalaire. Cette attaque est donc particulièrement puissante et nécessite des précautions particulières.

5.1 Description de l'attaque

L'attaque se déroule en trois parties distinctes dont seule la deuxième nécessite un accès physique au composant attaqué. Les deux autres peuvent s'effectuer *off-line*.

5.1.1 Génération des templates

Cette partie est une étude ouverte du module à attaquer. Elle nécessite pour l'attaquant de posséder un composant identique et suppose un accès total à celui-ci.

Dans cette première partie, l'attaquant va mesurer le comportement (PA ou EMA) de la carte au cours de multiplications scalaires $k \cdot P$ où les n premiers bits de k vont prendre toutes les valeurs possibles. On obtient ainsi $2n$ enregistrements appelés *templates*. Pour éviter une explosion de l'occupation mémoire de ces templates, on prend en pratique $3 \leq n \leq 10$.

5.1.2 Attaque du composant

Ici, l'attaquant va réaliser l'attaque proprement dite sur le composant. Il doit pour cela faire signer m fois un même message à la carte, m dépendant de n choisi à l'étape précédente (pour $n=3$, l'ordre de grandeur est celui de la centaine de mesures). Les signatures (r,s) correspondantes sont conservées.

Les mesures obtenues sont alors confrontées aux templates générés précédemment, ce qui permet, pour chacune d'entre elles, de découvrir les n premiers bits $(k_{l-1}k_{l-2}...k_{l-n})_2$ du scalaire aléatoire.

5.1.3 Résolution du système

La dernière partie de l'attaque consiste à appliquer une méthode de *lattices reduction* qui calcule la clef secrète d à partir des premiers bits de m scalaires k_i et des signatures $(r,s)_i$ correspondantes. Cette méthode est donnée dans [RS01].

Notons que l'attaque template sur le protocole ECDSA est rendue possible par la conjonction de deux faits :

- Le point P mis en jeu pour la multiplication scalaire lors d'une signature est fixe et connu.
- Grâce à la méthode de *lattices reduction*, la connaissance de quelques bits de scalaire aléatoire à chaque mesure suffit pour retrouver la clef secrète.

En particulier, sans la seconde condition, il serait nécessaire de retrouver k en entier. Cela n'est pas impossible en théorie, mais nécessiterait de construire des templates pour tous les bits de k, soit 2^n, où n est la taille de la clef. Avec les tailles de clef utilisées actuellement ($n \geq 192$), ce n'est même pas imaginable.

5.2 Contre-mesures applicables

Bien que le scalaire éphémère k soit tiré aléatoirement à chaque exécution, ce qui interdit en principe les attaques différentielles, ce sont les contre-mesures conçues pour se prémunir de ces attaques qui vont permettre de protéger une implémentation de signature ECDSA de la « menace *template* ».

Parmi les contre-mesures possibles, le masquage des coordonnées projectives du point de base (voir 4.2.3) rend l'attaque template de la signature ECDSA inapplicable.

Conclusion

Nous avons dressé dans cet article un rapide aperçu des contraintes sécuritaires qu'imposent les attaques *side-channel* à l'implémentation sur matériel embarqué de l'opération de multiplication scalaire dans le cadre des protocoles ECDSA et ECDH. Nous avons vu en particulier les différents types de contre-mesures – astuces d'implémentation et pièges algébriques – applicables pour se prémunir de ces attaques. Le sujet que nous avons survolé ici occupe à temps complet des équipes de chercheurs de par le monde. Autant dire que nous avons à peine balayé du regard un domaine dans lequel le petit jeu du chat et de la souris que l'on connaît si bien en sécurité de l'information est loin d'être terminé ! ■

Notes

[1] Un élément d'un groupe additif fini est dit générateur de ce groupe si et seulement si les multiples successifs de cet élément décrivent tous les éléments du groupe.

[2] En effet, les bornes de Hasse-Weil garantissent que le nombre de points sur une courbe elliptique sur un corps *Fp* est compris entre $(\sqrt{p}-1)^2$ et $(\sqrt{p}+1)^2$.

Remerciements

Je souhaite remercier vivement Christophe Giraud, Benoit Feix et Vincent Guyot pour leur minutieuse relecture. Je salue également Hugues Thiebeauld, Nicolas Morin et toute l'équipe pessacaise d'Oberthur Technologies pour le plaisir que j'ai eu à travailler avec eux. Merci enfin à Robert Erra, Benjamin Caillat et à tous les membres du mastère SIS de l'ESIEA pour leurs riches enseignements.

Références

[ANSI01] ANSI X9.63-2001., Public Key Cryptography for The Financial Service Industry : Key Agreement and Key Transport Using Elliptic Curve Cryptography, *American National Standards Institute*, Nov. 20, 2001.

[ANSI05] ANSI X9.62-2005., Public Key Cryptography for The Financial Service Industry : The Elliptic Curve Digital Signature Algorithm (ECDSA), *American National Standards Institute*, Nov. 16, 2005.

[BJ02] BRIER (E.) et JOYE (M.), Weierstraß Elliptic Curves and Side-Channel Attacks, Public Key Cryptography, *LNCS*, 2002.

[BK] BlueKrypt, Cryptographic Key Length Recommendation, http://www.keylength.com, 2008.

[CCJ03] CHEVALLIER-MAMES ((B.), CIET (M.) et JOYE (M.), Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity}, *Cryptology ePrint Archive*, http://eprint.iacr.org/, 2003.

[CF05] COHEN (H.), FREY (G.) ..., Handbook of Elliptic and Hyperelliptic Curve Cryptography, *Chapman & Hall/CRC*, 2005.

[Cor99] CORON (J.-S.), Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, CHES'99, *LNCS*, 1999.

[DH76] DIFFIE (W.) et HELLMAN (M. E.), New directions in cryptography, *IEEE Transactions on Information Theory*, 1976.

[DP09] DOTTAX (E.) et PROUFF (E.), « La carte à puce, cœur de sécurité des systèmes mobiles », *MISC* n°41, 2009.

[FGKS02] FISCHER (W.), GIRAUD (C.) KNUDSEN (E. W.) et SEIFERT (J.-P.), Parallel scalar multiplication on general elliptic curves over Fp hedged against Non-Differential Side-Channel Attacks, *Cryptology ePrint Archive*, http://eprint.iacr.org/, 2002.

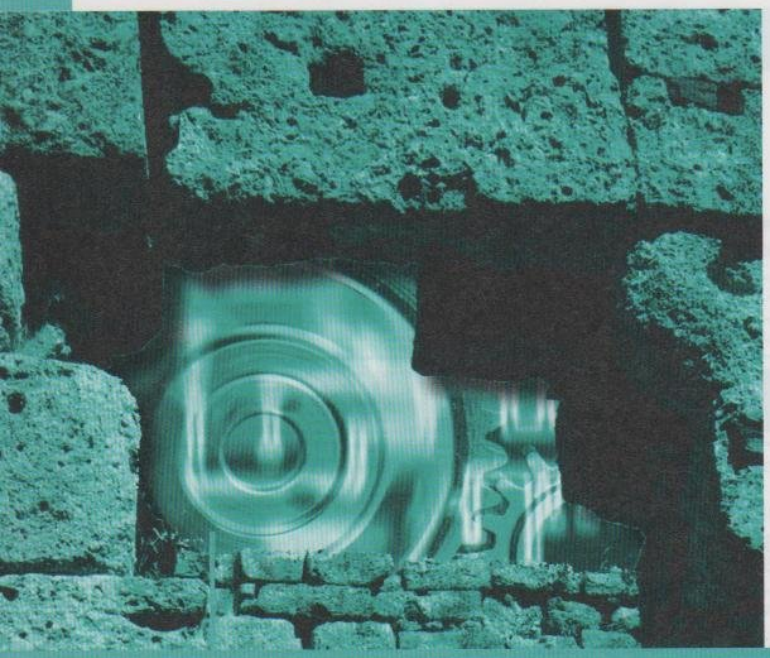
[FIPS] FIPS PUB 186-2., Digital Signature Standard, *National Institute of Standards and Technology*, Jan. 27, 2000.

[FV03] FOUQUE (P.-A.) et VALETTE (F.), The Doubling Attack: Why Upwards is Better than Downwards, CHES'03, *LNCS*, 2003.

[Gou03] GOUBIN (L.), A Refined Power-Analysis Attack on Elliptic Curve Cryptosystem, Public Key Cryptography, *LNCS*, 2003.

[Phi08] LAURENT (P.), « Courbes elliptiques et cryptographie : factorisation de grands nombres », *MISC* n°39, 2008.

[RS01] RÖMER (T.) et SEIFERT (J.-P.), Information leakage attacks against Smart Card implementations of the ECDSA, Esmart 2001, *LNCS*, 2001.



Adrien Derock DCNS/ESIEA
adrien.derock@dcnsgroup.com –
service Sécurité des Systèmes d'Information
adrien.derock@esiea-ouest.fr –
Laboratoire de virologie et de cryptologie opérationnelles

VULGARISATION DES ASPECTS FORMELS DE LA NOTION DE FURTIVITÉ

mots-clés : *furtivité / modèles formels / stéganographie / indistinguabilité calculatoire / graphe de référence*

Le rootkit représente une des principales menaces que combattent les éditeurs de logiciel de sécurité. Il s'agit d'un programme qui a pour principal but de se cacher au sein d'un système en utilisant des mécanismes de furtivité. Nous essayons dans cet article de vulgariser dans la mesure du possible les aspects formels décrivant la furtivité. Les rares ébauches de formalisation de ces mécanismes ont été développées dans le cadre spécifique de la modélisation des virus, ces derniers étant toujours assimilés à tort à des fonctionnalités malveillantes. Regardons de plus près les modèles formels qui nous permettent de mieux comprendre ce concept d'actualité.

Parmi les fonctionnalités anti-anti-virales les plus avancées, la furtivité est probablement l'une des plus sophistiquées. À ce jour, les *rootkits*[1] en constituent la réalisation la plus aboutie. Formaliser les techniques de furtivité est un exercice difficile. On peut cependant en donner la définition suivante :

L'objectif des techniques de furtivité est de dérober un programme (légitime ou non) à la surveillance de l'utilisateur et de tout programme d'analyse légitime (utilitaires système et/ou antivirus).

Remarquons qu'il découle de cette définition trois notions importantes :

1. La notion de furtivité s'applique à tout type de programmes (malveillant ou non).
2. La complexité du mécanisme de furtivité dépend aussi du degré de compétence de l'utilisateur visé.
3. Il semble évident qu'un utilisateur novice ne dispose pas des mêmes moyens de détection (matériel, logiciel ou humain) qu'un utilisateur expérimenté. Nous généraliserons dans la suite de ce document tous les moyens de détection existants par le terme de « détecteur ».

Enfin, la notion de furtivité pour tout code exige deux aptitudes :

- Résister à la détection dans un état statique (code malveillant inactif). On parlera ici de camouflage.
- Résister à la détection lors d'une activité. On parlera ici de furtivité totale.

On introduit également la notion de « détecteur » pour mettre en exergue le rapport de force omniprésent entre le mécanisme de furtivité et les outils de détection de modification tant bénigne que maligne.

Un détecteur est un moyen permettant de repérer, sur un système donné, son éventuel altération.

Ceci peut être opéré par le biais des outils mis à disposition par le système lui-même ou simplement par l'observation de l'utilisateur (par ses connaissances) ou bien encore par un outil matériel ou logiciel assurant l'intégrité du système.

Après ces brefs rappels, examinons de plus près les différentes facettes de la furtivité.

