misc
32 - Juillet/Août 2007

misc

Multi-System & Internet Security Cookbook

32 Juillet Août 2007

100 % SÉCURITÉ INFORMATIQUE

[DOSSIER]

QUE PENSER DE LA SÉCURITÉ SELON MICROSOFT ?

1/5 Vista : les nouveautés marquantes (p. 10)

2/5 Right Management System : la réponse aux fuites d'information (p. 20)

3/5 NAP et contrôle de conformité : comment rejeter les postes dangereux ? (p. 26)

4/5 Office 2007 & OpenXML : ce nouveau format standard et ouvert est-il vraiment plus sûr ? (p. 35)

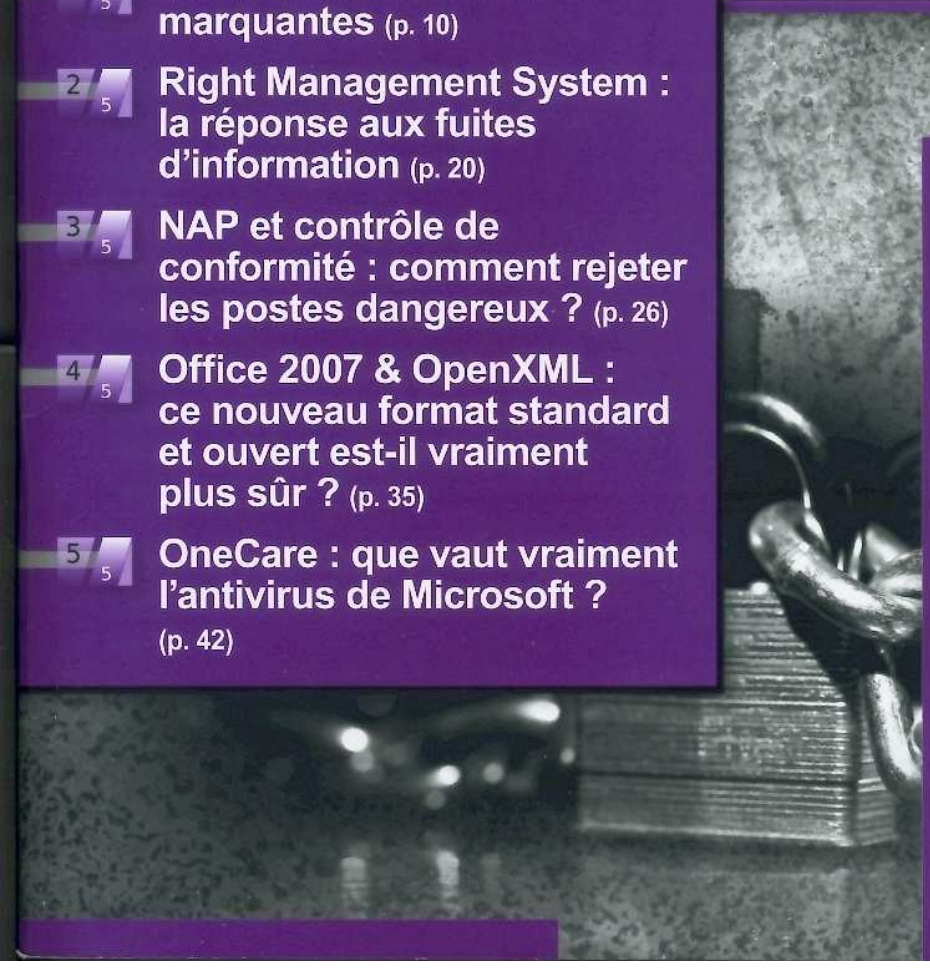
5/5 OneCare : que vaut vraiment l'antivirus de Microsoft ? (p. 42)

TCP/IP / filtre BPF / réseau / sniffer / paquets

FICHE TECHNIQUE

En pratique : Analyse réseau avancée avec tcpdump

(p. 68)



[PROGRAMMATION]

Comprendre l'architecture de la bibliothèque cryptographique OpenSSL (p. 52)

cryptographie / bibliothèque / OpenSSL

[RÉSEAU]

Manipulation du suivi de connexions de Netfilter (p. 62)

Linux / Netfilter / pare-feu / administration / sécurité

SCIENCE

Détecter les malwares à partir de leur comportement (p. 72)

Virus informatiques / détection antivirale / analyse comportementale / polymorphisme / schéma de détection / stratégie de détection

[Édito]

Lutte des classes

C'est l'été. Comme d'habitude, sur toi, je remonte le drap de bain, j'ai peur que tu aies des coups de soleil, comme d'habitude... Oups, pardon, je m'égare. Bref, il fait beau, il sentait bon le sable chaud... Oulalala, je ne vais pas y arriver ! J'ai vraiment besoin de vacances.

Mais, je dois dire que je ne déteste pas notre capitale l'été. Mis à part les bords de Seine qui sont bondés comme les plages de la Côte d'Azur, les rues sont presque vides, les gens souriants et le stationnement gratuit au mois d'août (hum, hum, enfin, sportif comme je suis, je roule à vélo bien sûr ;-)

En fait, cette ambiance détendue, festive, joyeuse rappelle très fortement celle de Rennes au début du mois de juin. Mais bon, avec le soleil en moins cette année : quel temps pourri ! Et froid qui plus est ! Il faut reconnaître que les organisateurs de SSTIC ont fait un boulot extraordinaire. Ils ont apporté quelques bonnes améliorations comme le cocktail, le t-shirt ou un *social event* digne de ce nom tout en maintenant un niveau impeccable pour les interventions. Ils ont juste oublié le soleil...

Je ne reviendrai pas sur le contenu des conférences, d'autant que de nombreux blogs et articles s'en chargent pour moi **[1]** et que tous les articles sont déjà en ligne **[2]**. Non, outre le froid, la pluie et le suspense, limite insoutenable et stressant, lié au retard de livraison des t-shirts (<message perso>pas taper</message perso>), ce qui m'a réellement frappé cette année, c'est le rajeunissement. Il y avait une foule de « jeunes » ! Certains diront que c'est parce que je vieillis. Moi, je verrais ça autrement : ils ont glandé et séché les cours (si, si, j'ai des noms, mais ce serait folie de balancer). Bref, pas (trop) de gens en costume noir, triste et vieux (le costume, pas la personne dedans), et, du coup, une ambiance encore plus sur-vitaminée.

Chose étonnante, il y avait un nid de *reversers* (vous savez, ceux qui lisent de l'assembleur comme d'autres du Balzac). À croire qu'ils s'étaient donnés rendez-vous ou qu'ils se reproduisent ! Certains font des concours pour mettre un maximum de saucisses dans leur bouche, pendant que d'autres gobent des verres (oups, en fait, c'est le même, et il y aurait même des photos). Il y a aussi ceux qui font chanter les disques durs (si, si, vraiment, on peut mixer avec un disque dur, bel exemple de recyclage des déchets) et se frottent à des cactus trop piquants. En vrac, citons encore les drogués des Chocapics, ceux qui déposent des gerbes n'importe où, les coureurs d'autographes, j'en passe et des meilleurs. Plus sérieusement, de ce côté-là, il y a pas mal de monde au portillon, avec un vrai talent et un vrai savoir-faire.

Heureusement, il y avait aussi d'autres personnes, juste un peu moins bizarres que les reversers, mais aussi capables. Tout ça pour dire qu'on commence à voir poindre une nouvelle génération qui déchire, mais que je ne sais pas si les boîtes sont prêtes à les digérer. Il est assez effarant de constater la différence entre le niveau moyen de sécurité de nos systèmes et les capacités réelles de quelques personnes, ce « quelques » ne faisant que croître.

Il y a quelques années, certains Directeurs du Service Informatique et autres RSSI disaient qu'ils ne craignaient rien parce que personne ne savait mener des attaques informatiques. Sécurité par l'ignorance. Je souris maintenant... grâce à cette magnifique revue que vous tenez, grâce à certaines conférences, grâce à l'ouverture d'Internet et des moyens de communication, la connaissance progresse. Mais du coup, que faire de ces gamins ? Ils risquent de se retrouver à faire des Power Point et autres tableaux Excel pour montrer sur de jolis camemberts en couleur que tout va bien. Peu d'entreprises savent utiliser ces compétences intelligemment et efficacement. Et je ne dis pas que c'est facile !!!

Pour terminer, je tiens à remercier vivement Nicolas Ruff pour ce numéro, car il s'est occupé tout seul de tout le dossier, du plan à la gestion des auteurs. Je remercie également les nombreux relecteurs mis à contribution (en vrac) : Aurélien Bordes, Benjamin Caillat, Renaud Feil, Sarah Nataf, Arnaud Pilon, Nicolas Pouvesle, Olivier Revenu.

Bonnes vacances !

Fred Raynal

[1] http://www.sstic.org/SSTIC07/presse.do

[2] http://actes.sstic.org/SSTIC07/

[Sommaire] 32

Advisory 02/2007 du projet Hardened-PHP

Dans cet article, nous présentons l'analyse d'un advisory pour la plate-forme de blogs WordPress, ainsi que l'exploit associé qui permet de récupérer le mot de passe de l'administrateur du blog. La faille a été décrite dans l'advisory 02/2007 du projet hardened-php [1].

mots clés : *failles PHP / Injections SQL*

Analyse de l'advisory

Le titre de l'advisory est : *WordPress Trackback Charset Decoding SQL Injection Vulnerability*. La faille se trouve donc dans la possibilité d'effectuer une injection SQL en jouant avec les jeux de caractères utilisés dans les *trackbacks* du blog. Pour rappel, un trackback ou rétrolien est un « système de liens inter-blogs semi-automatisé » (source Wikipedia). Cela permet aux auteurs des blogs de mentionner simplement dans leurs articles, des articles d'autres blogs.

Prenons maintenant une situation fictive : nous avons un auteur chinois qui a écrit un article sur son blog qui référence un autre article d'un blog en anglais. Le blog chinois va informer le blog anglais qu'il contient un article référençant un de ses articles par l'intermédiaire d'un trackback. Cependant les jeux de caractères utilisés dans les 2 blogs ne sont pas les mêmes, il faut donc gérer cela. PHP dispose d'une fonction nommée `mb_convert_encoding` à cette fin.

Cette fonction est utilisée dans WordPress dans le fichier `wp-trackback.php` :

```
$tb_url    = $_POST['url'];
$title     = $_POST['title'];
$excerpt   = $_POST['excerpt'];
$blog_name = $_POST['blog_name'];
$charset   = $_POST['charset'];

if ($charset)
    $charset = strtoupper( trim($charset) );
else
    $charset = 'ASCII, UTF-8, ISO-8859-1, JIS, EUC-JP, SJIS';

if ( function_exists('mb_convert_encoding') ) { // For international trackbacks
    $title     = mb_convert_encoding($title, get_settings('blog_charset'), $charset);
    $excerpt   = mb_convert_encoding($excerpt, get_settings('blog_charset'), $charset);
    $blog_name = mb_convert_encoding($blog_name, get_settings('blog_charset'), $charset);
}
```

Le trackback est fait par l'intermédiaire d'une requête HTTP utilisant la méthode POST qui transmet différentes informations comme l'URL, le titre de l'article, le nom du blog, un extrait et le *charset* utilisé. En examinant le code, nous voyons que trois informations sont converties en utilisant `mb_convert_encoding` : le titre, l'extrait et le nom du blog.

Dans l'advisory, il est précisé qu'en utilisant un jeu de caractères bien choisi, il est possible de contourner la protection contre l'injection SQL. Une question nous vient à l'esprit : mais où est donc la protection ? Nous n'en voyons aucune dans l'extrait de code précédent. Pourtant la faille se situe ici. La protection doit être faite ailleurs. Nous trouvons la réponse à notre question dans le fichier `wp-settings.php`.

```
// If already slashed, strip.
if ( get_magic_quotes_gpc() ) {
    $_GET    = stripslashes_deep($_GET   );
    $_POST   = stripslashes_deep($_POST  );
    $_COOKIE = stripslashes_deep($_COOKIE);
}

// Escape with wpdb.
$_GET    = add_magic_quotes($_GET   );
$_POST   = add_magic_quotes($_POST  );
$_COOKIE = add_magic_quotes($_COOKIE);
$_SERVER = add_magic_quotes($_SERVER);
```

Des caractères d'échappement sont donc automatiquement ajoutés à toutes les requêtes. Comment allons-nous pouvoir tourner cela à notre avantage ? Nous allons en fait retourner la protection contre elle-même en utilisant un jeu de caractères bien choisi.