1. La furtivité, l'art d'imiter

Durant les travaux de formalisation des virus, il a été arrêté que la forme infectée d'un programme peut agir selon trois possibilités (voir Figure 1) [1] en fonction des entrées fournies par l'utilisateur :

- Infection – Le programme, après avoir réalisé les fonctionnalités attendues, infecte d'autres programmes (il se reproduit).
- Fonctionnalité ajoutée – Le programme, en plus de ses fonctionnalités attendues, effectue d'autres actions. Ces actions peuvent être différées ou non, à caractère (généralement) offensif ou non et dont la nature dépend uniquement de l'infection initiale et non du programme infecté.
- Imitation – Le programme ne procède à aucune infection, ni action offensive, mais effectue juste les instructions légitimement attendues.

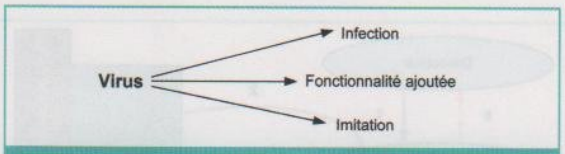


Figure 1 : Fonctionnalités des virus

Dans cette définition, la notion de « furtivité » transparaît dans la troisième possibilité. En effet, l'imitation est le but ultime de la furtivité. Les détecteurs doivent croire à un fonctionnement normal du système. Le système infecté doit autant que possible imiter le système sain et renvoyer une image du système ne déclenchant aucun soupçon d'infection de la part des détecteurs. La forme infectée du programme a exactement le même comportement que la forme saine (voir Figure 2).

Figure 2 : Mécanismes de furtivité dans un système

Bien entendu, cette fonctionnalité a été soulignée ici sans donner plus de précision sur la nature de cette imitation. Mais, cette base de départ a donné des idées à d'autres.

2. La furtivité, l'art de modifier le système

Les tentatives de formalisation des infections informatiques ont fourni plusieurs résultats très intéressants, notamment sur la complexité des ensembles viraux évolués (virus polymorphes, virus furtifs) et sur la détection de tels ensembles [2].

À travers ces travaux, l'exécution de tout programme est définie comme dépendant de deux paramètres :

- les données ;
- le contexte d'exécution (Système d'exploitation en exécution, différents programmes système ou utilisateurs).

Le principal intérêt de ces travaux est une mise en exergue de l'environnement intervenant dans l'exécution d'un programme. Un programme n'est pas autosuffisant pour réaliser sa tâche. Il fait appel généralement à des ressources du système. D'où l'importance du contexte d'exécution. Cette granularité permet au modèle proposé de détailler davantage le mécanisme de furtivité au sein du système.

2.1 Cadre formel

Voici ce qui a été proposé concernant les virus furtifs :

1. Un virus furtif lorsqu'il est en phase d'incubation[2] doit faire réaliser au programme infecté les mêmes actions que la version saine du programme.
2. Un virus furtif lorsqu'il se reproduit (infection d'un autre objet) doit sélectionner un programme à infecter et doit modifier l'ensemble (un ou plusieurs) des appels système pour leurrer les détecteurs.

L'apport majeur dans ce modèle est la notion de modification d'un appel système bien précis, dans le but de faire croire au système la non-infection d'un objet. Par exemple, cet appel système peut être celui chargé de consulter le système de fichiers dans un répertoire donné. En renvoyant une réponse adéquate, le système infecté paraîtra sain. Ainsi, à chaque fois qu'un objet est infecté, l'appel système susceptible de trahir la présence de cette infection va être modifié en conséquence.

Lorsque l'objet en question est infecté, l'appel système se comportera de telle manière à ne pas divulguer des informations pouvant alerter les détecteurs et le système de son infection. Lorsque l'objet est sain, le comportement de l'appel système est normal.

Ce formalisme donne une bonne définition de la notion de « virus furtif ».

2.2 Principe de fonctionnement d'un mécanisme de furtivité

Pour illustrer cette formulation, nous allons nous focaliser sur un exemple concret. Prenons comme cas de figure un virus qui va infecter un dossier du système de fichiers. (voir Figure 3)

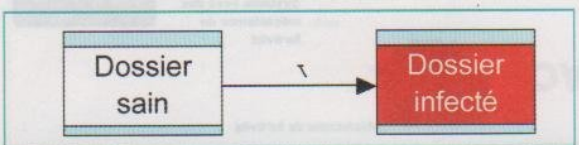


Figure 3 : Infection d'un dossier par un virus

Cet exemple peut être repris avec comme cible pour le virus, le gestionnaire de processus ou un autre gestionnaire d'objet du système.

La consultation d'un dossier du système de fichiers est régie de manière très simplifiée par la suite d'actions suivantes (voir Figure 4) :

1. Demande de lecture d'un fichier par un programme (ex : un détecteur).
2. Transmission de la requête au gestionnaire de fichiers.
3. Utilisation de l'appel système correspondant à la lecture d'un fichier (P_10).
4. Retour des informations lues au gestionnaire de fichier.
5. Transmission au programme des informations contenues dans le fichier.

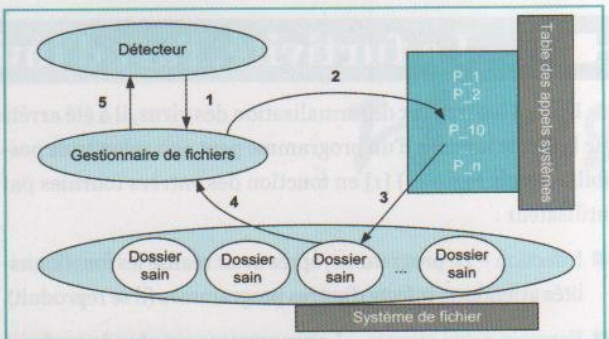


Figure 4 : Détection dans un cas normal

Dans le cas de la consultation d'un dossier infecté, le chemin d'exécution de la requête n'est pas le même. En effet, un élément du système, en l'occurrence, dans notre exemple, l'appel système P_10 a été modifié. La conséquence est que la réponse à la requête du détecteur fera apparaître un système sain malgré le dossier infecté (voir Figure 5).

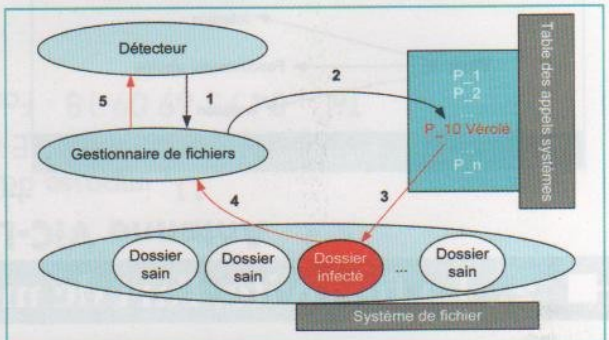


Figure 5 : Détection avec de la furtivité

3. La furtivité, l'art de se cacher

Dans ce climat de manque de formalisme général, il a été suggéré d'établir un parallèle entre la stéganographie et la furtivité [3]. Les deux concepts ayant comme objectif de dissimuler de l'information, il semble en effet judicieux de les comparer. Il a aussi été utilisé le concept de la « simulabilité des tests statistiques » pour proposer un aspect formel du problème lié à la détection de la furtivité [4].

3.1 Stéganographie et théorie de l'information

Un travail assez conséquent a été réalisé pour modéliser la stéganographie. Comme la comparaison est évidente entre furtivité et stéganographie, tout le travail effectué dans le domaine de la stéganographie a été réutilisé pour coller au domaine de la furtivité. Explicitons un peu plus la notion de stéganographie.

Stéganographie et Stéganalyse. *La stéganographie regroupe l'ensemble des techniques assurant non seulement la protection de l'information (aspect COMSEC), mais surtout la protection du canal de transmission de l'information (aspect TRANSEC). La stéganalyse regroupe l'ensemble des techniques permettant de détecter l'usage de la stéganographie et d'accéder à l'information stéganographiée.*

De cette définition, découle bien le distinguo à faire entre la dissimulation d'une part du code malveillant inactif et, d'autre part, des actions de ce code malveillant (en mémoire, sur le système de fichiers, etc.).

Je suis très émue de vous dire que j'ai bien compris, l'autre jour, que vous avez toujours une envie folle de me faire danser. Je garde un souvenir de votre baiser et je voudrais que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon affection toute désintéressée et sans calcul. Si vous voulez me voir ainsi dévoilée, sans aucun artifice mon âme toute nue, daignez donc me faire une visite. Et nous causerons en amis et en chemin. Je vous prouverai que je suis la femme sincère capable de vous offrir l'affection la plus profonde et la plus étroite amitié, en un mot, la meilleure amie que vous puissiez rêver. Puisque votre âme est libre, alors que l'abandon où je vis est bien long, bien dur et bien souvent pénible, ami très cher, j'ai le cœur gros, accourez vite et venez me le faire oublier. À l'amour, je veux me soumettre entièrement.

Votre poupée.

Figure 6 : Lettre avec de la stéganographie

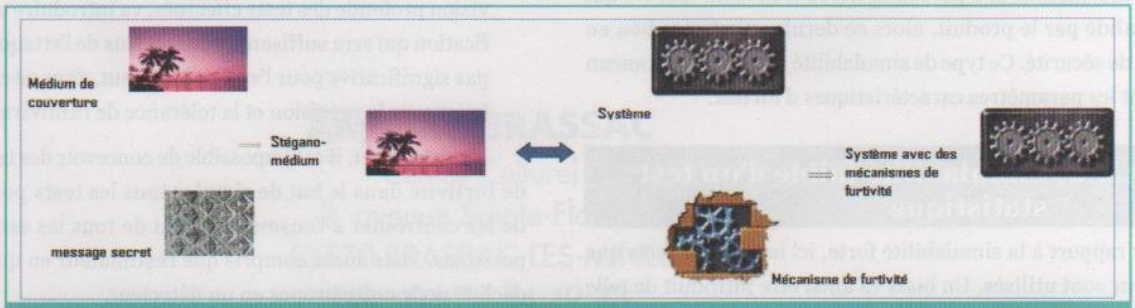


Figure 7 : Parallèle stéganographie – furtivité

La lettre suivante (voir Figure 6) est la plus connue pour illustrer la stéganographie. D'un effet romanesque au premier abord, elle se révèle d'une toute autre intention lorsque l'on ne considère qu'une ligne sur deux.

En reprenant la modélisation de Cachin et en l'adaptant au monde de la virologie informatique, on obtient ceci :

- Un médium de couverture désigne une donnée anodine dans laquelle un message secret peut être dissimulé (mais il ne l'est pas obligatoirement). Le médium de couverture correspond aux fichiers, structures et processus d'un système pouvant être utilisés par un code malveillant pour y dissimuler son code, ses données et ses actions.
- Le procédé de dissimulation est réalisé au moyen d'un algorithme de dissimulation lequel est dépendant ou non d'une clef secrète. Dans le cas de la furtivité classique, la clef est en général absente. Cet algorithme va cacher le message secret dans le médium de couverture. On obtient alors un stégano-médium.
- L'adversaire de la communication cherche, d'une part, à déterminer s'il existe des stégano-médiums, et, d'autre part, à accéder au message M à partir d'un modèle statistique décrivant la population des stégano-médiums.
- La population des stégano-médiums est modélisée par une distribution .

Le parallèle entre la furtivité et la stéganographie est donc bien établi (voir Figure 7).

Le dernier point nous montre que l'on peut définir le problème de la stéganographie comme un problème de tests d'hypothèses statistiques.

3.2 Simulabilité des tests statistiques

Les tests statistiques constituent un outil puissant et omniprésent. Les choix et les décisions prises font généralement suite à un ou plusieurs tests statistiques (élections, santé, etc.). Les tests sont utilisés du fait de l'incapacité de l'observateur et du décideur à acquérir une connaissance complète d'une population donnée. On va donc utiliser des sous-ensembles réduits de cette population, des échantillons. En vertu des lois de comportement de ces échantillons communément appelés « distribution d'échantillonnage », une décision pourra être prise concernant la population. Un test statistique est un protocole permettant de décider entre deux hypothèses : l'hypothèse nulle H_0 et l'hypothèse alternative H_1 (voir Figure 8, page suivante). L'outil principal utilisé est un estimateur. Selon l'une ou l'autre des hypothèses du test, cet estimateur est modélisé par une loi de probabilité différente. Un test d'hypothèse revient donc à décider en vertu des valeurs de l'estimateur sur un ou plusieurs échantillons, quelle est la loi qui gouverne la population que l'on étudie.

Dans la plupart des cas, H_1 est inconnue. Dans certains cas, elle est connue par un nombre limité de personnes, mais pas du testeur. Cette connaissance est à la base de la simulabilité des tests statistiques.

Deux types de simulabilité de tests existent :

- la simulabilité forte ;
- la simulabilité faible.

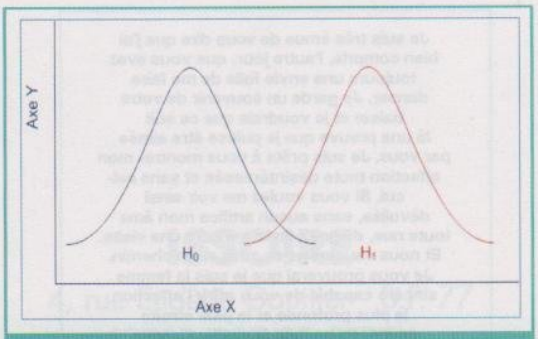


Figure 8 : Lois qui représentent H0 et H1

3.2.1 Simulabilité forte d'un test statistique

En termes de sécurité un produit est évalué bon par rapport à un nombre de tests précis validés. Simuler fortement ce produit veut dire que les réponses renvoyées face à tous ces tests sont conformes. Si une personne tierce trouve un test t+1 qui est invalidé par le produit, alors ce dernier n'est plus bon en termes de sécurité. Ce type de simulabilité ne considère à aucun moment les paramètres caractéristiques d'un test.

3.2.2 Simulabilité faible d'un test statistique

Par rapport à la simulabilité forte, ici les mêmes tests que le testeur sont utilisés. Un biais va ainsi être introduit de telle sorte que la sensibilité des tests ne peut le détecter.

Maintenant que la simulabilité des tests statistiques n'a plus de secret pour nous, voyons comment elle peut nous aider à formaliser la furtivité.

3.3 Nouvelle définition de la furtivité

Une nouvelle formalisation de la furtivité a été envisagée en considérant le concept de la simulabilité des tests. Par-dessus le risque intrinsèque d'erreur de décision pendant la détection antivirale, il existe un cas bien pire : le cas où l'attaquant utilise les techniques de détection contre le défendeur. Dès qu'une entité connaît les outils utilisés par l'autre, elle est capable de simuler les tests statistiques. Cette nouvelle approche met en relation la notion de « code furtif » et la notion de « détection ».

Définition : Simuler un test statistique consiste, pour un adversaire, à introduire, dans une population, un biais statistique qui ne peut pas être détecté par un analyste pendant le test. Il faut distinguer les deux types de simulabilité appliqués à la notion de furtivité :

- Dans le contexte de la détection de la furtivité, la simulabilité forte existe lorsque l'attaquant, qui a identifié une ou plusieurs techniques utilisées par l'antivirus, conçoit de nouvelles techniques de furtivité qui ne peuvent pas être détectées par l'antivirus.
- La simulabilité faible est le cas où l'attaquant, avec une vision profonde des tests effectués, va introduire une modification qui sera suffisante pour les fins de l'attaquant, mais pas significative pour l'antivirus. Il faut, dans ce cas précis, jouer avec la précision et la tolérance de l'antivirus.

Intuitivement, il est impossible de concevoir des techniques de furtivité dans le but de simuler tous les tests possibles et de les confronter à l'ensemble infini de tous les estimateurs possibles. Vous aurez compris que l'estimateur en question se décline pour notre propos en un détecteur.

3.4 Cas pratique

Prenons comme exemple deux photos d'un même paysage :

- l'une prise avec un appareil photo avec une résolution de 2 mégapixels ;
- l'autre prise avec un appareil d'une résolution de 10 mégapixels.

La comparaison des deux photos (voir Figure 9) avec un scanner d'une résolution inférieure conduira au postulat que les deux photos sont identiques. En revanche, une résolution du scanner plus importante montrera une différence entre les deux photos (pouvant servir à cacher des informations). Tout dépend donc de la sensibilité du détecteur.

4. La furtivité, l'art de s'immerger dans les profondeurs

Dans cette partie, on propose de s'intéresser à un formalisme qui englobe toutes les facettes possibles relatives aux mécanismes de furtivité [5]. Cela suppose de prendre en compte l'emploi de la furtivité aussi bien de manière bienveillante que malveillante. Cela implique aussi de prendre en compte le rapport de force existant avec les détecteurs. Enfin, il faut pouvoir proposer une granularité du système impacté beaucoup plus pertinente que dans les précédentes parties.

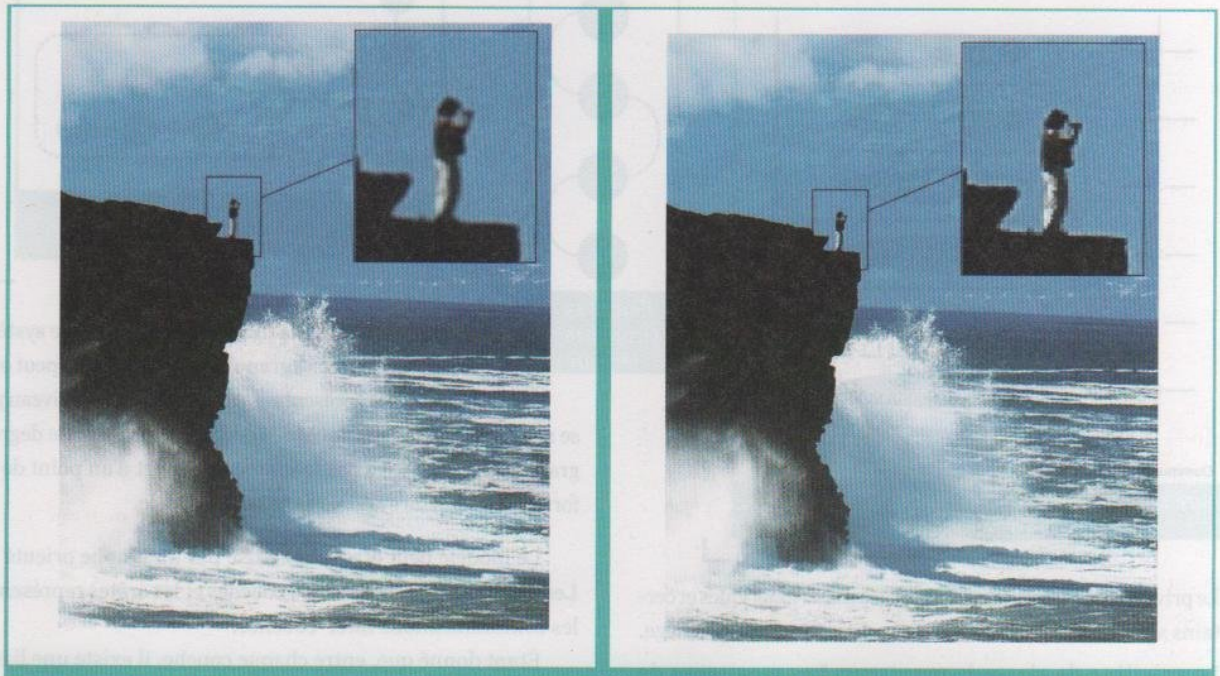


Figure 9 : Distinguabilité entre deux photos

4.1 Représentation du système

Traditionnellement, un système informatique se représente comme un empilement de couches. Chacune d'entre elles s'appuie sur une couche inférieure jusqu'à atteindre la couche la plus basse (généralement la couche matérielle). Plus la couche sur laquelle s'appuie le composant est profonde, plus le niveau de privilège utilisé est élevé. Le privilège est une notion très répandue dans les systèmes informatiques. Cette notion permet d'établir une hiérarchie pour les différents programmes du système. Le principe est de séparer les requêtes en fonction de leur importance, de leur criticité (voir Figure 10).

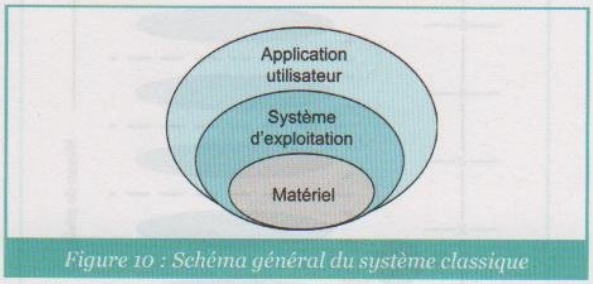


Figure 10 : Schéma général du système classique

Ce schéma conventionnel ne suffit plus pour représenter un système qui a fortement évolué. En effet, avec l'émergence des machines virtuelles, une couche supplémentaire doit être rajoutée. Elle correspond au moniteur virtuel appelé encore « système hyperviseur » qui va contrôler les systèmes d'exploitation virtuels des machines virtuelles. Voici donc le schéma que nous proposons, pour englober les récentes évolutions en complexité du système (voir Figure 11).

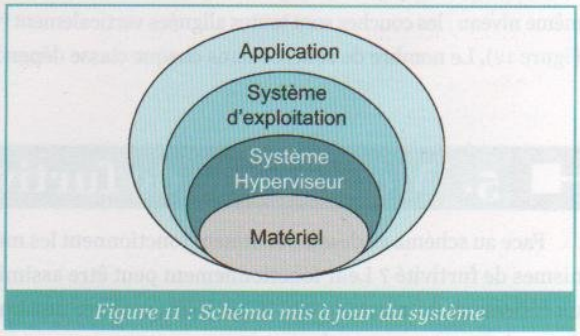


Figure 11 : Schéma mis à jour du système

En s'appuyant sur ce schéma, on peut désormais introduire les concepts suivants :

Un système est un ensemble constitué de couches. Les couches contiguës peuvent communiquer entre elles. Ces relations entre les couches sont représentées par des Communications inter-couches. On peut classifier ces couches en différentes catégories en fonction du niveau de privilège des programmes s'exécutant au sein de la couche. Cette classification est donc dépendante de la classe de privilège que l'on peut regrouper en quatre grandes classes :

- Le privilège de classe utilisateur. Une application utilisateur a ses programmes qui s'exécutent avec ce niveau de privilège.