« Magic Quotes » ou l'arroseur arrosé

Le jeu de caractères « bien choisi » doit comporter des caractères codés sur plusieurs octets. Comme nous allons le voir, cela permet de tromper l'échappement réalisé par `add_magic_quotes`.

L'advisory indique que les injections SQL ont été effectuées en utilisant le jeu de caractères UTF-7. Nous passerons par ce jeu de caractères UTF-7, mais aussi par SJIS (jeu de caractères japonais) et GBK (jeu de caractères chinois).

Regardons ce qui se passe lorsque, par exemple, le nom du blog comporte le caractère `0xBF27`, caractère codé sur 2 octets non valides en GBK. Ce caractère finit par `0x27` qui est le code ASCII de l'apostrophe. Ce caractère va donc être échappé par le caractère `\` qui a comme code ASCII `0x5C`. Nous nous retrouvons donc avec la suite d'octets `0xBF5C27`. Mais le caractère `0xBF5C` est un caractère valide en GBK ! Nous avons donc maintenant 2 caractères `0xBF5C` et `0x27` ceci après que la fonction `add_magic_quotes` a été exécutée ! Nous pouvons donc insérer des apostrophes propices aux injections SQL !

Préliminaires aux injections SQL

Il faut tout d'abord déterminer si le blog qui utilise WordPress est vulnérable. Pour cela, il nous faut la version de WordPress et savoir si PHP a été compilé avec le support des jeux de caractères étendus (multibytes).

Damien Aumaitre
damien@security-labs.org

Vincent Rasneur
vrasneur@free.fr

Récupération de la version de WordPress

Cet exploit fonctionne avec les versions de WordPress inférieures à la 2.0.6. Un moyen simple de vérifier la version est de regarder le flux RSS (1 ou 2) du blog. Voici un extrait du fichier `wp-rss2.php` :

```
<!-- generator="wordpress/<?php bloginfo_rss('version') ?>" -->
```

Il suffit donc de faire une requête GET sur l'un des flux RSS et de récupérer la version grâce à une expression régulière.

Support des multibytes

Comme nous l'avons vu, la faisabilité de l'exploit repose sur l'emploi de jeux de caractères « exotiques ». Mais PHP peut ne pas disposer des fonctions permettant de décoder ses jeux de caractères vers celui employé par le blog. Il nous faut donc tester d'abord si ces fonctions ont été compilées avec PHP.

Reprenons le début du code source de `wp-trackback.php` :

```
if (empty($title) && empty($tb_url) && empty($blog_name)) {
    // If it doesn't look like a trackback at all...
    wp_redirect(get_permalink($tb_id));
    exit;
}
```

Pour cela, nous allons tester le support multilingue en mettant la variable `$title` à une valeur spéciale qui fera que `$title` sera considéré comme vide si le support est activé et de longueur non nulle sinon.

En PHP, une chaîne est vide si elle est de longueur nulle ou est de longueur 1 et a comme seul caractère un 0. Pour déclencher la fonction `empty($title)`, il nous faudrait donc une chaîne de caractères qui nous donne 0 dans le jeu de caractères ASCII et plusieurs octets dans un jeu de caractères exotique. Cependant, nous avons déjà dit que le jeu de caractères GBK est un surensemble de l'ASCII. Un 0 en GBK est un 0 en ASCII. Il en est de même pour le jeu de caractères japonais SJIS. Il ne nous reste plus qu'UTF-7 pour savoir si PHP dispose des fonctions `mb_*`.

Rappelons brièvement le principe d'UTF-7. UTF-7 est un jeu de caractères conçu pour représenter tous les caractères grâce aux caractères [A-Za-z0-9+/]. Les chaînes codées en UTF-7 débutent par un '+' et se finissent par un '-' ou un caractère ne figurant pas dans [A-Za-z0-9+/]. Nous emploierons UTF-7 pour coder un '0'. Voici le déroulement du codage :

- Convertissons le '0' en UTF-16 : '0' = 0x30 en ASCII, et donc 0x0030 en UTF-16.
- Découpons ce caractère en paquets de 6 bits, remplissons avec des 0 si la valeur en bits du caractère UTF-16 n'est pas un multiple de 6 : 000000|000011|000000 (soient 2 bits de *padding*).
- Codons ces paquets en base64 modifiée [7] : ADA.
- Rajoutons le '+' au début et le '-' à la fin : +ADA-.

Il reste un problème : le '+' en début de la chaîne fait partie des caractères utilisés par la requête POST du trackback. Et si nous codions aussi ce '+', mais autrement ? Le contenu de la requête POST aura le « Content-Type » suivant : « application/x-www-form-urlencoded ». Nous pouvons donc coder ce '+' par le codage utilisé dans les URL, c'est-à-dire par '%' suivi de la valeur hexadécimale du code ASCII du caractère '+'. '0' en UTF-7 et paré pour le test des fonctions `mb_*` est donc '%2BADA-'.

Il suffit donc d'envoyer un trackback avec `$tb_url` et `$blog_name` contenant des chaînes de caractères vides et de mettre dans `$title` la valeur '%2BADA-'. Si l'on est redirigé, cela signifie que la version de PHP utilisée supporte les jeux de caractères étendus. Il faut maintenant récupérer le préfixe des tables utilisées par le blog.

Préfixe des tables MySQL utilisées par le blog

Un problème se pose : dans quelle table récupérer ces informations ? Nous pouvons le savoir en installant WordPress et en regardant la structure de sa base de données. Mais considérez la ligne suivante dans le fichier `wp-config.php` :

```
// You can have multiple installations in one database if you give each a unique prefix
$table_prefix  = 'foo_';   // Only numbers, letters, and underscores please!
```

L'utilisateur de WordPress peut contrôler le début du nom de ses tables. Il faut prendre cela en compte. Nous retrouvons le préfixe utilisé grâce à quelques lignes de code dans `wp-includes/wp-db.php`.

```
// If there is an error then take note of it..
if ( mysql_error() ) {
    $this->print_error();
    return false;
}
```

Regardons également ces quelques lignes issues de la fonction `print_error` :

```
if ( $this->show_errors ) {
    // If there is an error then take note of it
    print "<div id='error'>
    <p class='wpdberror'>"/*<strong>WordPress database error:</strong> [$str]<br />*/
    ."<code>$query</code></p>
    </div>";
} else {
    return false;
}
```

Si nous injectons du mauvais code SQL dans une requête, WordPress nous enverra en réponse la requête complète. Notons que si l'attribut `$show_errors` était à faux, nous ne pourrions pas

programmer d'exploit. Mais WordPress l'initialise à vrai dès l'instanciation de la classe et le laissera à vrai sauf dans quelques cas très particuliers (nous discutons plus en profondeur de ceci dans la dernière partie de l'article).

Injectons dans la joie et la bonne humeur

Voyons en détail le flot d'exécution de WordPress lors de la réception d'un trackback. Il commence au début du fichier `wp-trackback.php` :

⇨ Récupération des variables provenant de la requête POST.

⇨ Vérification de l'existence d'un trackback provenant du même auteur dans la base de données (voir partie 1)

⇨ Appel à la fonction `wp_new_comment` située dans `wp-includes/comment-funs.php`. Comme l'indique le nom de ce fichier, ces fonctions sont également exécutées lors de l'ajout de simples commentaires.

⇨ `wp_new_comment` appelle deux fonctions intéressantes.

↳ `wp_allow_comment` exécute quelques protections, notamment contre les commentaires/trackbacks en double (voir partie 2) et contre le *flood* (voir partie 3).

↳ Cette fonction avertit de manière systématique le modérateur du blog par email de l'insertion d'un trackback via la fonction `wp_notify_moderator`. `wp_insert_comment` permet d'insérer le commentaire/trackback dans la base de données. (voir partie 4)

Notre but est donc de détourner ce flot d'exécution via des injections SQL en essayant de minimiser les traces de notre exploitation. Les différentes injections seront contenues dans les variables issues du trackback et sur lesquelles nous pouvons contrôler le jeu de caractères.

Afin d'aider à la compréhension de l'exploitation, voici les extraits du code de WordPress correspondant aux parties mentionnées :

Partie 1

```
$dupe = $wpdb->get_results("SELECT * FROM $wpdb->comments WHERE comment_
post_ID =
 '$comment_post_ID' AND comment_author_url = '$comment_author_url'");
if ( $dupe )
 trackback_response(1, 'We already have a ping from that URI for this
 post.');
```

Partie 2

```
// Simple duplicate check
$dupe = &amp;quot;SELECT comment_ID FROM $wpdb-&amp;gt;comments WHERE comment_
post_ID = '$comment_post_ID' AND (
comment_author = '$comment_author' &amp;quot;;
if ( $comment_author_email )
        $dupe .= &amp;quot;OR comment_author_email = '$comment_author_email'
        &amp;quot;;
$dupe .= &amp;quot;) AND comment_content = '$comment_content' LIMIT 1&amp;quot;;
if ( $wpdb->get_var($dupe) )
        die( __('Duplicate comment detected; it looks as though you\'ve
        already said that!') );
```

Partie 3

```
// Simple flood-protection
if ( $lasttime = $wpdb->get_var("SELECT comment_date_gmt FROM $wpdb-
>comments WHERE
comment_author_IP = '$comment_author_IP' OR comment_author_email =
'$comment_author_email' ORDER BY
comment_date DESC LIMIT 1") ) {
    $time_lastcomment = mysql2date('U', $lasttime);
    $time_newcomment  = mysql2date('U', $comment_date_gmt);
    if ( ($time_newcomment - $time_lastcomment) < 15 ) {
        do_action('comment_flood_trigger', $time_lastcomment, $time_
        newcomment);
        die( __('Sorry, you can only post a new comment once every 15
        seconds. Slow down
cowboy.') );}
}
```

Partie 4

```
$result = $wpdb->query("INSERT INTO $wpdb->comments
(comment_post_ID, comment_author, comment_author_email, comment_author_
url,
comment_author_IP, comment_date, comment_date_gmt, comment_content,
comment_approved, comment_agent, comment_type, comment_parent, user_id)
VALUES
('$comment_post_ID', '$comment_author', '$comment_author_email',
'$comment_author_url', '$comment_author_IP', '$comment_date',
'$comment_date_gmt', '$comment_content', '$comment_approved',
'$comment_agent', '$comment_type', '$comment_parent', '$user_id')
");
```

L'exploitation fonctionne en deux temps. Un premier trackback est envoyé ; il permet de récupérer le préfixe des tables SQL et de ne pas avertir le modérateur lors des injections ultérieures. Les autres trackbacks envoyés récupèreront le *hash* du mot de passe de l'administrateur stocké dans la base de données.

Première étape, premier trackback

Nous aimerions récupérer le hash sans que le modérateur ne reçoive de message d'avertissement. Le flot d'exécution doit donc être arrêté avant l'appel à `wp_notify_moderator`. Les deux protections situées juste au-dessus se terminent par un `die`, et peuvent nous permettre de stopper le flot au bon moment. Comment les activer ? La protection contre les duplicatas utilise une variable que nous pouvons contrôler en temps voulu. Le plus important ici est de forcer la protection contre le flood à toujours s'exécuter lors des prochains trackbacks en insérant ce premier trackback avec une date se situant dans le futur. Du coup, la différence `$time_newcomment - $time_lastcomment` sera toujours négative, et la protection contre le flood sera activée.

WordPress assigne la valeur de `$blog_name` à la variable `$comment_author` dans `wp-trackback.php`. Or, nous contrôlons la valeur et le jeu de caractères de `$blog_name`. Nous pouvons donc insérer les données que nous voulons dans la requête `INSERT INTO` de la fonction `insert_comment`. Pour passer la protection anti-flood par la suite, mettons `comment_date_gmt` à une date future et `comment_author_email` à "". Pour ne pas avoir de problèmes avec les guillemets des différentes valeurs à insérer, nous les coderons de la même façon que le "0" du test des fonctions `mb_*`, soit %2BACc- pour un guillemet.

Il nous reste la récupération du préfixe grâce à une requête SQL invalide. Puisque la variable contenant l'injection SQL (`$comment_author`) est la même pour la protection contre les duplicatas et l'insertion dans la base de données, la requête de protection ne pourra pas être correcte et forcera WordPress à afficher un message d'erreur. Et nous aurons notre préfixe.

Deuxième étape, 128 trackbacks

Place à la récupération du hash proprement dit ! Nous devons passer la partie 1, `$comment_author_url` doit donc être différent de celui du trackback que nous venons d'insérer précédemment. Par la suite, nous arrêterons le flot d'exécution avant l'insertion des trackbacks dans la base de données, nous pouvons garder le même `$comment_author_url` pour les prochains trackbacks à envoyer.