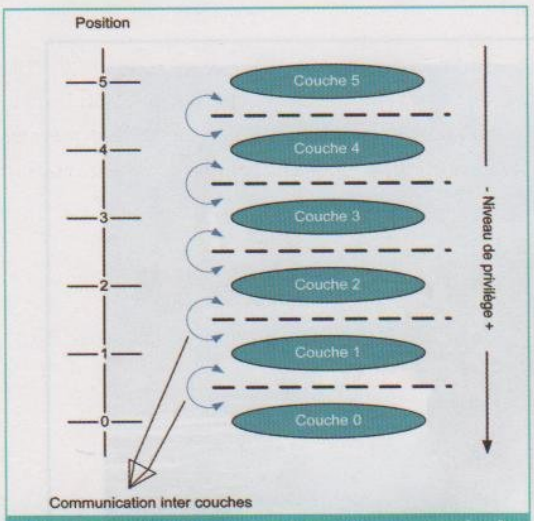


Figure 12 : Représentation du système en couche de notre modèle

- Le privilège de classe noyau. Les pilotes de périphériques et certains services du système ont besoin de ce niveau de privilège.
- Le privilège de classe hyperviseur. Les moniteurs de machines virtuelles ont des programmes qui s'exécutent dans cette classe de privilège.
- Le privilège de classe matériel. Les codes qui utilisent directement le matériel sont dans cette classe de privilège.

Afin de diminuer la complexité d'un tel schéma, il sera pris en considération qu'il ne peut y avoir plusieurs couches sur un même niveau : les couches sont toutes alignées verticalement (voir Figure 12). Le nombre de couches dans chaque classe dépend de la granularité à laquelle on veut étudier le système. En utilisant une granularité plus fine, on peut obtenir une représentation dans laquelle un niveau peut se subdiviser en un ensemble de couches. Cependant, le degré de granularité ne change pas fondamentalement d'un point de vue formel la vision du système dans notre modèle.

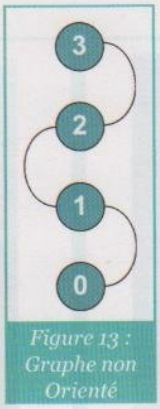


Figure 13 : Graphe non Orienté

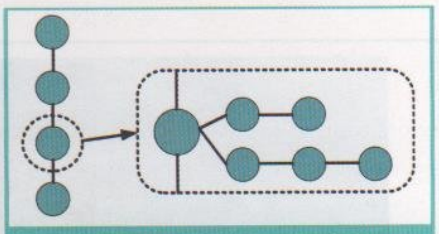


Figure 14 : Développement de la granularité d'un sommet

Le modèle peut être schématisé par un graphe orienté [6]. Les sommets représentent les couches et les arêtes représentent les communications inter-couches.

Étant donné que, entre chaque couche, il existe une liaison bidirectionnelle, on peut de ce fait simplifier notre modèle en utilisant un graphe simple non orienté (voir Figure 13).

Bien entendu, la vision de chaque sommet du graphe est dépendante de la granularité à laquelle on considère un système. Un sommet peut révéler une arborescence ayant une structure d'arbre (voir Figure 14).

À travers cette représentation, ressort la vision complexe du système sur lequel le mécanisme de furtivité doit s'installer.

5. Mécanisme de furtivité dans le schéma proposé

Face au schéma ci-dessus, comment fonctionnent les mécanismes de furtivité ? Leur fonctionnement peut être assimilé à un réflecteur, à un miroir, qui va toujours projeter une image du système qui est normale.

Dans le cas du camouflage, le principe est de modifier l'environnement, mais de telles façons que le détecteur ne puisse pas s'en rendre compte (COMSEC en stéganographie). Dans le cas de la furtivité totale, le principe est de modifier autant l'environnement que les communications entre les programmes (TRANSEC en stéganographie).

Le miroir est positionné entre deux couches contiguës, et toutes les couches au-dessus du « miroir » verront une image du système normal. Le canal de communication inter-couches entre ces deux couches est intercepté et modifié. Dans le cadre d'un code malveillant, l'attaquant peut alors modifier un certain nombre de données en dessous de ce miroir (voir Figure 15).

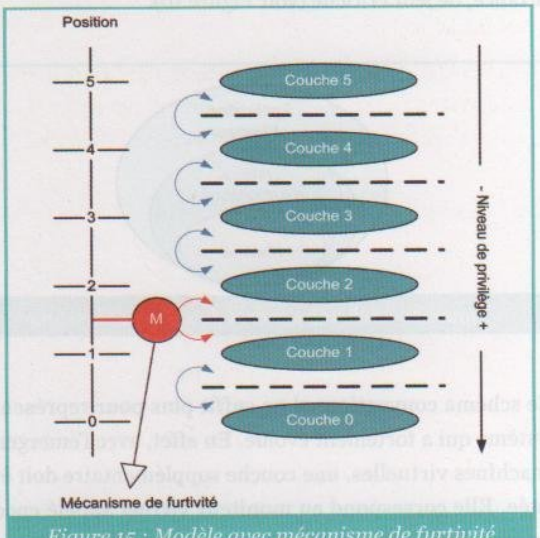


Figure 15 : Modèle avec mécanisme de furtivité

Le niveau de l'interception détermine la classe du mécanisme de furtivité. Tout comme les privilèges, ils sont répartis en quatre grandes catégories :

- le mécanisme de furtivité de classe utilisateur ;
- le mécanisme de furtivité de classe noyau ;
- le mécanisme de furtivité de classe hyperviseur ;
- le mécanisme de furtivité de classe matériel.

Le formalisme distingue bien les deux aspects de la furtivité (camouflage et furtivité totale). Cependant, la réalité de la furtivité dépend du détecteur. Intéressons-nous désormais au problème de la position du détecteur par rapport au mécanisme de furtivité.

6. Relation détecteur – mécanisme de furtivité

Il a été évoqué plus haut qu'il y a une importante relation qui lie le mécanisme de furtivité au détecteur. Le concept de furtivité relative[3] est ainsi introduit. Un mécanisme de furtivité est toujours qualifié comme tel en fonction des caractéristiques du détecteur existant sur le système. La furtivité absolue existe si l'ensemble infini des détecteurs est simulable par le mécanisme de furtivité. Or, ceci est théoriquement impossible. Il n'y a pas et il n'y aura jamais de furtivité absolue (100% indétectable) contrairement à ce que peuvent affirmer certains [7, 8].

6.1 Position du détecteur face aux mécanismes de furtivité

Une première condition sur le détecteur est nécessaire pour parfaire le formalisme. Cette condition est fonction de la position du détecteur. Cela revient à dire que si le détecteur est positionné en dessous du mécanisme de furtivité, alors les modifications apportées au système lui sont probablement visibles (voir paragraphe suivant). Sinon, le détecteur ne voit qu'une image normale du système (voir Figure 16).

Le cas problématique de l'égalité entre la position du détecteur et du mécanisme de furtivité restera flou. Le défenseur peut se dire qu'il existera toujours le moyen de rechercher plus bas la présence de code furtif, mais il faut prendre en compte le cas où la position du mécanisme de furtivité et la position du détecteur est la plus basse possible.

Comment faire alors pour garantir malgré tout une protection du système dans cette situation ? Nous donnons une esquisse de solution à ce problème dans les paragraphes suivants.

6.2 Sensibilité et efficacité du détecteur

La position du détecteur est une condition nécessaire, mais pas suffisante. Il faut en effet prendre en compte le cas où le détecteur se situe à un niveau inférieur par rapport à un mécanisme de furtivité et que, malgré cela, ce dernier arrive à simuler un système sain face aux différents tests lancés par le détecteur.

Il est toujours question de la relativité du mécanisme de furtivité par rapport au détecteur, mais en tenant compte des caractéristiques intrinsèques de ce dernier.

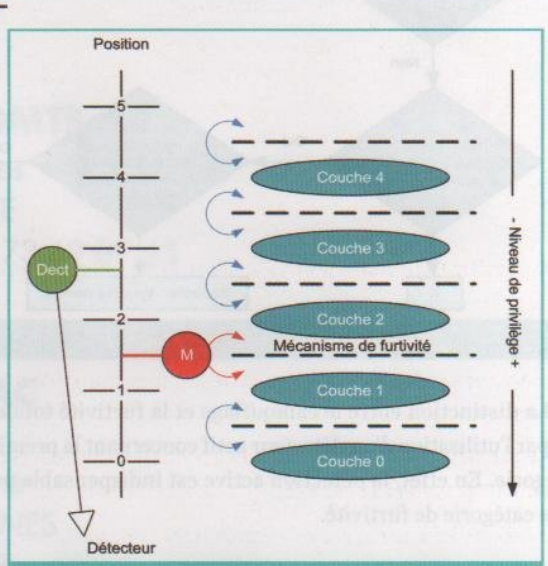


Figure 16 : Détecteur versus mécanismes de furtivité

Le détecteur peut agir de différentes manières, mais on distingue bien deux catégories fonctionnelles :

- Détecteur passif : entité qui est en écoute sur tout ou partie des activités du système. (type de détecteur utile pour la détection des mécanismes de furtivité de type total).
- Détecteur actif : détecteur passif qui, en plus, recherche dans le système les objets illégitimes. (type de détecteur utile pour détecter les deux types de furtivité).

On distingue deux cas de simulabilité du système sain :

- La simulabilité forte : le mécanisme de furtivité se situe à un niveau inférieur par rapport au détecteur. Il peut donc répondre correctement aux différents tests du détecteur.
- La simulabilité faible : le mécanisme de furtivité se situe à un niveau supérieur par rapport au détecteur. Il simule des réponses de façon à ce que la distribution de ces dernières soit statistiquement indistinguable de la distribution des réponses obtenues par un détecteur sur un système sain.

On peut aussi introduire la notion d'« indistinguabilité calculatoire ». De façon informelle, il s'agit de ne pas pouvoir distinguer les deux distributions à l'aide d'un algorithme polynomial.

6.3 Formalisme général des mécanismes de furtivité

On obtient ainsi le diagramme suivant :

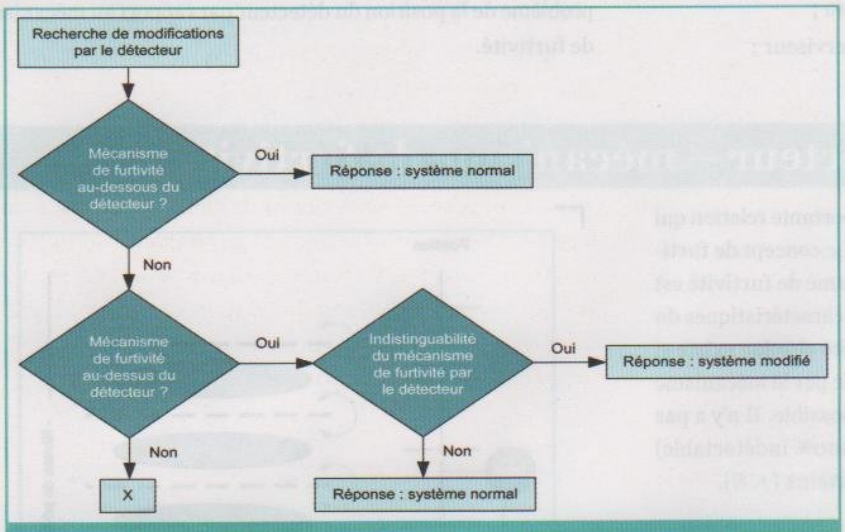


Figure 17

La distinction entre le camouflage et la furtivité totale se fait par l'utilisation d'un détecteur actif concernant la première catégorie. En effet, la détection active est indispensable pour cette catégorie de furtivité.

6.4 Traitement du cas particulier X

Le modèle évoqué ci-dessus n'est pas complet pour le cas où le mécanisme de furtivité est positionné sur le même sommet que le détecteur. Malheureusement, ce cas restera flou en raison de la méconnaissance de la réaction du système face à deux modifications sur le même sommet.