Passons à la partie 2. Comme nous l'avons déjà dit, nous contrôlons `$blog_name`, soit `$comment_author`. Nous pouvons exécuter n'importe quelle requête SQL dans la protection anti-duplicatas ; forçons WordPress à nous renvoyer un bit du hash du mot de passe administrateur. WordPress met la valeur nulle dans `$comment_author_email` dans le cas d'un trackback. La requête SQL sera donc :

```
SELECT comment_ID FROM $wpdb->comments WHERE
comment_post_ID = '$comment_post_ID' AND ( comment_author =
'$comment_author' ) AND comment_content = '$comment_content' LIMIT 1
```

`$comment_author` va devoir ici forcer le `SELECT` à ne rien renvoyer (par exemple par `"'AND 1=0)"`, la récupération du bit demandé du hash se fera par un `UNION SELECT`.

Une valeur de `$comment_author` pour récupérer l'injection est (après la conversion de l'apostrophe servant à l'injection) : `"' AND 1=0) UNION SELECT MAKE_SET(SUBSTRING(LPAD(CONV(SUBSTRING((SELECT user_pass from <préfixe des table SQL>users WHERE user_login=char(97,100,109,105,110)),<numéro du caractère voulu>,1),16,2),4,0),<numéro du bit dans le caractère>,1),1) #"`.

Détaillons l'`UNION SELECT` :

⇨ On commence par demander le hash du mot de passe d'un utilisateur s'appelant 'admin', c'est-à-dire 97,100,109,105,110 avec des caractères ASCII codés en décimal. On obtient une chaîne de caractères composée de 32 lettres pouvant avoir comme valeur [0-9a-f] (le hash est un md5, donc il fait 128 bits. Un chiffre hexadécimal code 4 bits et 128/4 = 32.)

⇨ On sélectionne la lettre désirée : SUBSTRING(hash, numéro de la lettre, longueur de la sous-chaîne = 1 octet). Rappelons que cette lettre est un chiffre hexadécimal.

⇨ On convertit ce chiffre hexadécimal en base 2 : CONV(chiffre hexadécimal, 16, 2).

⇨ On rajoute des 0 à gauche si nous n'avons pas 4 chiffres binaires : LPAD(valeur binaire, 4, 0). Par exemple, si le chiffre hexadécimal est 2, sa représentation en binaire sera 10 et non pas 0010 comme nous le désirons.

⇨ On sélectionne le bit désiré dans cette suite de bits : SUBSTRING(suite de bits, position du bit, longueur de la sous-chaîne=1)

⇨ La dernière fonction n'est pas nécessaire, mais elle permet au SELECT de ne rien renvoyer au lieu de renvoyer 0. (MAKE_SET(1, 'a', 'b') renvoie 'a', MAKE_SET(2, 'a', 'b') renvoie 'b' et MAKE_SET(0, 'a', 'b') ne renvoie rien.)

Avec cette technique, 128 requêtes seront nécessaires pour récupérer le mot de passe. En revanche, nous contrôlons entièrement le flot d'exécution et aucun message d'ajout de trackback ne sera envoyé à l'administrateur.

Il nous reste un point à prendre en compte pour avoir une requête valide : le problème vient de la variable `$comment_content`. Elle est définie par WordPress comme `$comment_content = "<strong>$title</strong>\n\n$excerpt";` dans `wp-trackback.php`. Les retours à la ligne vont faire croire à WordPress qu'il y a une autre requête SQL après celle que nous venons de forger. Pour pallier cela, nous pouvons mettre un "#" dans `$excerpt`, variable que nous contrôlons. Une solution alternative serait de mettre un début de commentaire multiligne, c'est-à-dire "/*", à la fin de `$comment_author` pour ne pas avoir de problème.

Une fois que notre requête aura été correctement exécutée, un bit demandé ayant la valeur 1 entraînera l'envoi du message « *Duplicate comment detected; it looks as though you've already said that!* ». Si le bit est à 0, le test sera faux. Nous continuerons ainsi l'exécution vers la protection anti-flood. Puisque nous avons inséré un trackback avec une date future, l'expression `$time_newcomment - $time_lastcomment` sera toujours négative, donc inférieure à 15 ! Nous recevrons donc le message « *Sorry, you can only post a new comment once every 15 seconds. Slow down cowboy.* ». Le message retourné révèle donc la valeur du bit désiré du hash du mot de passe administrateur ; une simple boucle nous permettra de récupérer la totalité (du hash).

Une pincée de force brute

L'advisory nous recommande de créer un faux cookie pour nous identifier en tant qu'administrateur du blog. Cependant, il paraît plus simple de récupérer directement le mot de passe administrateur à partir du hash. Avant d'aller plus loin, regardons un extrait du fichier `wp-admin/install.php` servant à l'installation de WordPress :

```
// Set up admin user
$random_password = substr(md5(uniqid(microtime())), 0, 6);
...
$wpdb->query("INSERT INTO $wpdb->users (ID, user_login,
user_pass, user_email, user_registered, display_name, user_nicename) VALUES
( '1', 'admin', MD5('$random_password'), '$admin_email', NOW(),
'$display_name', 'admin')");
```

Le mot de passe administrateur est contenu dans `$random_password`, variable remplie par les 6 premiers caractères d'un MD5. Le mot de passe administrateur par défaut se compose donc de 6 caractères compris entre '0' et 'f' et reste facile à « bruteforcer » si l'administrateur ne l'a pas changé.

Comment corriger le problème ?

Le moyen de contrer l'exploitation est très simple : il consiste à enlever les *backslashes* ajoutés par les « magic quotes » ou par WordPress lui-même. Le rajout des caractères d'échappement

sera effectué uniquement après le déroulement de la conversion, c'est-à-dire après l'exécution des fonctions `mb_convert_encoding`.

Voici le patch tel qu'il est présenté dans la version 2.0.6 :

```
$tb_url  = $_POST['url'];
$charset = $_POST['charset'];

// These three are stripslashed here so that they can be properly escaped
after mb_convert_encoding()
$title     = stripslashes($_POST['title']);
$excerpt   = stripslashes($_POST['excerpt']);
$blog_name = stripslashes($_POST['blog_name']);

if ($charset)
    $charset = strtoupper( trim($charset) );
else
    $charset = 'ASCII, UTF-8, ISO-8859-1, JIS, EUC-JP, SJIS';

if ( function_exists('mb_convert_encoding') ) { // For international
trackbacks
    $title     = mb_convert_encoding($title, get_option('blog_charset'),
    $charset);
    $excerpt   = mb_convert_encoding($excerpt, get_option('blog_charset'),
    $charset);
    $blog_name = mb_convert_encoding($blog_name, get_option('blog_
    charset'), $charset);
}

// Now that mb_convert_encoding() has been given a swing, we need to
escape these three
+ $title     = $wpdb->escape($title);
+ $excerpt   = $wpdb->escape($excerpt);
+ $blog_name = $wpdb->escape($blog_name);
```

Il faudrait aussi rendre WordPress moins bavard, notamment lors d'erreurs de syntaxe dans les requêtes SQL. Si WordPress n'envoyait pas au client la requête complète, il sera difficile de récupérer la table dans laquelle récupérer le hash du mot de passe (à part si l'utilisateur du blog n'a pas pris soin de modifier le préfixe par défaut).

Regardons le début du code de la partie de WordPress qui gère la partie base de données, c'est-à-dire `wp-includes/wp-db.php`.

```
class wpdb {
    var $show_errors = true;
    [...]
    function print_error($str = '') {
        global $EZSQL_ERROR;
        if (!$str) $str = mysql_error();
        [...]
        // Is error output turned on or not..
        if ( $this->show_errors ) {
            // If there is an error then take note of it
            print "<div id='error'>
            <p class='wpdberror'><strong>WordPress database error:</
            strong> [$str]<br />
            <code>$query</code></p>
            </div>";
        } else {
            return false;
        }
    }
```

Le niveau de verbosité de WordPress est contrôlé par l'attribut `$show_errors`. Celui-ci est par défaut à vrai lors de l'instanciation de la classe `wpdb`. Voici le motif que WordPress utilise lorsqu'il ne veut pas afficher une erreur SQL :

```
$wpdb->hide_errors();
// requêtes SQL dont il ne faut pas afficher les erreurs
$wpdb->show_errors();
```

Le problème de ce motif est que WordPress met automatiquement `$show_errors` à vrai lorsqu'il a fini d'exécuter ses requêtes. WordPress exécute notamment de telles requêtes dans le fichier `wp-settings.php`, juste avant l'exécution du fichier `wp-trackback.php`. Nous aurons donc toujours un WordPress bavard et un exploit possible. Il aurait été plus simple de regarder la valeur de `$show_errors` avant de la remettre à vrai.

Reprenons la discussion sur l'exploit et ses limites : son principe repose sur l'envoi de 2 messages d'erreur différents.

```
die( __('Duplicate comment detected; it looks as though you\'ve already
said that!') );
die( __('Sorry, you can only post a new comment once every 15 seconds.
Slow down cowboy.') );
```

Nous voyons ici que les chaînes de caractères sont encadrées par "_('...')". Cela est dû à l'utilisation de la bibliothèque `gettext` qui gère l'internationalisation des logiciels. Notre exploit fonctionne avec la version anglaise de WordPress, mais, comme nos expressions régulières recherchent des chaînes de caractères en anglais l'exploit ne fonctionnera plus sur une version française par exemple. Il faudra donc changer les expressions régulières de l'exploit et mettre leurs équivalents français.

Références

[1] L'advisory PHP :
http://www.hardened-php.net/advisory_022007.141.html

[2] Encodage base64 modifié :
http://en.wikipedia.org/wiki/Modified_Base64

[3] Encodage GBK :
http://shiflett.org/archive/184

[4] Encodage SJIS :
http://www.postgresql.org/docs/techdocs.50

[5] Encodage UTF-7 :
http://en.wikipedia.org/wiki/UTF-7

[6] Le code source de l'exploit :
http://miscmag.com/[ou?]

Sécurité de Windows Vista

Présenter la sécurité d'un logiciel de 70 millions de lignes de code (6 milliards de dollars de coût de développement) en quelques pages relève de l'impossible. Aussi, cet article entend mettre en avant quelques nouveautés marquantes, impactant la sécurité du système d'exploitation phare de Microsoft.

mots clés : ***sécurité / nouveautés / Microsoft Windows Vista***

Introduction

Avant de parler sécurité sous Windows Vista, encore faut-il être capable de définir la liste des fonctions de sécurité du système. Cette tâche est plus ardue qu'il n'y paraît : par exemple le *Protected Media Path* (PMP) est destiné à la protection de contenus numériques (DRM), mais pourrait être détourné de son usage initial par des *malwares*. D'autre part, comment classer les fonctions de support de la sécurité, comme le nouveau moteur de journalisation basé sur XML ?

Dans cet article, nous évoquerons successivement :

- ⇨ la gestion des utilisateurs et des permissions ;
- ⇨ les stratégies de sécurité ;
- ⇨ les services ;
- ⇨ les mécanismes d'isolation des processus ;
- ⇨ la sécurité des accès réseau ;
- ⇨ les protections techniques contre les débordements de *buffer* ;
- ⇨ les failles déjà connues.

Ce qui donne un ordre du jour assez dense.

Les points trop spécifiques ou abondamment traités dans la littérature seront laissés de côté. Je pense particulièrement à PatchGuard v2, qui n'est présent que sur la version 64 bits de Vista et qui a été largement décrit dans un article du magazine *Uninformed* [1]. De même, Internet Explorer 7 mérite un article à lui seul et ne sera pas abordé ici.

Nouveautés élémentaires

Comptes et groupes

La configuration « par défaut » des comptes et des groupes locaux a été revue. Parmi les nouveautés, on peut citer :

⇨ Le compte « administrateur » *built-in* (rid=500) est désactivé par défaut.

⇨ Le groupe « utilisateurs avec pouvoir » est déprécié (*deprecated*).

↳ Ce groupe existe toujours pour assurer la compatibilité ascendante, mais il n'a plus de droits par défaut, en particulier sur les répertoires « *Program Files* » et « Windows ».

↳ Il était de toute façon largement démontré que les membres de ce groupe pouvaient élever leurs privilèges vers ceux du groupe « administrateurs » sans problème (ex. : droit « modifier » sur le répertoire « Windows »).

⇨ Les comptes « HelpAssistant » et « SUPPORT_388945a0 », utilisés par la fonction d'assistance à distance, ont été supprimés. L'assistance à distance s'exécute désormais comme un programme « classique » dans la session de l'utilisateur.