Toutefois, concernant la sécurité des systèmes, on peut mettre en avant la solution suivante.

Comme décrit dans le paragraphe précédent, si le détecteur n'est pas simulable, alors il est théoriquement impossible pour un attaquant de concevoir un code pouvant le leurrer.

De ce fait, la solution au problème de l'égalité peut être envisagée par l'utilisation de technologies de furtivité dans les détecteurs (notion de « détecteurs prophylactiques »[4]). Ainsi, ces derniers ne seraient plus simulables par les attaquants.

7. Problème de détection et d'éradication des mécanismes de furtivité

Grâce au formalisme précédent et au modèle en graphe proposé, le problème de la détection des mécanismes de furtivité est plus abordable.

La furtivité totale modifie autant les arêtes (la nature des communications) que les sommets du graphe alors que le camouflage ne modifie que les sommets. La présence d'un mécanisme de furtivité correspond à la présence d'un sommet supplémentaire ou modifié et/ou la présence ou modification également d'arêtes incidentes à ce sommet. Ceci ne dépend exclusivement que du degré de granularité avec lequel a été construit le graphe représentant le système sain. Les différents cas à prendre en compte sont schématisés ci-dessous.

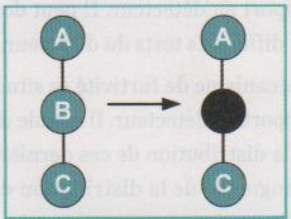


Schéma 1 : Le système furtif s'insère – sommet modifié (système furtif s'insère au sein d'une couche)

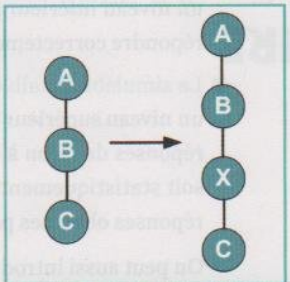


Schéma 2 : Ajout de sommet

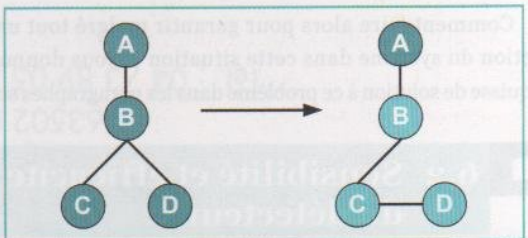


Schéma 3 : Modification d'arêtes

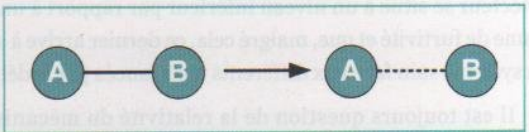


Schéma 4 : Modification de la nature de la communication d'une arête.

7.1 Détection du mécanisme de furtivité

Dès lors que l'on parle de modification, on parle toujours de comparaison d'un état de référence avec un état courant. La comparaison requiert alors la sauvegarde de l'état de référence ou du moins la connaissance de ses caractéristiques principales.

La sauvegarde de l'état de référence du système consiste à sauvegarder le graphe associé et la valuation des sommets et des arêtes. Le lecteur comprendra que l'efficacité du modèle repose sur l'extraction d'un modèle de référence pertinent. De plus, ce modèle doit être mis à jour en fonction des modifications légitimes apportées au système. La pertinence de ce graphe dépend de la politique de sécurité et de sauvegarde mis en place.

7.1.1 Détection dans le cas du schéma 1

Le schéma 1 représente la modification d'un sommet dans le graphe. Cette modification entraîne forcément un changement de la valuation de ce sommet.

Par exemple, cette valuation peut être calculée par une fonction de hash sur la taille en octets des sommets.

7.1.2 Détection dans le cas du schéma 2

Le schéma 2 représente le cas d'un ajout de sommet dans le graphe. Ce cas de figure implique obligatoirement une modification de la cardinalité du graphe. On a donc la relation suivante qui est vérifiée :

7.1.3 Détection dans le cas du schéma 3

Le schéma 3 nous montre le cas où le chemin menant à un sommet est modifié. Cela se traduit en particulier par la suppression et l'ajout d'une arête. Ce qui signifie que certains sommets vont voir leur degré modifié.

La détection se résume alors en un parcours des sommets du graphe de référence et à comparer les degrés de ses sommets avec ceux des sommets du graphe courant.

Remarque : Cette détection peut s'effectuer aussi, en comparant les valuations des arêtes et des sommets, ce qui est plus coûteux en temps.

7.1.4 Détection dans le cas du schéma 4

Le schéma 4 met en exergue l'utilisation frauduleuse d'une communication au sein du système. Une arête aura donc été détournée pour permettre d'autres types de communication que celle habituellement autorisée.

7.2 Éradication des mécanismes de furtivité

La détection étant démontrée comme un problème abordable en suivant notre modèle théorique[5], intéressons-nous au problème beaucoup plus délicat de l'éradication.

Pour bien comprendre de quoi il en retourne, il faut garder en mémoire que les mécanismes de furtivité, dans la grande majorité des cas, ne se contentent pas d'ajouter du code dans le système. Le système en ressort plus ou moins altéré. Ainsi, la suppression du code binaire lié au mécanisme de furtivité ne permet pas de retrouver un système opérationnel. Ceci implique une contrainte forte qui est la nécessité de reconstruire le système.

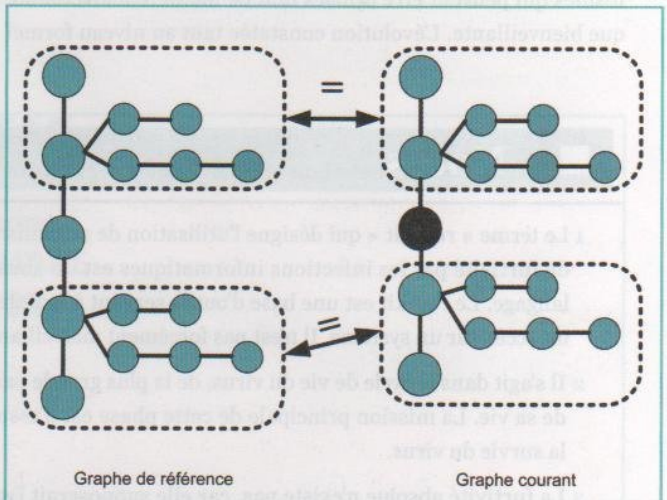


Figure 18 : Détection et éradication d'un sommet modifié

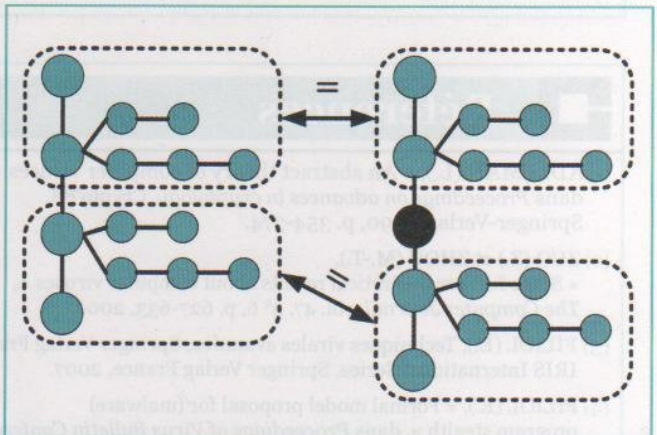


Figure 19 : Détection et éradication d'un sommet ajouté

Une reconstruction du système implique une connaissance de ce dernier. Dans notre modèle, la connaissance nécessaire est acquise dans le graphe de référence. Donc, toute la réussite du processus d'éradication dépend de la pertinence de l'extraction de ce graphe (Indistinguabilité statistique ou calculatoire).

Pour tous les cas de figure évoqués plus haut, le principe est identique et élémentaire. On doit rechercher les sommets du système courant qui diffèrent avec les sommets du système de référence et recopier les sommets originels le cas échéant.

La même technique s'applique aux arêtes (voir Figure 18).

Une fois ces portions identifiées, il ne reste plus qu'à réaliser le remplacement sur les autres.

Bien entendu, le cas où il y a un simple rajout de sommet sans modification des sommets adjacents (« *man in the middle* ») implique la suppression de ce dernier (voir Figure 19).

Conclusion

Comme nous avons pu le voir, un certain nombre de travaux intéressants ont été réalisés pour formaliser les mécanismes de furtivité. Les définitions qui ont été données nous donnent une meilleure vision de l'avantage et des inconvénients de ces mécanismes qui peuvent être utilisés tant de manière malveillante que bienveillante. L'évolution constatée tant au niveau formel que technique conduit tout naturellement à repenser la manière de protéger le système, notamment en utilisant ces mécanismes de furtivité pour lutter contre les codes malveillants. Nous conclurons le débat sur la furtivité en soutenant ardemment que tout concept de protection utilisant la furtivité se révélera plus efficace qu'un mécanisme de protection classique. ■

Notes

1 Le terme « rootkit » qui désigne l'utilisation de mécanismes de furtivité par les infections informatiques est un abus de langage. Le rootkit est une base d'outils servant à maintenir un accès sur un système. Il n'est pas forcément malveillant.

2 Il s'agit dans le cycle de vie du virus, de la plus grande partie de sa vie. La mission principale de cette phase est d'assurer la survie du virus.

3 La furtivité absolue n'existe pas, car elle supposerait l'existence de programmes qui seraient 100% indétectables sans aucune limite de temps et de moyen.

4 Le terme « prophylactique » provient du domaine médical. Il représente la faculté d'empêcher l'infection. Un détecteur prophylactique est capable d'empêcher la compromission des couches égales ou supérieures à sa position.

5 Ce problème est plus compliqué en pratique lorsque l'on prend en compte la complexité de la construction du graphe de référence (l'extraction de la distribution et la détection dans des graphes plus élaborés demandant un niveau de granularité dans le système extrêmement profond).

Références

[1] ADLEMAN (L.), « An abstract theory of computer viruses », dans *Proceedings on advances in cryptology*, Crypto'88, Springer-Verlag, 1990, p. 354-374.

[2] ZUO (Z.) et ZHOU (M.-T.), « Some further theoretical results about computer viruses », *The Computer Journal*, vol. 47, n° 6, p. 627-633, 2004.

[3] FILIOL (E.), Techniques virales avancées, Springer Verlag France, IRIS International Series, Springer Verlag France, 2007.

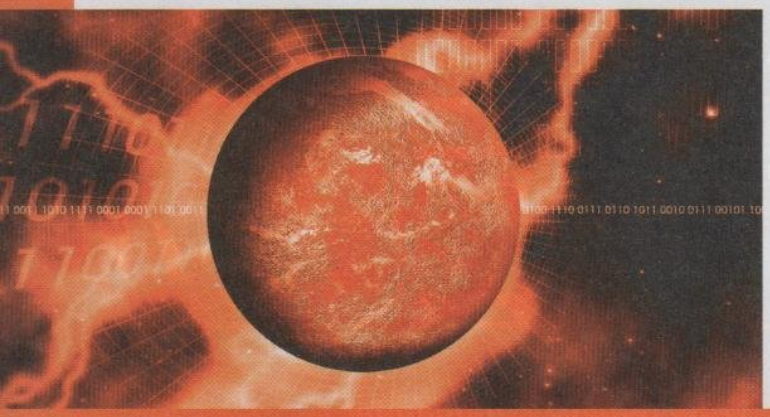
[4] FILIOL (E.), « Formal model proposal for (malware) program stealth », dans *Proceedings of Virus Bulletin Conference*, VB2007, 2007.