⇨ *A contrario*, de nombreux groupes ont été ajoutés pour permettre aux utilisateurs « non-administrateurs » de réaliser des tâches privilégiées. Citons, par exemple, les groupes prédéfinis suivants :

- ↳ *Cryptographic Operators;*
- ↳ *Event Log Readers;*
- ↳ *Performance Log Users;*
- ↳ *Performance Monitor Users.*

⇨ Quant aux nouveaux SID built-in, on peut citer par exemple le SID « *OWNER RIGHTS* » (S-1-3-4) qui sera décrit dans la section suivante.

Permissions

Les permissions sur les objets système (fichiers, clés de base de registres, etc.) ont été durcies par rapport aux versions antérieures de Windows. Mais surtout, deux changements fondamentaux ont été apportés au modèle de sécurité :

⇨ Le propriétaire d'un objet n'a plus nécessairement de prérogatives inaliénables sur cet objet. Il peut être soumis au contrôle d'accès conformément au modèle de sécurité Windows.

Par défaut, le contrôle d'accès est compatible avec les versions antérieures de Windows, ce qui signifie que le propriétaire d'un objet possède le droit implicite et inaliénable de modifier les droits d'accès sur cet objet. Toutefois si le SID built-in « OWNER RIGHTS » apparaît dans la DACL, alors les prérogatives du propriétaire actuel de l'objet sont limitées par le contenu de cette ACE. Contrairement au SID « CREATOR OWNER », qui est remplacé par l'identifiant utilisateur au moment où l'ACL est mise en place, le SID « OWNER RIGHTS » est remplacé par l'identifiant utilisateur au moment où l'ACL est évaluée.

Remarque : dans le cas où plus personne n'a accès à l'objet, un administrateur doit en prendre possession pour pouvoir changer l'ACL (c'est-à-dire l'administrateur possède toujours une prérogative inaliénable sur les objets).

⇨ Les services peuvent se voir affecter un identifiant de sécurité (SID), délivré par l'autorité « NT SERVICE » (S-1-5-80), et donc participer au contrôle d'accès (c'est-à-dire apparaître dans des ACL).

Nicolas Ruff
EADS-IW SE/CS
nicolas.ruff@eads.net

Le contrôle d'accès par service est mis en œuvre par le *firewall* intégré (voir plus loin) et par *Windows Resource Protection* (WRP) [2]. Ce mécanisme est le successeur de *Windows File Protection* (WFP). Il apporte en plus la protection des répertoires et des clés de base de registres. WRP est implémenté sous forme de permissions autorisant uniquement le service « *TrustedInstaller* » à modifier les éléments protégés par WRP (essentiellement des fichiers du répertoire « Windows »). Le service en question s'appelle « *Windows Modules Installer* » et son exécutable se trouve dans `C:\Windows\Servicing\TrustedInstaller.exe`.

⇨ Il existe un SID permettant de limiter les opérations d'écriture : « *NT AUTHORITY\WRITE RESTRICTED* » (S-1-5-33).

Lorsque cet identifiant est présent dans un jeton de sécurité, ce jeton est traité comme « restreint » *pour les opérations d'écriture uniquement* (pour mémoire, un processus obtient un jeton restreint sous Windows XP SP2 lorsqu'il est exécuté avec « bouton droit/exécuter en tant que/protéger mon ordinateur [...] »). Ce mécanisme est principalement utilisé par les services. Par exemple, le service de *firewalling (Microsoft Protection Service)* est de type « restreint en écriture », ce qui l'empêche de pouvoir modifier les règles qu'il applique :

```
C:\> sc qsidtype mpssvc
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: mpssvc
SERVICE_SID_TYPE:  RESTRICTED
```

Système de fichiers

L'une des nouveautés du système de fichiers Windows Vista est le support des liens symboliques (*symlinks*). Windows supporte depuis longtemps (c'est-à-dire depuis Windows 2000) les « points de jonction » (montage d'une partition dans un répertoire), les *hard links* (sur un même volume logique) et les raccourcis (fichiers « .LNK »). Ce sont des fonctionnalités méconnues, accessibles par la commande `fsutil`.

Les liens symboliques sont une nouvelle fonctionnalité à part entière, accessible par la commande `mklink`. On regrettera néanmoins que cette fonctionnalité ne soit pas supportée par le système de fichiers, mais par l'environnement – un lien symbolique vu depuis un système d'exploitation antérieur à Windows Vista n'est qu'un simple fichier...

Qui dit « lien symbolique », dit « *race condition* » possible (attaque bien connue dans les environnements *nix). Toutefois, les liens symboliques sont quasiment inutilisés par les applications actuelles – seul Windows s'en sert, par exemple pour créer un lien nommé « Documents and Settings » vers le répertoire « Users » (le nouveau répertoire de profils).

Stratégies de sécurité

Depuis Windows 2000, l'ensemble des options de configuration impactant directement la sécurité sont regroupées dans le composant d'administration « Stratégie de sécurité ». Les rubriques suivantes y sont disponibles :

⇨ stratégies de comptes : pas de changement ;

↳ à noter que Windows Longhorn Server devrait introduire la possibilité d'avoir des stratégies de mot de passe différentes pour les utilisateurs d'un même domaine ;

⇨ stratégies locales/Audit : pas de changement ;

⇨ stratégies locales/Droits utilisateur : 6 nouveaux droits (voir ci-dessous) ;

⇨ stratégies locales/Options de sécurité : une dizaine de changements (voir ci-dessous) ;

⇨ Windows Firewall : nouvel onglet, qui sera traité ci-après ;

⇨ stratégies de clé publique ; stratégie de restriction logicielle ; stratégies IPsec : ne seront pas traitées dans cet article.

Droits utilisateur

Windows Vista introduit 6 nouveaux droits :

1▶ *Access credential manager as a trusted caller*

Le *Credential Manager* est l'emplacement de stockage « sécurisé » des mots de passe mémorisés par le système (ex. : la fameuse case à cocher « mémoriser mon mot de passe » qui apparaît sur les boites de dialogue de connexion à des ressources réseau). Ca n'a l'air de rien, mais Microsoft possède un brevet [3] là-dessus quand même.

Sous le capot, Windows utilise l'API de DPAPI `Crypt[Un]ProtectData()` pour chiffrer les mots de passe et les stocker dans la base de registres. Normalement, les données enregistrées par un utilisateur ne sont visibles que de lui seul tandis que les données enregistrées par le compte machine sont visibles de tous les utilisateurs. Il semblerait que ce comportement ait changé dans Vista (possibilité d'avoir des secrets machine inaccessibles aux utilisateurs), mais ce point reste à confirmer.

Valeur par défaut : vide.

2▶ *Change time zone* : que dire de plus ?

Valeur par défaut : Administrateurs, Utilisateurs, LOCAL SERVICE.

3▶ *Create symbolic links* : Windows Vista supporte désormais les liens symboliques, présentés précédemment.

Valeur par défaut : Administrateurs.

4▶ *Modify an object label* : le « label » correspond au niveau d'intégrité des processus, qui sera présenté dans la suite de l'article.

Valeur par défaut : vide.

5▶ *Synchronize directory service data* : démarrer une synchronisation des annuaires Active Directory.

Valeur par défaut : vide.

6▶ *Increase a process working set* : le « *working set* » correspond à la quantité de mémoire physique allouée au processus –

tout ce qui est au-delà étant déporté dans le fichier d'échange (*swapfile*).

Valeur par défaut : Utilisateurs.

Options de sécurité

Les modifications apportées aux options de sécurité se classent en 3 catégories :

1▶ Les nouvelles options.

2▶ Les options dont la valeur par défaut est plus sûre.

3▶ Les options dont la valeur par défaut est moins sûre (!).

Parmi les nouvelles options, on peut citer :

⇨ « *Audit: Force audit policy subcategory settings to override audit policy category settings* »

↳ Les nouveautés en matière d'audit font l'objet d'un paragraphe à part entière. Cette option fait référence aux sous-catégories d'audit, qui permettent un audit plus granulaire dans Windows Vista.

⇨ « *Network access: Remotely accessible registry paths and sub-paths* »

↳ Windows Vista permet de spécifier si toutes les sous-clés d'une clé de base de registres accessible à distance de manière anonyme sont également accessibles (le comportement par défaut de Windows XP SP2).

⇨ « *Network access: Restrict anonymous access to named pipes and shares* »

↳ Cette option permet d'activer ou désactiver globalement l'accès anonyme aux canaux nommés et aux partages. Lorsque l'accès anonyme est globalement désactivé, les options permettant d'autoriser l'accès anonyme au cas par cas sont prises en compte.

⇨ « *System settings: Optional subsystems* »[1]

↳ Les sous-systèmes sont une fonctionnalité méconnue de Windows offrant une couche de compatibilité avec des binaires OS/2 ou POSIX par exemple (le sous-système par défaut est Win32). Tous les guides de sécurité recommandent de désactiver les sous-systèmes optionnels. C'est désormais possible via une option de sécurité, ce qui évite d'avoir à éditer la base de registres.

⇨ « *System settings: Use certificate rules on Windows executables for software restriction policies* »

↳ Cette option permet d'activer ou désactiver globalement le support des signatures numériques pour les stratégies de restriction logicielle.

Quant aux options permettant de régler le comportement du composant UAC, elles ne sont pas détaillées ici, car UAC fait l'objet d'un paragraphe à part entière.

Parmi les options dont la valeur par défaut est « plus sûre », on peut citer :

⇨ « *Network Access : Named Pipes that can be accessed anonymously* »

⇨ « *Network access: remotely accessible registry paths* »

⇨ « *Network access: shares that can be accessed anonymously* »

Il est difficile de comparer de manière rigoureuse et exhaustive un Windows XP SP2 et un Windows Vista sur ces points, car la configuration dépend du type de système étudié (Vista Business joint à un domaine, Vista Ultimate en *workgroup*, etc.) et des rôles configurés (partage de fichier anonyme activé, etc.). Néanmoins la configuration « par défaut » d'un Vista semble être toujours au moins aussi robuste que celle de Windows XP SP2. Par exemple, sur Windows XP SP2 les partages « COMCFG » et « DFS$ » sont déclarés comme accessibles anonymement, même si ceux-ci n'existent pas sur le système considéré – ce qui n'est plus le cas avec Windows Vista.

Une amélioration notable pour la sécurité des mots de passe provient du fait que Vista ne stocke plus par défaut le *hash* LM. De plus, Windows Vista s'authentifie uniquement en NTLMv2 (mais il accepte toujours les réponses LM et NTLM[2]). Néanmoins, comme l'a montré Aurélien Bordes lors de SSTIC 2007 [4], le hash LM est toujours présent en mémoire lorsque l'utilisateur est logué.

Parmi les options « moins sûres », on peut citer notamment le fait que l'installation de *drivers* non signés réussit silencieusement (l'utilisateur n'est plus notifié). Il est vrai que cette option embêtait beaucoup les fabricants de hardware, dont très peu ont fait l'effort (financier) de faire signer leurs drivers…

Audit

La granularité de l'audit sous Windows Vista a été grandement améliorée par l'introduction de « sous-catégories », dans chacune des 9 catégories principales d'audit.

```
C:\> auditpol /list  /subcategory:*
Category/Subcategory
System
  Security State Change
  Security System Extension
  System Integrity
  IPsec Driver
  Other System Events
Logon/Logoff
  Logon
  Logoff
  Account Lockout
  IPsec Main Mode
  IPsec Quick Mode
  IPsec Extended Mode
  Special Logon
  Other Logon/Logoff Events
Object Access
  File System
  Registry
  Kernel Object
  SAM
  Certification Services
  Application Generated
  Handle Manipulation
  File Share
```

[1] *Cette option existait également dans Windows 2003 Server.*

[2] *Paramètre LMCompatibilityLevel à 3.*

```
  Filtering Platform Packet Drop
  Filtering Platform Connection
  Other Object Access Events
Privilege Use
  Sensitive Privilege Use
  Non Sensitive Privilege Use
  Other Privilege Use Events
Detailed Tracking
  Process Creation
  Process Termination
  DPAPI Activity
  RPC Events
Policy Change
  Audit Policy Change
  Authentication Policy Change
  Authorization Policy Change
  MPSSVC Rule-Level Policy Change
  Filtering Platform Policy Change
  Other Policy Change Events
Account Management
  User Account Management
  Computer Account Management
  Security Group Management
  Distribution Group Management
  Application Group Management
  Other Account Management Events
DS Access
  Directory Service Access
  Directory Service Changes
  Directory Service Replication
  Detailed Directory Service Replication
Account Logon
  Credential Validation
  Kerberos Ticket Events
  Other Account Logon Events
```

Ainsi, il est désormais possible d'activer l'audit « *Object Access* » sur les objets de type « *File System* » ou « *Registry* » uniquement. Les connaisseurs apprécieront…

Il n'est pas (à ma connaissance) possible de séparer les évènements de nature différente dans plusieurs journaux. Si l'audit des paquets autorisés et rejetés par le firewall est activé, le journal sécurité risque donc de tourner assez rapidement… au détriment d'autres évènements moins fréquents, tels que les ouvertures de sessions.