[5] DEROCK (A.), VERON (P.), « Another Formal Proposal for Stealth », dans *Proceedings of International Conference in Security and Safety of Complex System*, SSCS08, 2008.

[6] BERGE (C.), Théorie des graphes et ses applications, Dunod, 1958.

[7] RUTKOWSKA (J.), Introducing Stealth Malware Taxonomy, dans *Black Hat Conference*, 2006.

[8] FERRIE (P.) et al. : http://pferrie.tripod.com/papers/vtrootkits.pdf



Axelle Apvrille
axelle.apvrille@miscmag.com

LA SÉCURITÉ DES WIKIS

mots-clés : wiki / authentification / chiffrement / web

Pour peu qu'on y fasse attention, il n'y a pas de doute, ces 5 dernières années, nos habitudes web ont quelque peu changé :

- *édition des sites web à l'aide de wikis, CMS (Content Management System), blogs ;*
- *pages web plus configurables grâce à la mouvance du Web 2.0 ;*
- *messageries instantanées à tout va, que ce soit depuis le pop-up d'un PC ou d'un téléphone portable.*

Bientôt, nous passerons pour des dinosaures auprès de nos enfants : « dis, papa, maman, c'est vrai que tu modifiais ton site web *à la main* ? » La puissance des PC, la démocratisation de l'accès à Internet et notamment le haut débit ont grandement contribué à ces changements. Il n'y a pas à dire, le web, c'est plus facile maintenant qu'avant. Il subsiste cependant en arrière-pensée une petite inquiétude : « oui, mais... », une sorte de déformation professionnelle de l'ingénieur sécurité : toutes ces nouveautés sont-elles sûres ? Ont-elles intégré la notion de sécurité informatique ? Cette inquiétude est d'autant plus fondée que les systèmes mis en œuvre sont plus complexes : savoir qui prend en charge quelles données n'est pas forcément évident, à la différence des vieilles méthodes « j'édite dans Emacs (ou vi – inutile de froisser quiconque) et j'*uploade* par FTP ». Cet article tente d'éclaircir la question pour un *wiki* appelé DokuWiki [1] : tout d'abord, le processus d'authentification de DokuWiki, puis la sécurisation du contenu du wiki.

1. Qu'est-ce qu'un wiki et pourquoi DokuWiki en particulier ?

Les wikis sont des logiciels intégrés à certains sites web et qui sont destinés à en faciliter l'édition : par exemple, l'édition peut se faire directement depuis un navigateur web, sans avoir à connaître les balises de l'HTML.

Quand on voit la pléthore de wikis existants (mediawiki, moinmoin, pmwiki, xwiki... [2]), un seul article ne peut prétendre à en faire le tour du point de vue sécurité. J'ai choisi DokuWiki parce que c'est un outil que j'aime bien : son installation et son utilisation sont simples. Il n'y a pas besoin de base MySQL, son apparence par défaut n'est pas tape-à-l'œil, et le tout est configurable à souhait. J'ai donc commencé à en étudier la sécurité essentiellement en espérant me rassurer (non, je n'en dis pas plus, il faut laisser le suspens).

2. L'authentification sous DokuWiki

Sous DokuWiki, certaines actions ou contenus peuvent être sujets à une authentification de la part de l'utilisateur (bouton « Login »).

2.1 Configuration de la méthode d'authentification

La méthode d'authentification en elle-même est configurable soit directement en modifiant les fichiers en question (`conf/dokuwiki.php`), soit graphiquement, depuis un navigateur, par le biais du *plugin* de configuration installé par défaut avec DokuWiki. Il suffit alors de se loguer en tant qu'administrateur DokuWiki, puis d'appuyer sur le bouton « Admin » qui apparaît alors en bas de chaque page, et enfin « Configuration settings ». Tous les paramètres concernant l'authentification sont groupés (voir Figure 1) : il suffit de sélectionner/cocher/remplir les champs qui nous intéressent.

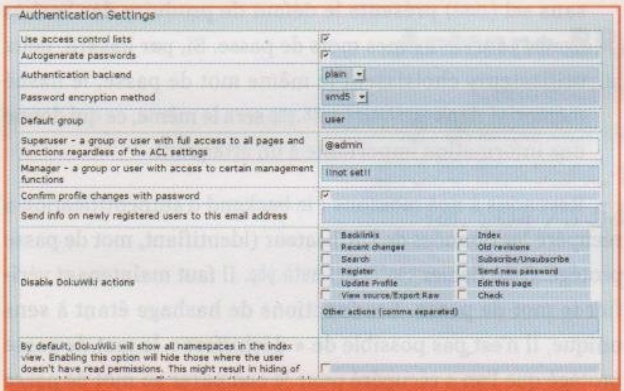


Figure 1 : Paramètres d'authentification, dans le plugin de configuration de DokuWiki

2.2 Envoi du login

L'authentification de l'utilisateur se fait dans un « bête » formulaire HTML – formulaire configurable au sein du fichier `inc/html.php` (voir Figure 2).

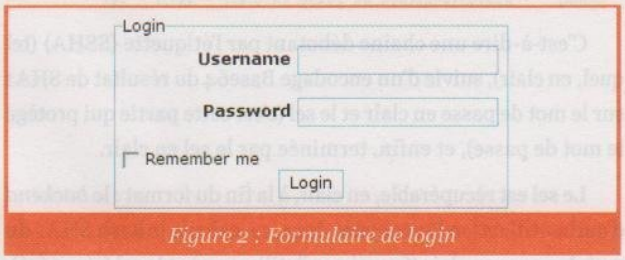


Figure 2 : Formulaire de login

Dès que l'utilisateur appuie sur le bouton « Login » (après avoir saisi son identifiant et mot de passe), le navigateur génère une requête HTTP POST qu'il envoie au serveur web. C'est là ma première déception : une écoute sur le réseau révèle l'identifiant et surtout le mot de passe en clair... (voir Figure 3).

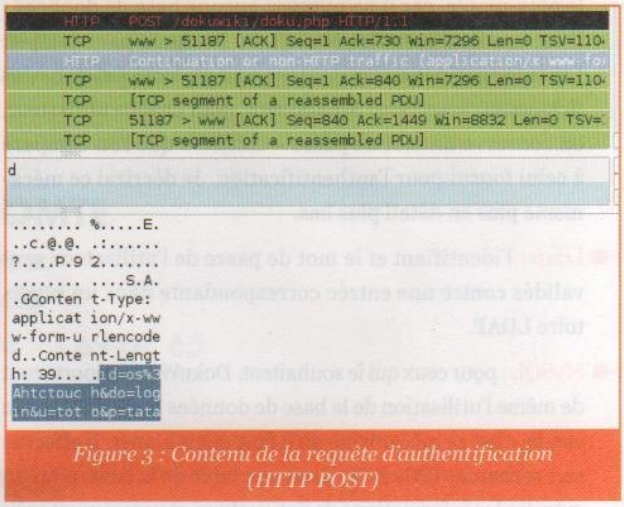


Figure 3 : Contenu de la requête d'authentification (HTTP POST)

La requête HTTP POST envoie quatre champs en clair :

- `id` : le nom de la page actuelle, pour laquelle on s'authentifie (exemple : `os :htctouch`).
- `do` : l'action réalisée. Ici, il s'agit d'une authentification, donc `login`.
- `u` : l'identifiant de l'utilisateur (exemple `toto`).
- `p` : le mot de passe de l'utilisateur (exemple `tata`).

Bien sûr, il est possible d'installer DokuWiki sur un serveur supportant SSL/TLS. Cela pallie effectivement le fait que les champs soient envoyés en clair, mais quelle usine à gaz pour si peu : il suffisait d'utiliser une fonction de *hashage* pour masquer le mot de passe. Par exemple, le serveur aurait pu envoyer un challenge (pour éviter le rejeu) que le client aurait hashé avec son mot de passe. Quel dommage !

2.3 Validation du mot de passe sur le serveur

Du côté du serveur web, DokuWiki doit maintenant valider les données qui lui ont été envoyées, c'est-à-dire valider que `toto` est un utilisateur valide et que son mot de passe est bel et bien `tata`. Sachant que le mot de passe est envoyé en clair, on peut craindre le « pire », à savoir que le mot de passe soit mémorisé en clair sur le serveur. Je vous rassure tout de suite : ouf, il n'en est rien (tout de même !).

DokuWiki propose plusieurs mécanismes pour valider l'identifiant et le mot de passe qu'il reçoit. Ce mécanisme est paramétrable (voir Figure 1) par le champ `authentication backend` ou, pour les férus d'édition de fichiers, dans `conf/dokuwiki.php`, le paramètre `authtype`. Il définit comment l'utilisateur est authentifié et, par la même occasion, où sont mémorisées les informations utiles à l'authentification. Les choix possibles sont :

- Plain : il s'agit du mécanisme par défaut. C'est le mécanisme le plus simple, car il ne requiert aucune base de données : il est inutile d'installer MySQL ou Postgres par exemple. Les informations de l'utilisateur sont mémorisées dans des fichiers sur le serveur. Notamment, pour chaque utilisateur, on conserve un mot de passe « chiffré » que l'on compare à celui fourni pour l'authentification. Je décrirai ce mécanisme plus en détail plus bas.
- Ldap : l'identifiant et le mot de passe de l'utilisateur sont validés contre une entrée correspondante dans un répertoire LDAP.
- MySQL : pour ceux qui le souhaitent, DokuWiki supporte tout de même l'utilisation de la base de données MySQL. Dans ce cas-là, c'est ce mécanisme qu'il faut sélectionner. Il effectue une recherche (SELECT) dans une table de la base, pour en extraire les informations de l'utilisateur, et notamment valider le mot de passe fourni lors de l'authentification contre celui mémorisé dans la table. Les commandes SQL à utiliser, le nom des tables ainsi que d'autres options sont paramétrables dans le fichier de configuration `conf/mysql.conf.php`.
- PgSQL : ce mécanisme est similaire au *backend* MySQL, si ce n'est qu'il est adapté aux bases de données Postgres.
- Punbb : l'identifiant et le mot de passe de l'utilisateur sont validés contre les informations mémorisées dans un cookie PunBB.

Dans tous les cas, l'identifiant et le mot de passe à valider sont fournis en clair à la procédure d'authentification (hélas pour le mot de passe...), mais le mot de passe est conservé de manière sécurisée (chiffré ou hashé – cela dépend du paramétrage...) dans une localisation propre au mécanisme (dans des fichiers texte pour « plain », dans un répertoire LDAP pour « ldap », etc.).

Si l'on a sélectionné le mécanisme d'authentification par défaut, plain, l'authentification de l'utilisateur se passe de la manière suivante. D'abord, le *backend* d'authentification récupère les données de l'utilisateur dans le fichier `conf/users.auth.php`. Ce fichier, de manière similaire à un `/etc/passwd` sous Unix, contient pour chaque utilisateur son nom complet, le hash de son mot de passe (et non – fort heureusement – le mot de passe en clair), son e-mail et les groupes auxquels il appartient. Par exemple :

```
axelle:343e4632a6278bdc0550859ec34d661e728a1d7e:Axelle
Apvrille:nospam@nospam.com:admin,user
pico:a3c83f81b03e2ad8598b29910ea4689d2f60e6db:Pico:grande@gueule.
com:user
```

L'algorithme utilisé pour protéger le mot de passe est paramétrable (voir Figure 1, champ `password encryption method` ou, dans le fichier `conf/dokuwiki.php`, le paramètre `passcrypt`). On peut choisir entre l'utilisation directe des fonctions de hashage SHA1 ou MD5, coupler les fonctions de hashage à un sel, utiliser des fonctions propres à MySQL ou enfin la fonction `crypt()` d'Unix. La meilleure solution – ou plutôt la moins mauvaise – me semble être SHA1 avec sel (« ssha »), car MD5 est tout sauf recommandable [3, 4]. SHA1 n'est peut-être pas d'une longévité assurée [5], mais il n'est au moins pas « cassé » à l'heure actuelle.