Toutefois, un nouveau mécanisme de « souscription » permet de récupérer, via WMI, des logs filtrés sur une machine distante. Pour un usage plus « intensif », Microsoft recommandera sans doute l'achat de Microsoft Audit Collection Server 2007.

La grande nouveauté est que les journaux sont désormais disponibles au format XML, donc interrogeables via des requêtes XPath et facilement analysables par des outils tiers. Pour faciliter encore cette analyse, les évènements Vista ont été renumérotés selon la règle :

```
Evènement Windows Vista = Evènement Windows XP + 0x1000
```

À noter qu'un outil d'analyse de log non adapté à Vista sera donc complètement « aveugle ».

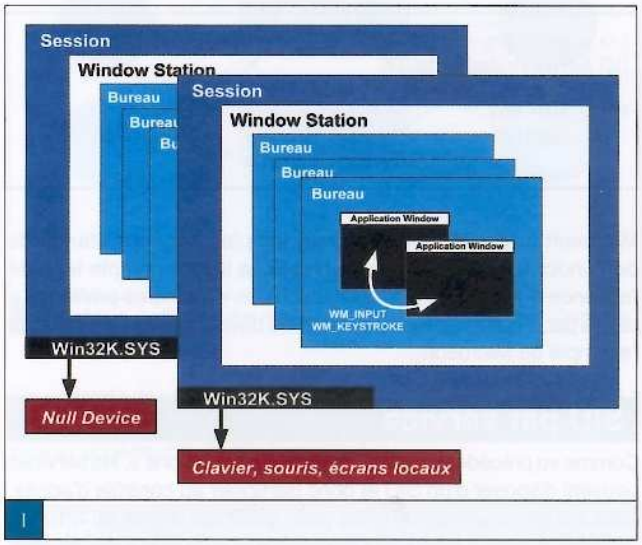
Services

À tout seigneur tout honneur, Cyril Voisin [5] de Microsoft France a réalisé une très bonne série d'articles sur le sujet et je ne ferai que résumer les principales nouveautés.

Isolation de la « Session 0 »

Tout d'abord, un rappel succinct sur le modèle de gestion des fenêtres dans Windows. Au plus haut niveau, se trouve la « Session ». Pour ne pas confondre « Session » avec « Session de Logon » [6], on trouve dans la littérature Microsoft le nom de « *Terminal Server Session* », bien que la « Session 0 » soit la console du système. Une « Session » contient une ou plusieurs « *Window Station* ». Chaque « Window Station » contient à son tour un ou plusieurs « *Desktop* »[3], dans lesquels sont affichées les fenêtres.

Dans Windows XP SP2, il existe 3 bureaux par défaut dans la Window Station « WinSta0 » : le bureau de l'utilisateur, le bureau affiché lorsqu'on appuie sur [Ctrl]+[Alt]+[Suppr] (géré par WinLogon), et le bureau de l'écran de veille.



Les présentations étant faites, entrons maintenant dans le vif du problème. Il existe une classe d'attaque locale permettant une élévation de privilèges, baptisée « *Shatter Attacks* » et récurrente dans les logiciels tiers. Cette attaque est possible, car des processus exécutés sous le compte SYSTEM peuvent afficher des fenêtres sur le bureau de l'utilisateur, et donc potentiellement recevoir des messages « dangereux »[4]. Ces attaques ont déjà été largement traitées dans la littérature [7].

Ce problème est très difficile à corriger pour Microsoft. Il nécessiterait de positionner des ACL sur les messages, ce qui poserait des problèmes de performance et nécessiterait une modification profonde de l'OS. Une solution intermédiaire proposée dans Windows XP SP2 consiste à positionner la clé de base de registres *NoInteractiveServices* pour empêcher les services d'envoyer des fenêtres à l'utilisateur.

[3] *Windows supporte donc nativement le multi-bureau, et même le multi-bureau multiutilisateur, comme la plupart des systèmes *nix.*

[4] *Tel que WM_TIMER, qui permet d'appeler une fonction de callback arbitraire à l'expiration d'un timer.*

La solution retenue dans Vista est encore plus radicale : l'utilisateur n'a désormais plus accès à la « Session 0 », c'est-à-dire que l'utilisateur de la console ouvre en fait une session *Terminal Server* en local ! Ce qui est finalement assez cohérent avec la fonction *Fast User Switching* qui fonctionne sur le même principe.

Je laisse le soin au lecteur de découvrir les détails d'implémentation dans la documentation (ou dans l'exemple de code disponible ici [8]). Afin d'éviter les blocages, Microsoft a quand même fourni le service « UI0Detect » capable de détecter une activité interactive dans la session 0 (ex. : boite de dialogue) et de notifier l'utilisateur.

Principe du moindre privilège

Un service peut désormais spécifier explicitement les privilèges minimums qu'il requiert pour fonctionner, même s'il s'exécute sous un compte de service plus privilégié. Note : ce mécanisme ne peut servir qu'à diminuer ses privilèges, pas à les augmenter au-delà des possibilités du compte de service !

Prenons par exemple le service « Client DHCP » : il ne requiert que 2 privilèges.

```
C:\> sc qprivs dhcp
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: dhcp
        PRIVILEGES      : SeChangeNotifyPrivilege
                        : SeCreateGlobalPrivilege
```

Microsoft a réalisé ce travail pour tous les services standards de Windows, ce qui n'est pas une mince affaire compte tenu de l'absence d'outil d'aide à l'identification des « moindres privilèges » requis par un service. Reste à voir si les développeurs tiers suivront l'exemple de Microsoft…

SID par service

Comme vu précédemment au chapitre « permissions », les services peuvent disposer d'un SID et donc participer au contrôle d'accès. Ceci est principalement utilisé pour gérer les règles du firewall.

Un service possède un SID dès lors qu'il est de type *Restricted* (3) ou *Unrestricted* (1), la valeur par défaut étant *None* (0). La commande permettant de changer le type d'un service est `sc sidtype`.

La commande permettant d'obtenir le SID assigné à un service est la suivante :

```
C:\> sc showsid dhcp

NAME: dhcp
SERVICE SID: S-1-5-80-2940520708-3855866260-481012779-327648279-1710889582
```

Firewalling

Le dernier point concerne la définition des règles de firewalling applicables aux services. Là encore, Microsoft a fourni un travail important sous le nom de code *Windows Service Hardening* : la plupart des services sont protégés par des règles les empêchant de communiquer sur le réseau à tort et à travers.

Un point intéressant à noter est que ces règles ont été définies « en dur » et ne sont pas modifiables ni par les interfaces graphiques, ni par les API disponibles. Elles peuvent toutefois être consultées dans les clés de base de registres :

```
HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\
RestrictedServices\Static
```

Par exemple, pour le service DHCP, on trouve les règles suivantes, n'autorisant que DHCPv4 et DHCPv6 :

```
V2.0|Action=Allow|Dir=Out|LPORT=68|RPort=67|Protocol=17|App=%SystemRoot%\
system32\svchost.exe|Svc=DHCP|Name=@%SystemRoot%\system32\dhcpcsvc.dll|[...]

V2.0|Action=Allow|Dir=In|LPORT=68|RPort=67|Protocol=17|App=%SystemRoot%\
system32\svchost.exe|Svc=DHCP|Name=@%SystemRoot%\system32\dhcpcsvc.dll|[...]

V2.0|Action=Allow|Dir=In|LPORT=546|RPort=547|Protocol=17|App=%SystemRoot%\
system32\svchost.exe|Svc=DHCP|Name=@%SystemRoot%\system32\dhcpcsvc.dll|[...]

V2.0|Action=Allow|Dir=Out|LPORT=546|RPort=547|Protocol=17|App=%SystemRoot%\
system32\svchost.exe|Svc=DHCP|Name=@%SystemRoot%\system32\dhcpcsvc.dll|[...]

V2.0|Action=Block|Dir=In|App=%SystemRoot%\system32\svchost.exe|Svc=DHCP|[...]

V2.0|Action=Block|Dir=Out|App=%SystemRoot%\system32\svchost.exe|Svc=DHCP|[...]
```

Les auteurs de services tiers sont invités à écrire leurs propres règles via l'API *INetFwServiceRestriction* (différente de l'API *INetFwRule* plutôt destinée aux applications qu'aux services). Voici un exemple de script VBS proposé par Cyril Voisin et autorisant la connexion d'un service donné vers n'importe quelle machine sur le port TCP/8080. Sur le papier, ça a l'air simple ; il reste à voir combien d'auteurs de services tiers fourniront un profil de connexion…

```
Dim fwPolicy2
Set fwPolicy2 = CreateObject("HNetCfg.FwPolicy2")

Dim ServiceRestriction
Set ServiceRestriction = fwPolicy2.ServiceRestriction

ServiceRestriction.RestrictService ServiceName, ProgramName, TRUE, FALSE

Dim CurrentRule
set CurrentRule = CreateObject("HNetCfg.FwRule")
CurrentRule.Name = "MySvc network restriction"
CurrentRule.ApplicationName = ProgramName
CurrentRule.ServiceName = ServiceName
CurrentRule.Protocol = 6
CurrentRule.RemotePorts = 8080
CurrentRule.Direction = NET_FW_RULE_DIR_OUT
CurrentRule.Enabled = TRUE

ServiceRestriction.Rules.Add CurrentRule
```

Gestion des processus

UAC, UIPI et autres TLA[5]

Comment parler de Vista sans dire un mot de la fonction *User Account Control* (UAC), qui a tant fait parler d'elle ? [9]

Pour présenter rapidement cette fonction, il faut tout d'abord dire qu'un utilisateur Windows Vista, même membre du groupe « administrateurs », n'a pas les droits « administrateurs » actifs par défaut.

Il s'agit d'un changement de paradigme énorme pour Microsoft. Jusqu'à présent, les utilisateurs devaient être « administrateurs » pour pouvoir réaliser certaines tâches simples, telles que monter un tunnel VPN.

Windows 2000 a introduit la fonction « exécuter en tant que », qui s'avère peu pratique, car l'utilisateur doit connaître les tâches qui vont nécessiter une élévation *avant de les exécuter*.

Windows XP a exploré le concept de délégation de tâches via de nouveaux groupes (tels que *Network Operators*) ou de nouveaux droits. Néanmoins, cette voie est sans issue pour au moins deux raisons :

- Il est très difficile d'identifier précisément toutes les tâches et leur périmètre technique (en termes d'objets sécurisables accédés).
- Les éditeurs tiers ne se sont jamais sentis obligés d'écrire des applications capables de s'exécuter sans droits « administrateurs ». C'est le cas, par exemple, de toutes les applications qui écrivent dans leur répertoire de programme (pour y enregistrer des journaux, des configurations ou des mises à jour par exemple).

Au final, beaucoup d'utilisateurs Windows sont aujourd'hui « administrateurs » de leur poste. Ce qui permet au premier site malveillant venu d'installer des *rootkits* en mode noyau à travers une simple faille ANI par exemple.

Dans ces conditions, Microsoft a changé son fusil d'épaule et propose une logique inspirée de Mac OS X et Linux à travers UAC. Il s'agit de signaler à l'utilisateur *le moment où il a besoin des droits administrateurs*, ce qui permet de bloquer le comportement potentiellement anormal d'une application (sous réserve de comprendre ce qui se passe avant de cliquer sur « oui », bien entendu...).

De nombreuses critiques se sont élevées contre UAC, telles que « on passe son temps à cliquer sur oui ». Cela démontre simplement le nombre d'applications tierces « mal programmées »...

La « faille » trouvée par Joanna Rutkowska est plus astucieuse : il s'avère que tous les programmes dont le nom contient *setup* ou *install* déclenchent automatiquement un *prompt* UAC. C'est une fonctionnalité de compatibilité qui peut être désactivée en entreprise via une GPO[6] et dont le « risque » associé semble assez limité.

Niveaux d'intégrité

Le niveau d'intégrité d'un processus est un label qui peut prendre 4 valeurs : bas, moyen, haut et système. Le niveau par défaut est attaché au fichier exécutable du processus sous forme d'ACE[7]. Il peut être consulté grâce à la commande `ICACLS` ou aux utilitaires `AccessChk` de SysInternals et `CHML` de Mark Minasi.