- Les fonctions `mysql` (fonction `OLD_PASSWORD`) et `my411` (fonction `PASSWORD` de MySQL >=4.1.1) sont, d'après la documentation de MySQL [6], à réserver à la protection des mots de passe MySQL et non de tierces applications.
- La fonction `crypt` est quelque peu dépassée.
- Enfin, l'utilisation de la fonction de hashage SHA1 en direct sans sel (`sha1`) présente le défaut de produire des hashés égaux pour de mêmes mots de passe. Si, par hasard, deux utilisateurs choisissent le même mot de passe, le hashé mémorisé dans `conf/users.auth.php` sera le même, ce qui donne une information importante à un attaquant.

Revenons à nos moutons : le backend d'authentification a récupéré les données de l'utilisateur (identifiant, mot de passe protégé...) du fichier `conf/users.auth.php`. Il faut maintenant vérifier le mot de passe. Les fonctions de hashage étant à sens unique, il n'est pas possible de « déchiffrer » le mot de passe protégé que l'on a récupéré pour le comparer au mot de passe en clair fourni par l'utilisateur. Plutôt, il faut le recalculer et comparer, c'est-à-dire appliquer la fonction de hashage sur le mot de passe en clair et comparer le résultat avec le mot de passe protégé mémorisé pour l'utilisateur. Si on utilise une fonction avec un sel, cela se complique, car il faut au préalable récupérer le sel. Ce sel est mémorisé au sein du mot de passe protégé.

Par exemple, à la figure 4, page suivante, lorsqu'on utilise SHA1 avec un sel, le mot de passe protégé utilise le format suivant :

```
"{SSHA}"  + Base64(sha1(mot de passe en clair + sel)) + sel
```

C'est-à-dire une chaîne débutant par l'étiquette {SSHA} (tel quel, en clair), suivie d'un encodage Base64 du résultat de SHA1 sur le mot de passe en clair et le sel (c'est cette partie qui protège le mot de passe), et enfin, terminée par le sel en clair.

Le sel est récupérable, en clair, à la fin du format : le *backend* d'authentification le récupère donc, re-calcule le hash SHA1 du mot de passe en clair (fourni par l'utilisateur) et le sel (récupéré),

l'encode en Base64 et compare ce résultat avec le milieu du mot de passe protégé (NB : il est aussi possible de décoder l'encodage base64 du mot de passe protégé et de comparer directement les deux hashs, c'est exactement pareil). Si les deux résultats sont identiques, l'authentification peut être validée. Sinon, elle doit être refusée.

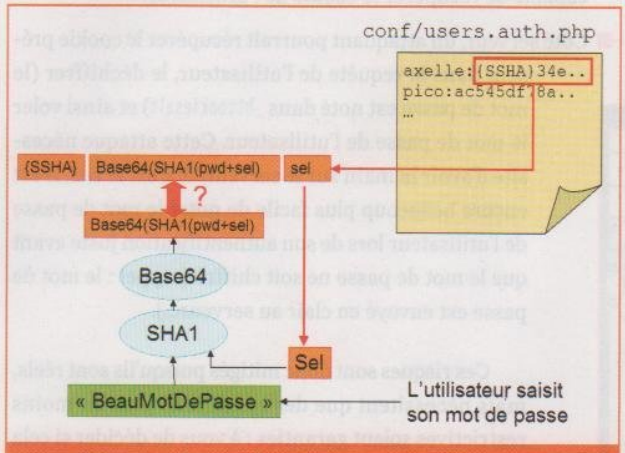


Figure 4 : Étapes de sécurisation du mot de passe

Ce procédé d'authentification est simple et efficace. Il est souvent utilisé en cryptographie.

2.4 Contrôle d'accès aux pages protégées

Toute page de DokuWiki peut être protégée : l'administrateur peut restreindre la lecture (ou l'édition, la création, la suppression, le téléchargement d'images) de certaines pages à certains utilisateurs. Cela se configure soit graphiquement via l'interface de gestion des contrôles d'accès dans la page d'administration – ce qui est la méthode recommandée par DokuWiki – soit directement dans le fichier `conf/acl.auth.php`.

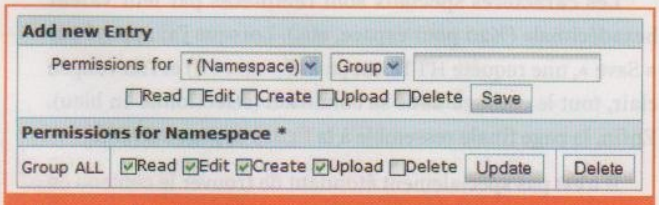


Figure 5 : Interface de gestion des contrôles d'accès (dans le cas présenté, toutes les actions sont permises sur cette page, sauf la suppression).

Chaque fois qu'un utilisateur tente d'effectuer une action protégée, DokuWiki vérifie s'il en a les droits. En particulier, si l'action est restreinte à un utilisateur donné (« seule Axelle a le droit d'effacer cette page »), il faut vérifier l'identité de l'utilisateur (êtes-vous bien Axelle ?). Cependant, du point de vue de l'utilisateur, cela signifierait que l'utilisateur doit saisir son identifiant et mot de passe à chaque action privilégiée, ce qui serait certainement bien trop fastidieux. Par conséquent, pour éviter cela, DokuWiki ne requiert qu'une seule vraie authentification de la part de l'utilisateur, mémorise toutes les informations utiles dans un cookie et tente d'effectuer une sorte d'authentification automatique toutes les prochaines fois. Bien entendu, ce procédé présente toujours un certain risque et donc les informations de l'utilisateur ne sont considérées valides pour une authentification automatique que pendant un certain laps de temps. Lorsque ce dernier est écoulé, une nouvelle « vraie » authentification est nécessaire.

À l'issue d'une « vraie » authentification (la façon de s'authentifier a été détaillée précédemment), DokuWiki crée donc un cookie sur la machine de l'utilisateur sous le nom de DW+quelquechose, où le quelque chose est le hashé de l'URL à laquelle le cookie correspond (voir Figure 6). Ce cookie contient, encodé en Base64, le nom de l'utilisateur et son mot de passe chiffré par l'algorithme symétrique Blowfish [7]. La clé de chiffrement du mot de passe est tirée aléatoirement (fonction PHP `rand()`), puis mémorisée dans le fichier `data/meta/_htcookiesalt`. En revanche, côté serveur, DokuWiki mémorise une session PHP par utilisateur avec le nom de l'utilisateur, son mot de passe chiffré, la date/heure de création de la session PHP, et d'autres informations sur l'utilisateur.

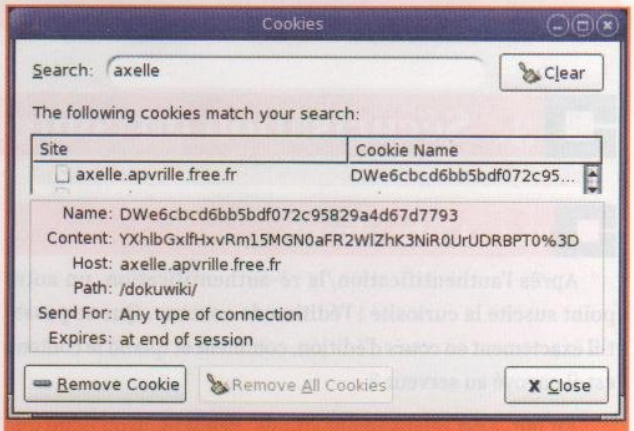


Figure 6 : Cookie d'authentification dont le nom commence par DW

La prochaine fois que l'utilisateur effectue une action privilégiée, son navigateur envoie (entre autres) le cookie de l'utilisateur (voir Figure 7). DokuWiki reçoit la requête et vérifie tout d'abord qu'il a une session PHP au nom de l'utilisateur. Si ce n'est pas le cas, il s'agit d'une « première » authentification, l'utilisateur doit fournir explicitement son identifiant et mot de passe. Si une session PHP existe, DokuWiki vérifie également qu'elle

n'est pas trop vieille, c'est-à-dire, par défaut, qu'elle n'a pas plus de 900 secondes (cela se configure dans le plugin de configuration de DokuWiki, section « *Authentication Settings* » ou dans `conf/dokuwiki.php`, paramètre `auth_security_timeout`). Enfin, il vérifie que le nom d'utilisateur, le mot de passe chiffré et autres informations présentes dans la session PHP correspondent à ce qui se trouve dans le cookie. Si toutes ces conditions sont réunies, l'authentification est automatiquement acceptée, sinon elle est rejetée et une « vraie » authentification doit avoir lieu.

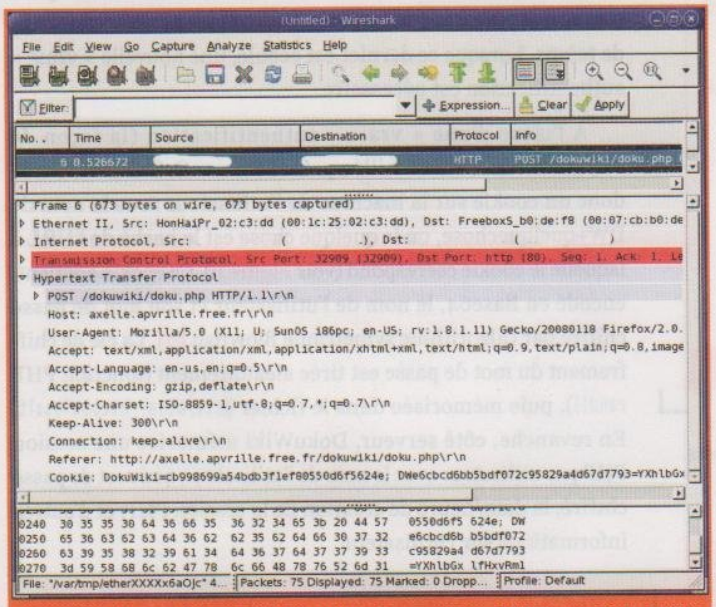


Figure 7 : Le cookie est joint à toutes les requêtes HTTP vers DokuWiki

Les risques liés à ce mécanisme d'authentification automatique sont les suivants :

- Un utilisateur capable de forger un message contenant le cookie de l'utilisateur peut se faire passer pour l'utilisateur. Cette attaque ne peut réussir que si une session PHP est actuellement active pour l'utilisateur et si l'attaquant est capable de récupérer le cookie de l'utilisateur.
- Côté serveur, un attaquant pourrait récupérer le cookie présent dans la requête de l'utilisateur, le déchiffrer (le mot de passe est noté dans `_htcookiesalt`) et ainsi voler le mot de passe de l'utilisateur. Cette attaque nécessite d'avoir la main sur le serveur. Si c'est le cas, il est encore beaucoup plus facile de noter le mot de passe de l'utilisateur lors de son authentification juste avant que le mot de passe ne soit chiffré (rappel : le mot de passe est envoyé en clair au serveur...).