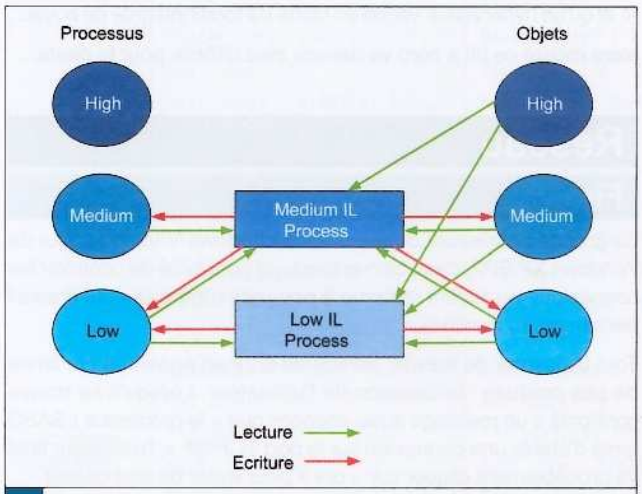
Par défaut, un processus sans label est de niveau moyen, tandis qu'un processus exécuté en tant qu'administrateur est de niveau haut.

Le concept de niveau d'intégrité est intéressant, il s'agit d'un glissement dans le modèle de sécurité Windows du contrôle d'accès *par utilisateur* au contrôle d'accès *par processus*.

À noter qu'un processus possède des droits d'accès, il aurait donc été possible d'implémenter ce mécanisme par des ACL. Mais les processus appartenant à l'utilisateur, il lui aurait également été possible de modifier ces droits d'accès, ce qui n'a pas de sens lorsque l'utilisateur *ne fait pas confiance aux processus qu'il exécute* (bienvenue dans le Web 2.0).

Les niveaux d'intégrité permettent de bloquer tout ou partie des 3 actions suivantes : lecture, écriture et exécution, entre deux processus, ou un processus et un objet, de niveaux d'intégrité différents. Par objet, on entend « tout objet sécurisable qui n'est pas un processus ou un *thread* ». Cette définition s'applique en particulier aux fichiers et aux clés de base de registres.

Le schéma des relations autorisées « par défaut » est donné ci-dessous.



2 *Schéma « officiel » des relations par défaut entre niveaux d'intégrité*

On constate dans ce schéma que les processus de niveau « bas » peuvent, malgré tout, lire des données dans des processus de niveau supérieur (fuite d'information).

PMP

Protected Media Path est une fonction principalement destinée à la gestion des droits numériques (DRM). Un processus utilisateur « protégé » (au sens de PMP) ne peut pas être accédé par un autre processus – c'est-à-dire que la fonction `OpenProcess()`, mais également la fonction plus « bénigne » `NtQueryInformationProcess()` renvoient « accès refusé ».

Il ne s'agit pas d'une permission (ACL) positionnée sur le processus, ni d'un niveau d'intégrité dédié, mais bien d'une implémentation particulière dans le noyau.

Alex Ionescu a fait grand bruit en prétendant avoir « cassé la DRM de Microsoft » **[10]**. En pratique, cette protection est triviale à contourner, puisqu'il s'agit d'un bit dans le jeton de processus, documenté dans les symboles de débogage :

[5] *Three Letters Acronym*
[6] *Group Policy Object*
[7] *Access Control Entry*

```
kd> dt _EPROCESS
[…]
   +0x224 Flags2           : Uint4B
[…]
   +0x224 ProtectedProcess : Pos 11, 1 Bit
[…]
```

Cette protection peut sembler triviale, mais il faut la replacer dans le contexte de Windows Vienna, le successeur de Windows Vista (attendu pour 2009). Si on imagine :

⇨ que la majorité des drivers ont été déplacés en espace utilisateur ;

⇨ que les quelques drivers noyau sont signés par Microsoft après vérification du code, et révocables via Windows Update ;

⇨ et qu'un hyperviseur vérifie en tâche de fond l'intégrité du noyau ;

alors mettre ce bit à zéro va devenir plus difficile pour le pirate…

Réseau

Firewall

La grande nouveauté du firewall de Windows Vista sur celui de Windows XP SP2, c'est, bien entendu, la possibilité de contrôler les connexions *sortantes* (comme à peu près n'importe quel *firewall personnel* du marché).

Tout utilisateur de firewall personnel connaît également la limite de ces produits : la décision de l'utilisateur. Lorsqu'il se trouve confronté à un message aussi abscons que « le processus LSASS tente d'établir une connexion sur le port TCP/88 », l'utilisateur final va probablement cliquer sur « oui » pour éviter de tout casser.

C'est pourquoi Microsoft a fourni un effort de pré-configuration des applications système, pour chacun des types de réseaux prédéfinis (public, privé et domaine). On parle ici de centaines de règles… que vous trouverez listées et commentées dans un document publié par Symantec [11].

Malgré ses 116 pages, ce document reste une référence en la matière et lève quelques problèmes mineurs, mais néanmoins intéressants tels que :

⇨ Les règles de firewall persistantes (dans la plupart des cas, Windows Vista active et désactive les règles adéquates en fonction des services dont souhaite disposer l'utilisateur – mais certaines règles ne « s'effacent » pas).

⇨ Quelques ports toujours ouverts dans les configurations par défaut, comme le port TCP/5357 (utilisé par le service de découverte réseau).

Muni de ces informations, il est par exemple possible d'obtenir une réponse SYN/ACK, RST et « pas de réponse » à un SYN Scan, ce qui permet de *fingerprinter* un Windows Vista très précisément.

On notera que le firewall Windows Vista supporte IPv4 comme IPv6.

Autres protocoles

Les couches réseau ont été profondément modifiées dans Windows Vista. Parmi les changements notables, on peut citer une pile IP entièrement réécrite, supportant nativement IPv4 et IPv6 ; les protocoles SMBv2 et RDP 6.0, etc. La pile IPv6 est active par défaut, et utilisée préférentiellement par les applications ; elle est également capable d'utiliser automatiquement de nombreux mécanismes d'encapsulation IPv6 sur IPv4 (ISATAP, 6to4 et surtout Teredo) – dans ces conditions, il n'est pas rare de rencontrer des machines Windows Vista visibles sur l'Internet v6 mondial, même derrière un firewall d'entreprise…

De nouveaux protocoles ont également été développés : LLTD (*Link Layer Topology Discovery*), PNRP (*Peer Name Resolution Protocol*), LLMNR (*Link Local Multicast Name Resolution*), PNM (*People Near Me*), etc. Sans entrer dans les détails, tous ces protocoles servent de support aux fonctions *Peer-to-Peer* et configuration réseau automatique (l'équivalent du protocole Apple *Bonjour*) de Windows Vista.

Il est clair que la maturité de ces nouveaux protocoles et/ou de ces nouvelles implémentations va être un challenge pour la sécurité de Windows Vista. Lors des phases de développement préliminaires (Bêta puis RC), de nombreuses failles TCP/IP « classiques » (ex. : *Land Attack*) ont été redécouvertes dans la pile IP de Windows Vista. A priori, toutes les attaques connues ont été corrigées dans la version finale (inutile donc de ressortir vos outils d'attaque Windows 98 ;).

De même, des attaques contre le protocole LLTD ont été présentées par Symantec. Bien que relativement simple et bien programmé (donc peu sujet aux débordements de buffer), il reste vulnérable à des attaques en *spoofing* (il ne faut accorder aucune confiance à la topologie découverte, les éléments ne s'authentifiant pas mutuellement).

Encore une fois, toutes ces attaques sont clairement détaillées dans le papier de Symantec, « *Windows Vista Attack Surface* » [11].

Sous le capot…

Principes de développement

Windows Vista applique les principes du *Security Development Lifecycle* (SDL), une méthodologie proposée par Microsoft pour réduire le nombre de failles dans le code et leur impact potentiel. Cette méthodologie a été décrite dans un livre [12]. On peut en résumer les grandes lignes par :

⇨ des développeurs sensibilisés à la sécurité ;

⇨ des revues de code effectuées par des tiers ;

⇨ des outils de développement sûrs (API sûre, analyse statique, *Security Annotation Language*) ;

⇨ une surface d'exposition minimale par défaut (configuration par défaut fiable, services activés à la demande).

Du classique, mais que seul Microsoft semble mettre en œuvre efficacement dans l'industrie logicielle d'aujourd'hui…

Protections génériques contre l'exploitation de failles

DEP

Comme Windows XP SP2, Windows Vista exploite la capacité matérielle des processeurs récents[8] à disposer de pages mémoires non exécutables. Cette fonction, appelée DEP

(*Data Execution Prevention*), est en mode « OptIn » par défaut (seuls les exécutables du système sont protégés).

On notera qu'en mode « OptOut » (plus rigoureux), des exceptions automatiques restent appliquées, comme les logiciels protégés par StarForce ou ASPack. Ceux-ci sont identifiés d'après les noms des sections exécutables.

À noter que DEP n'est pas disponible en software uniquement, contrairement à W^X sous OpenBSD ou PaX sous Linux. Le support matériel par le processeur est nécessaire.

ASLR

Windows Vista met en place également le mécanisme connu sous Linux sous le nom d'*Address Space Layout Randomization* (ASLR). Toutes les zones mémoires sont distribuées aléatoirement de la manière suivante, selon le document Symantec [13] :

- sections du programme : 256 possibilités, changent à chaque *reboot* ;
- piles (*stacks*) : 16384 possibilités, changent à chaque exécution ;
- tas (*heaps*) : 32 possibilités, changent à chaque exécution ;
- PEB[9] : 16 possibilités, change à chaque exécution.

Bien que ce papier présente également des défauts dans le générateur d'aléas de Windows Vista, on peut dire que l'exploitation « fiable du premier coup » d'une faille de type *buffer overflow* reste largement hypothétique… sauf à pouvoir obtenir des informations sur le *layout* du processus distant, ce qui est parfois le cas avec les failles RPC.

/GS

Windows Vista a été massivement recompilé avec l'option « /GS » de Visual Studio. Cette option rajoute un cookie dans la pile, rendant facilement détectable l'écrasement d'une adresse de retour en cas de débordement de buffer.

Cette protection était déjà présente dans Windows XP SP2 (c'est d'ailleurs une nouveauté majeure du SP2 par rapport aux versions antérieures), et son efficacité n'est plus à démontrer.

Là encore Symantec a produit une étude [14] des binaires protégés par « /GS » dans Windows Vista (Symantec et Microsoft, une longue histoire d'amour :). Une liste de fichiers exécutables (EXE, DLL et drivers) non protégés par « /GS » est donnée. On notera que les drivers tiers intégrés dans le système (tel que le driver vidéo NVIDIA) sont généralement moins protégés.

Protection du tas

Comme dans Windows XP SP2, l'allocateur dynamique de mémoire est protégé. Néanmoins, la protection va plus loin que le simple cookie de Windows XP SP2, puisque, dans Windows Vista, les informations du *chunk* mémoire sont *XOR-ées*. Pour le vérifier, lançons le programme de test fourni par Kostya Kortchinksy dans son article sur le heap Windows (dont vous trouverez une copie ici [15]).

	Windows XP SP2	Windows Vista
Première exécution	*** HeapAlloc *** Self size : 65 Previous size : 8 Segment index : 240 Flags : 1 Unused bytes : 8 Tag index : 0 *** HeapFree *** Self size : 304 Previous size : 8 Segment index : 240 Flags : 16 Unused bytes : 8 Tag index : 0 Next chunk : 0x00360178 Previous chunk : 0x00360178	*** HeapAlloc *** Self size : 58482 Previous size : 29332 Segment index : 229 Flags : 57 Unused bytes : 0 Tag index : 8 *** HeapFree *** Self size : 58751 Previous size : 32661 Segment index : 229 Flags : 57 Unused bytes : 0 Tag index : 0 Next chunk : 0x003300c4 Previous chunk : 0x003300c4
Deuxième exécution	*** HeapAlloc *** Self size : 65 Previous size : 8 Segment index : 222 Flags : 1 Unused bytes : 8 Tag index : 0 *** HeapFree *** Self size : 304 Previous size : 8 Segment index : 222 Flags : 16 Unused bytes : 8 Tag index : 0 Next chunk : 0x00360178 Previous chunk : 0x00360178	*** HeapAlloc *** Self size : 53124 Previous size : 1619 Segment index : 89 Flags : 149 Unused bytes : 0 Tag index : 8 *** HeapFree *** Self size : 52873 Previous size : 2898 Segment index : 89 Flags : 149 Unused bytes : 0 Tag index : 0 Next chunk : 0x000200c4 Previous chunk : 0x000200c4

On constate rapidement :

- Que l'adresse de base du heap est aléatoire sous Windows Vista, contrairement à Windows XP SP2.
- Que les informations d'en-tête du chunk sont également « chiffrées » avec une clé aléatoire.

SafeSEH

Comme dans Windows XP SP2, les gestionnaires d'exception peuvent être déclarés statiquement à la compilation afin d'empêcher le transfert de l'exécution vers un gestionnaire inconnu. Mais, contrairement à Windows XP SP2, une bonne partie des exécutables livrés avec Windows Vista sont compilés avec cette protection.

[8] *Technologie NoeXecute (NX) chez AMD et eXecutionDisable (XD) chez Intel.*
[9] *Process Environnement Block*

À noter qu'un trampoline via un gestionnaire autorisé reste possible. Cette technique a permis l'exploitation fiable de la faille DNS RPC (MS07-029) sur Windows 2003 SP0/SP1/SP2 (un pur bijou d'exploitation, soit dit en passant).

Nouveau CRT (C RunTime)

Microsoft a également revu l'API C offerte aux développeurs pour leur éviter les erreurs « classiques ». Ainsi, le spécificateur de format « %n » (très rarement utilisé à vrai dire) n'est plus supporté par les fonctions de la famille `printf()`. Ceci n'empêche pas les attaques de type *format string* (en cas de faille, il reste possible de lire la mémoire du processus distant), mais leur exploitation (il n'est plus possible d'écrire dans cette mémoire).

De même, les API considérées comme dangereuses (ex. : `strcpy()`) ont été dépréciées et remplacées (quand c'est possible) par des versions « sûres » (ex. : `strcpy_s()`) **[16]**. Les versions « sûres » prennent en paramètre la taille du buffer destination.

Contournement des protections techniques

Il est inutile de présenter ici en détail les nombreux contournements qui ont été développés pour toutes ces technologies défensives.

⇨ DEP est traditionnellement contourné par un enchaînement de « retours dans la LibC ». À noter que DEP peut être désactivé pour un processus en cours d'exécution par un appel à `NtSetInformationProcess()` – le code nécessaire se trouvant déjà dans NTDLL, comme montré par le projet Metasploit **[17]**.

⇨ ASLR peut être défait par toute attaque permettant de réduire le hasard : fuite d'information sur le layout du processus distant, saturation de la mémoire avec des motifs réguliers, etc.... ou tout simplement en trouvant une adresse fixe dans le processus distant (ex. : code non relogeable) !

⇨ « /GS » peut être contourné en écrasant un pointeur de fonction avant l'adresse de retour ou en écrasant un gestionnaire d'exception dans la pile. Il faut noter également que la protection de pile n'est ajoutée que sur les fonctions présentant des caractéristiques particulières (ex. : buffer local contenant des éléments de taille unitaire 1 ou 2 octets).

⇨ La protection du tas semble plus difficile à contourner. L'approche utilisée aujourd'hui consiste à essayer de contrôler le layout du tas pour amener un chunk mémoire intéressant (ex. : contenant un pointeur de fonction) juste derrière le chunk débordant.

Conclusion sur les protections techniques

Considérée individuellement, chaque protection peut être contournée plus ou moins facilement. Mais si 2 protections ou plus sont simultanément activées (ex. : DEP + ASLR), alors l'exploitation d'une faille de type buffer overflow devient très difficile.

Le cœur du système d'exploitation est relativement bien protégé de ce point de vue. On ne peut pas en dire autant des applications tierce partie actuellement, compte tenu des pré-requis :

⇨ avoir été compilé avec Visual Studio 2005 SP1 (pour bénéficier de l'ASLR via l'option « /DYNAMICBASE » du compilateur) ;

⇨ avoir été compilé avec les options « /GS », « /SAFESEH » et éventuellement « /NXCOMPAT » ;

⇨ ne pas être protégé par un *packer* logiciel ;

⇨ utiliser la bibliothèque C fournie par Microsoft (ce qui exclut les applications Delphi, Cygwin, etc.).

Autant dire que les applications tierces vérifiant ces pré-requis se comptent sur les doigts d'une main (la première condition étant déjà très sélective). C'est ainsi qu'à la conférence RSA Security 2007, un bogue affectant le logiciel de sauvegarde CA BrightStor a pu être exploité avec succès sur Windows Vista **[18]**.

Les failles existantes

Les failles connues actuellement et affectant Windows Vista se classent en 3 catégories.

1▶ Les failles « à l'ancienne », comme la faille « ANI » (MS07-017).

Il s'agit d'un « simple » buffer overflow, qui a échappé à toutes les protections techniques :

⇨ DEP est désactivé sur Internet Explorer pour améliorer la compatibilité avec les *plugins*.

⇨ ASLR est inutile, car il est possible d'écraser partiellement l'adresse de retour et donc de revenir à un emplacement relatif à l'adresse de début du programme. Il est également possible de tester les 256 valeurs possibles, grâce à un gestionnaire d'exception qui intercepte silencieusement les violations d'accès.

⇨ « /GS » n'a pas ajouté de cookie dans la pile de la fonction vulnérable, car elle ne contient pas de buffer (la copie s'effectue dans une structure).

Microsoft fait son mea culpa sur le blog du SDL **[19]**. La seule protection qui a fonctionné est le « mode protégé » d'Internet Explorer, qui limite les possibilités d'actions « post-exploitation » (par défaut le *shellcode* s'exécute en niveau d'intégrité bas).

2▶ Les failles techniques complexes et indétectables par des outils.

Parmi ces failles, on peut citer la faille `NtRaiseHardError()`. Cette fonction contient une faille de type *double free* (par des chemins d'exécution non triviaux) qu'il est possible d'exploiter localement en la combinant à une fuite d'information sur la mémoire.

Aucun outil automatique ne sait actuellement détecter des failles liées à des évènements décorrélés, et la détection par un humain n'est pas triviale non plus.

3▶ Les failles non techniques.

On peut citer par exemple la « fameuse » faille qui consiste à jouer un enregistrement sonore dans l'espoir de déclencher la fonction commande vocale de Windows Vista !

L'existence de ces failles démontre au moins que l'industrie de la sécurité ne va pas disparaître avec Windows Vista :)

Conclusion

Malgré sa taille, cet article n'est qu'une brève introduction à la sécurité Windows Vista... Des technologies (telles que CryptoNG ou AuthIP) et des produits de sécurité natifs (tels que BitLocker ou Windows Defender) auraient également mérité d'être étudiés.

Pour conclure, on peut raisonnablement affirmer que Windows Vista représente une réelle avancée en matière de sécurité – à la mesure de l'enjeu auquel est confronté Microsoft.

On pourrait ergoter sur la finition de l'ensemble[10]. Mais le réel problème que va poser Windows Vista aux entreprises est la maîtrise d'un tel niveau de complexité (le fameux TCO[11]). Quand on sait que la plupart des grandes entreprises françaises achèvent encore péniblement leur migration vers un domaine Windows 2000, on peut se demander qui possède les ressources informatiques pour contrôler la dizaine de milliers d'options de configuration, vérifier la compatibilité des pilotes, détecter les tunnels Teredo en bordure, et bloquer les nombreuses connexions intempestives vers des sites Microsoft plus ou moins identifiés.

Liens

[1] *Uninformed: Subverting PatchGuard version 2*,
http://uninformed.org/?v=6&a=1&t=sumry

[2] *Windows Resource Protection on Windows Vista*,
http://msdn2.microsoft.com/en-us/library/aa372868.aspx

[3] *Credential management*,
http://www.freepatentsonline.com/7210167.html

[4] BORDES (Aurélien), « Secrets d'authentification sous Windows », SSTIC 2007,

[5] VOISIN (Cyril), *Services isolation in Session 0 of Windows Vista and Longhorn Server*, série de 5 articles,
http://blogs.technet.com/voy/archive/2007/02/23/services-isolation-in-session-0-of-windows-vista-and-longhorn-server.aspx
http://www.sstic.org/SSTIC07/programme.do#BORDES

[6] Utilitaire « LogonSessions » de SysInternals,
http://www.microsoft.com/technet/sysinternals/utilities/LogonSessions.mspx

[7] MOORE (Brett), « *Shattering By Example* »,
https://www.blackhat.com/presentations/bh-usa-04/bh-us-04-moore/bh-us-04-moore-whitepaper.pdf

[8] *WinSta0 Isolation Explained*,
http://bartdesmet.net/blogs/bart/archive/2007/03/05/windows-vista-winsta0-isolation-explained.aspx

[9] RUTKOWSKA (Joanna), « *Running Vista Every Day!* »,
http://theinvisiblethings.blogspot.com/2007/02/running-vista-every-day.html

[10] IONESCU (Alex), « *Why Protected Processes Are A Bad Idea* »,
http://www.alex-ionescu.com/?p=34

[11] *Windows Vista Network Attack Surface Analysis*,
http://www.symantec.com/avcenter/reference/Vista_Network_Attack_Surface_RTM.pdf

[12] *The Security Development Lifecycle*,
http://www.microsoft.com/mspress/books/8753.aspx

[13] *An Analysis of Address Space Layout Randomization on Windows Vista*,
http://www.symantec.com/avcenter/reference/Address_Space_Layout_Randomization.pdf

[14] *Analysis of GS protections in Microsoft Windows Vista*,
http://www.symantec.com/avcenter/reference/GS_Protections_in_Vista.pdf

[15] Heap de Windows : structure, fonctionnement et exploitation,
https://www.securinfos.info/jerome/DOC/heap-windows-exploitation.html

[16] *Security Enhancements in the CRT*,
http://msdn2.microsoft.com/en-us/library/8ef0s5kh(VS.80).aspx

[17] *Bypassing Windows Hardware-enforced Data Execution Prevention*,
http://www.uninformed.org/?v=2&a=4

[18] *CA backup bug exploitable on Vista*,
http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1242436,00.html

[19] *Lessons learned from the Animated Cursor Security Bug*,
http://blogs.msdn.com/sdl/archive/2007/04/26/lessons-learned-from-the-animated-cursor-security-bug.aspx

[10] *Par exemple, le mode protégé d'Internet Explorer empêche de copier/coller une URL vers le bloc-notes...*
[11] *Total Cost of Ownership, voir à ce sujet « A Cost Analysis of Windows Vista Content Protection » de Peter Gutmann.*

25

RMS : évitez la fuite de vos informations

Pour contrôler la fuite des informations sensibles, comme des plans, des photos, des informations liées aux ressources humaines, les codes sources d'un logiciel, des solutions existent sur le marché sous le nom de « data loss prevention ». Dans le cadre du management de l'information, cet article propose d'étudier la nouvelle approche de Microsoft baptisée RMS (Rights Management Services).

mots clés : ***RMS / chiffrement / Windows Vista / XrML / protection fuite information***

1. Introduction

Comment éviter que les fichiers sensibles ne sortent de l'entreprise ? Avez-vous déjà évalué le risque de diffusion des données de votre entreprise vers l'extérieur, vers un concurrent ou utilisées par un ver robot-spammeur ? Les études fleurissent sur le Net et vont toutes dans le même sens : la fuite volontaire, malveillante ou accidentelle des informations confidentielles est un problème d'actualité pour les entreprises.

L'idée est de labelliser comme « sensibles » certains fichiers manipulés sur votre système d'information. Certaines actions seront alors contrôlées comme la copie de fichier (sur clé USB), le copier-coller d'une partie d'un document, une capture d'écran, l'envoi d'un mail avec le document en pièce jointe, la durée de validité d'un document ou tout simplement une impression.

La première solution évidente est de positionner des droits de lecture et écriture, voire un mot de passe. C'est insuffisant… La deuxième solution est le chiffrement. Mais ce même document peut être archivé en clair sur un autre répertoire partagé ! Toujours insuffisant… Une fois le fichier ouvert et lisible sur l'écran de l'utilisateur, tout est permis : l'envoyer par mail, l'imprimer et laisser traîner la feuille sur son bureau, perdre sa clé USB où tout est archivé… Sans compter les utilisateurs nomades, les commerciaux ou les ingénieurs travaillant à domicile.

Microsoft propose l'IRM (*Information Rights Management*), une technologie de protection persistante des fichiers qui permet de protéger la propriété intellectuelle numérique contre une utilisation non autorisée. Seuls les documents bureautiques (depuis la version MS-Office 2003) sont concernés. Ces fonctionnalités sont rendues possibles sur son propre système d'information grâce à l'installation d'un service appelé RMS (*Rights Management Services*).

Nous allons voir dans cet article comment RMS implémente ces nouvelles fonctionnalités, avec ses avantages et ses limites.

Au niveau juridique, la confidentialité des données en entreprise est réglementée, notamment par l'utilisation de la cryptologie depuis la LCEN **[1]** en France et Sarbanes-Oxley aux États-Unis.

L'employeur doit fournir des moyens pour protéger la confidentialité des documents, mais doit également pouvoir accéder aux fichiers provenant ou non de « la sphère privée » du salarié sur un ordinateur de l'entreprise **[1a]**. Nous verrons que RMS répond à la LCEN, mais que certaines précautions sont indispensables lors du déploiement.