Ces risques sont donc mitigés puisqu'ils sont réels, mais nécessitent que des conditions plus ou moins restrictives soient garanties : à vous de décider si cela vous paraît acceptable. Néanmoins, il aurait certainement été préférable d'opter pour une stratégie plus sûre de ré-authentification. Par exemple, de manière similaire à SSL/TLS, une clé de session pourrait être négociée entre le serveur et l'utilisateur (client).

3. Sécurisation du contenu sous DokuWiki

3.1 Côté client

Après l'authentification/la ré-authentification, un autre point suscite la curiosité : l'édition de contenu. Que se passe-t-il exactement en cours d'édition, comment et quand le contenu est-il envoyé au serveur ?

En fait, c'est relativement simple. Pendant qu'on édite une page, tout se trouve dans un fichier `sessionstore.js` (`.mozilla/firefox/<monprofil>` ou sous Windows `C :\Documents and Settings\<mon identifiant>\Application Data\Mozilla\Firefox\Profiles\<mon profil>\sessionstore.js`). Ce contenu n'est envoyé au serveur qu'au moment où l'on appuie sur le bouton « Save » (ce qui explique d'ailleurs qu'on perde toutes ses modifications si on ne sauvegarde pas...) au sein d'une requête HTTP POST.

Un tel exemple se trouve illustré dans les figures 8 et 9. En cours d'édition, j'ai écrit « Les crocodiles sont des êtres verts à grande gueule ». J'ai retrouvé dans le fichier `sessionstore.js` :

```
Les%20crocodiles%20sont%20des%20%C3%AAtres%20verts%20%C3%A0%20
grande%20gueule.
```

Les caractères spéciaux sont remplacés par leur valeur hexadécimale (%20 pour espace, etc.). Lorsque j'ai appuyé sur « Save », une requête HTTP est partie (Figure 8) et l'on voit, en clair, tout le contenu dans sa condition (sélectionné en bleu). Enfin, la page finale ressemble à la figure 9, page suivante.

Il n'est pas spécialement étonnant de trouver le contenu en clair dans une requête HTTP, car ce dernier n'est pas forcément confidentiel. Si les circonstances l'imposent, pensez cependant à mettre l'accès au wiki en https :// : toutes les requêtes qui transitent seront alors chiffrées et/ou signées par le protocole TLS (suivant la configuration du serveur web).

De même, retrouver le contenu en cours d'édition dans `sessionstore.js` en local sur le poste de l'utilisateur ne peut probablement pas être considéré comme une faille – du moins

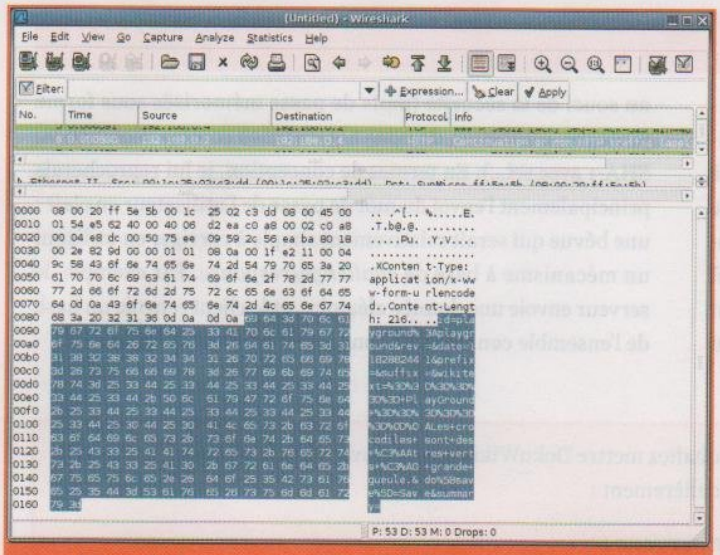


Figure 8 : Le contenu passe en clair dans une requête HTTP

Figure 9 : Article correspondant à la requête de la figure 8.

dans les conditions les plus courantes. Cependant, il est important de se rendre compte que cela peut poser des soucis d'intégrité ou de confidentialité si la machine est partagée par plusieurs utilisateurs ou si les données sont vraiment confidentielles. Dans ce cas-là, il pourrait par exemple être intéressant de mettre le profil Firefox (ou autre navigateur) sur une partition chiffrée.

3.2 Côté serveur

Sur le serveur, le contenu est mémorisé dans le sous-répertoire `data/pages`. Chaque page est représentée par un fichier texte d'extension `.txt`. Ce fichier est en clair, directement lisible par l'être humain. Dans des cas où l'on prône la simplicité, ceci peut tout à fait être considéré comme un avantage par rapport à l'utilisation d'une base de données SQL.

Les *namespaces* de DokuWiki (qui permettent de regrouper certaines pages et notamment de leur assigner en masse des droits d'accès) sont représentés comme des répertoires. Cette structure simple est particulièrement intéressante pour sauvegarder le wiki : on peut facilement sauvegarder les portions les plus intéressantes du wiki, éventuellement de manière incrémentale, compressée ou même chiffrée. Sous Unix, la commande `tar`, par exemple, fera merveille.

Dans l'exemple ci-dessous, on note en l'occurrence 2 namespaces par défaut créés par DokuWiki : `playground` pour pouvoir faire des essais (une sorte de `tmp`) et `wiki`, dédié à un peu de documentation sur DokuWiki. On voit également 4 pages (`hardware`, `jardinage`, `start` et `todo`) et l'on constate que celles-ci sont effectivement lisibles (oui, si vous connaissez des astuces pour venir à bout des fourmis, ça m'intéresse !).

```
[/var/apache2/htdocs/data/pages]:ls -al
total 20
drwxrwxrwx 4 1000     1000     512  Sep 24 22:01 .
drwxrwxrwx 9 1000     1000     512  Sep 16 18:42 ..
-rw-r--r-- 1 webservd webservd 2343 Sep 16 18:43 hardware.txt
-rw-r--r-- 1 webservd webservd 529  Sep 24 22:01 jardinage.txt
drwxrwxrwx 2 1000     1000     512  Jun 26 2007  playground
-rw-r--r-- 1 webservd webservd 126  Sep 16 18:35 start.txt
-rw-r--r-- 1 webservd webservd 268  Sep 24 22:20 todo.txt
drwxrwxrwx 2 1000     1000     512  Jun 26 2007  wiki
[/var/apache2/htdocs/data/pages]:tail jardinage.txt
==== Comment venir à bout des fourmis ====
Le produit en pulvérisation est une poudre à diluer (5 grammes
par litres)
[...]
```

Attention, tous les fichiers disposent des mêmes droits d'accès Unix (lire pour tous, écrire pour l'utilisateur associé au serveur web) : les droits d'accès aux pages ne sont pas gérés par le système d'exploitation lui-même, mais par DokuWiki (cf. paragraphe « contrôle d'accès aux pages protégées »). Par conséquent, il faut veiller à bien protéger le compte `webservd`. La page sécurit´ de DokuWiki regroupe quelques informations utiles à ce sujet [8].

3.3 Code embarqué

Par défaut, DokuWiki est limité à l'insertion de textes, schémas et images au sein des articles, selon une syntaxe classique aux wikis. Cependant, cela pourrait s'avérer trop restrictif si les auteurs d'articles désirent une mise en page plus complexe ou l'inclusion de scripts. Il faudra alors considérer de modifier la configuration de DokuWiki (plugin de configuration, voir Editing Settings) et autoriser l'inclusion d'HTML ou de PHP (`allow embedded HTML` et `allow embedded PHP` ou paramètre `htmlok` et `phpok` dans `conf/dokuwiki.php`). **Attention ! Trou de sécurité potentiel** : cocher ces options signifie avoir une certaine confiance dans les auteurs d'articles : ils doivent être sensibilisés à ne pas inclure n'importe quel script par exemple, car sinon DokuWiki pourrait héberger des pages malicieuses !

Conclusion

Il est difficile de conclure de manière générique sur la sécurité d'un outil, puisque celle-ci dépend des conditions dans lesquelles on se trouve et des risques que nous sommes prêts à prendre. Au mieux, il me semble que l'on peut dire que pour une utilisation peu exigeante (pas de données hautement confidentielles, client et serveur évoluant dans un cadre raisonnablement sécurisé), DokuWiki offre une sécurité potentiellement « convenable », avec un code source ouvert, lisible et présentant un souci de la sécurité (mots de passe mémorisés sous forme protégée, utilisation de bons algorithmes comme Blowfish ou SHA-1 avec sel...). En termes de conception, je lui reprocherais principalement l'envoi du mot de passe de l'utilisateur en clair : une bévue qui serait relativement simple à corriger en utilisant un mécanisme à base de *challenge/response*. Par exemple, le serveur envoie une donnée aléatoire, et le client renvoie un hash de l'ensemble constitué de son mot de passe et de l'aléa. ■

+ code source assez facile à relire
+ utilisation de Blowfish pour protéger le cookie
+ utilisation de SHA-1 avec un sel pour protéger le mot de passe de l'utilisateur
+ facilité à poser des droits d'accès sur certaines pages ou groupes de pages
+ facilité à sauvegarder le wiki
- protocole d'authentification envoyant le mot de passe de l'utilisateur en clair
- mécanisme de ré-authentification (authentification automatique) quelque peu risqué
- certains algorithmes à faible sécurité sont proposés (MD5, crypt...) – probablement pour des raisons de compatibilité.

Si vous souhaitez mettre DokuWiki en œuvre avec plus de sécurité, je conseille tout particulièrement :

Sur le serveur :

>> d'installer DokuWiki sur une partition chiffrée, afin d'en protéger le contenu confidentiel. Sur Linux, il existe une multitude de solutions : TrueCrypt [9], EncFS [10], dmcrypt [11]...,

>> de configurer le protocole TLS pour l'accès web à DokuWiki : cela protègera la communication entre le client et le serveur,

>> de réduire la durée de vie du cookie d'authentification (`auth_security_timeout`),

>> de protéger le compte associé au serveur web,

>> de vérifier / auditer le générateur de nombres aléatoires,

>> d'interdire l'inclusion de code HTML dans les articles.

Sur le client :

>> de mettre le profil du navigateur sur une partition chiffrée, afin de protéger les contenus confidentiels en cours d'édition.

Références

[1] Dokuwiki, http://wiki.splitbrain.org/wiki:dokuwiki

[2] List of wiki software, http://en.wikipedia.org/wiki/List_of_wiki_software

[3] MD5 and MD4 Collision Generator, http://www.stachliu.com/research_collisions.html

[4] SOTIROV (Alexander), STEVENS (Marc), APPELBAUM (Jacob), LENSTRA (Arjen), MOLNAR (David), OSVIK (Dag Arne), DE WEGER (Benne), « MD5 Considered Harmful Today, Creating a rogue CA certificate », *CCC*, 30 décembre 2008, http://www.phreedom.org/research/rogue-ca/

[5] DE CANNIÈRE (Christophe), MENDEL (Florian), RECHBERGER (Christian), « Collisions for 70-step SHA-1 : On the Full Cost of Collision Search », In Proceedings of Selected Areas in Cryptography (SAC 2007), *LNCS* 4876, p. 56-73, Ottawa, Ontario, Canada, 16-17 août, 2007.

[6] Implications des modifications de mot de passe pour les applications, http://dev.mysql.com/doc/refman/5.0/fr/application-password-use.html

[7] The Blowfish Encryption Algorithm, http://www.schneier.com/blowfish.html

[8] Security, DokuWiki, http://www.dokuwiki.org/security

[9] TrueCrypt, http://www.truecrypt.org

[10] EncFS, http://www.argo.net/encfs

[11] dm-crypt, http://www.saout.de/misc/dm-crypt/