2. Solutions du marché

Microsoft se distingue par son approche orientée utilisateur : c'est l'auteur du document qui décide des droits et de son utilisation (publication et consommation). Les autres solutions du marché **[2 à 6]** sont orientées filtrage réseau par mots clés ou mode comportemental, mais ne sont pas liées au format du document ni au système d'exploitation utilisé. Pas de solution *Open Source* équivalente pour l'instant…

Exemples de Solutions	Editeurs	Technologies
Rights Management Services	Microsoft	Chiffrement + droits d'utilisations du document
Data loss prevention host et gateway	Mc Afee	Localisation physique, mots clés, application génératrice
Vericept data loss prevention	Vericept	Localisation physique, mots clés, application génératrice
Content protection suite	WebSense	Localisation physique, mots clés, application génératrice
Vontu 7	Vontu	Localisation physique, mots clés, application génératrice
Digital guardian	Verdasys	Chiffrement + Localisation physique, mots clés, application génératrice

TI *Solutions concurrentes à RMS*

3. Solution Microsoft RMS

3.1 Architecture

Une architecture RMS utilise des technologies de serveur et de client. Les technologies Services Web RMS assurent les fonctionnalités serveur de base :

⇨ La sous-inscription des serveurs fournit des certificats de licences serveur secondaires aux serveurs de licences. Ces certificats permettent aux serveurs de licences d'émettre des licences de publication et des licences d'utilisation.

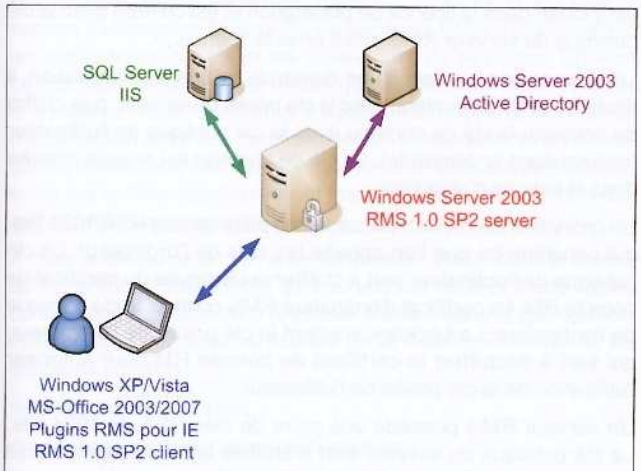
⇨ La certification de compte des entités approuvées distribue les certificats de comptes RM aux utilisateurs (stockés dans une base SQL Server). Les utilisateurs doivent disposer de ces certificats pour pouvoir obtenir des licences de publication et des licences d'utilisation leur permettant de créer et d'utiliser un contenu protégé par RMS.

⇨ La gestion de licences relatives aux informations protégées par RMS.

⇨ Les fonctions d'administration de RMS.

Thierry Martineau
thierrymartineau@yahoo.fr
École Supérieure et d'Application des Transmissions – Rennes

Les services Web RMS sont rendus accessibles par le biais de l'interface SOAP ou de .NET Remoting. RMS utilise les services d'inscription et d'activation hébergés par Microsoft pour fournir une racine commune d'approbation pour le système à son initialisation.



1 *Architecture RMS*

La composante serveur est constituée de serveurs SQL Server, Active Directory, IIS et RMS 1.0 serveur sous Windows 2003 Server. La composante client est constituée du système d'exploitation Windows Vista (natif) ou XP avec client RMS 1.0 et MS-Office 2003/2007.

Les technologies clientes RMS permettent aux utilisateurs du système de créer, de publier et d'utiliser un contenu protégé par RMS. Les applications compatibles RMS sont nécessaires pour créer et pour publier du contenu protégé par RMS. Les applications peuvent être développées spécifiquement pour RMS ou les applications existantes peuvent être réécrites pour fonctionner avec RMS (cf. SDK).

Afin de sécuriser son architecture, RMS s'appuie sur des machines de confiance, des utilisateurs et des droits d'utilisation persistants liés aux documents. Un **certificat d'ordinateur RMS** identifie un ordinateur particulier comme étant approuvé par RMS. Il se trouve sous `%USERPROFILE%\Local Settings\Application Data\Microsoft\DRM\CERT-Machine.drm`

Une **Lockbox** comporte la clé privée d'un ordinateur et un certificat correspondant, qui contient la clé publique de l'ordinateur. Les Lockbox résident sur les ordinateurs clients. Un ordinateur reçoit une Lockbox lors de l'installation et de l'activation du client RMS. Chaque Lockbox est construite sur un identificateur de matériel, afin que la Lockbox soit à la fois unique et liée à un ordinateur particulier.

Un **certificat de comptes RM** identifie un utilisateur particulier comme étant approuvé par RMS. Il se trouve sous `%USERPROFILE%\Local Settings\Application Data\Microsoft\DRM\GIC*`.

Un **certificat de licence client** permet à l'utilisateur de publier du contenu protégé par RMS lorsqu'il est déconnecté du réseau. Il se trouve sous `%USERPROFILE%\Local Settings\Application Data\Microsoft\DRM\CLC*`.

Une **licence de publication** définit les droits d'utilisation d'un document.

Une **licence d'utilisation** permet aux utilisateurs d'utiliser du contenu protégé par RMS sous `%USERPROFILE%\Local Settings\Application Data\Microsoft\DRM\LUF*`.

Un compte utilisateur possède un certificat d'ordinateur unique, un fichier GIC et un fichier CLC, mais plusieurs fichiers LUF (*Licence Utilisateur Final*) pour chaque élément de contenu auquel il accède.

Le client RMS doit être installé et l'ordinateur utilisé doit être activé. Les applications compatibles RMS permettent aux créateurs de contenu d'associer, sous la forme de licences de publication, des droits d'utilisation aux fichiers qu'ils créent afin de contrôler le mode d'utilisation de ce contenu. Les applications compatibles RMS traitent également les informations des fichiers chiffrés et permettent aux utilisateurs d'utiliser le contenu en fonction des autorisations définies dans la licence de publication.

En utilisant le kit de développement logiciel (SDK) pour client RMS, les développeurs peuvent concevoir des applications compatibles RMS qui accordent des licences, publient et utilisent du contenu protégé par RMS.

Les utilisateurs ne disposant pas d'autre application compatible RMS pour utiliser du contenu protégé par RMS dans des messages électroniques ou dans des pages Web peuvent obtenir et utiliser le module complémentaire pour Internet Explorer. Par exemple, les utilisateurs de Microsoft Outlook Web Access (webmail OWA) peuvent s'en servir pour manipuler des emails protégés par RMS.

RMS utilise Active Directory pour les opérations suivantes :

⇨ Assurer l'authentification des utilisateurs. Active Directory fournit les services d'annuaire utilisés pour authentifier les utilisateurs de RMS.

⇨ Résoudre l'adhésion de groupe et les identités de compte d'utilisateur. Active Directory fournit des informations concernant l'adhésion de groupe que RMS utilise pour accorder des licences d'utilisation sur le contenu protégé par RMS, alors que la licence de publication accorde des droits à des groupes plutôt qu'à des comptes d'utilisateurs. Pour réduire le nombre de requêtes LDAP soumises à Active Directory, RMS place les informations obtenues dans un cache local et dans une base de données centralisée de services d'annuaire.

⇨ Stocker l'emplacement de détection des services RMS. Les demandes de service doivent être envoyées à l'URL du module exécutable du service Web accordant la demande. Toutes les demandes de service commencent par une requête Active Directory correspondant à l'URL du service Web de serveur (Server.asmx), qui, à son tour, fournit l'URL appropriée de la demande de service.

Chaque élément de contenu protégé par RMS est publié avec une licence qui stipule ses règles d'utilisation et chaque utilisateur de ce contenu reçoit une licence unique qui lit, interprète et applique ces règles d'utilisation. Dans ce contexte, une licence est un type particulier de certificat.

RMS utilise un vocabulaire XML pour exprimer les droits numériques d'utilisation d'un contenu protégé par RMS : **XrML** (*eXtensible rights Markup Language*). L'interprétation et la gestion des licences sont facilitées par divers systèmes de gestion des droits, qui opèrent conjointement par le biais d'une utilisation commune de la norme XrML. La gestion en ligne des licences

fournit un accès aisé à partir de n'importe quel emplacement. Une fois la licence téléchargée, la gestion des droits est effective en ligne et hors connexion, car les droits accompagnent le fichier dans ses déplacements.

Extrait de l'en-tête d'un fichier Word protégé par XrML :

```
<?xml version="1.0"?>
<XrML version="1.2" xmlns="">
<BODY type="Microsoft Rights Label" version="3.0">
<ISSUEDTIME>2007-05-30T15:31</ISSUEDTIME>
<DESCRIPTOR><OBJECT><ID type="MS-GUID">{D5C53708-820C-4787-A51E-0F2300BA055F}</ID><NAME>LCID
1036:NAME Ne pas distribuer:DESCRIPTION Autorisation actuellement restreinte. Seuls les
utilisateurs spÃ©cifiÃ©s peuvent accÃ©der Ã  ce contenu.; </NAME></OBJECT></DESCRIPTOR>
<ISSUER><OBJECT type="Group-Identity"><ID type="Windows">S-1-5-21-798008851-1132006241-
1500355952-1116</ID><NAME>mtu@misc.net</NAME></OBJECT>
<PUBLICKEY><ALGORITHM>RSA</ALGORITHM><PARAMETER name="public-exponent"><VALUE encoding="int
eger32">65537</VALUE></PARAMETER><PARAMETER name="modulus"><VALUE encoding="base64" size="1
024">fZRY9hbtIanUjQ5dCWXYxXGoPvWYyMcHAQhqResZunqFneWjKp7c9O7eMoyjYk1XRQlup6Cc7GKWcoiWFeMAXqNH
mI3gk65ZHKUOlWrdYoScMB3SAPchqRcbtnjCGyGFiOJKmTwpDLK9qEtqMQcKiUcZYBQQZ5J61Sjd5bJAqeU=</
VALUE></PARAMETER></PUBLICKEY>
```

Le contenu protégé est toujours chiffré. Les certificats et les licences utilisés par RMS sont également susceptibles de contenir du contenu chiffré, qui peut être déchiffré uniquement par une entité appropriée. Tous les serveurs RMS, les ordinateurs clients et les comptes d'utilisateurs ont une paire de clés RSA de 1024 bits. RMS utilise ces clés pour chiffrer la clé de contenu qui figure dans les licences de publication et d'utilisation et pour signer les certificats et les licences RMS. Ce processus garantit que le serveur autorise l'accès uniquement aux utilisateurs et aux ordinateurs habilités.

Clés	Utilisations
Clés de serveur	Clé publique : chiffre la clé de contenu qui figure dans une licence de publication, afin que seul le serveur RMS puisse extraire la clé de contenu et émettre des licences d'utilisation pour cette licence de publication. Clé privée : signe tous les certificats et toutes les licences qui sont émis par le serveur.
Clés d'ordinateur	Clé publique : chiffre une clé privée de certificat de compte RM. Clé privée : déchiffre un certificat de compte RM.
Clés de licence client	Clé publique : chiffre la clé de contenu symétrique dans les licences de publications qu'elle émet. Clé privée : signe les licences de publication qui sont émises localement lorsque l'utilisateur n'est pas connecté au réseau.
Clés de l'utilisateur	Clé publique : chiffre la clé de contenu qui figure dans une licence d'utilisation afin que seul un utilisateur déterminé puisse, par le biais de cette licence, utiliser du contenu protégé par RMS. Clé privée : permet à un utilisateur d'utiliser du contenu protégé par RMS.
Clés de contenu	Chiffre du contenu protégé par RMS lorsque l'auteur le publie.

T2 *Les clés RMS*

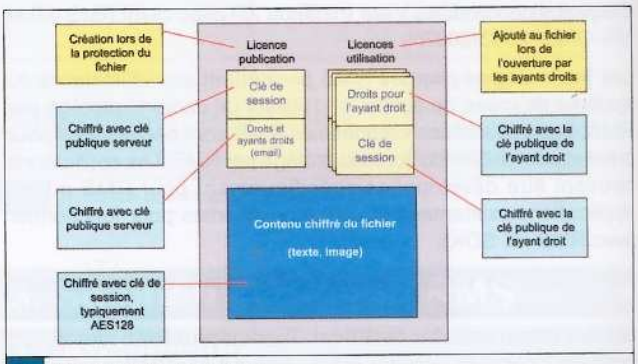
Un utilisateur RMS a une paire de clés RSA de 1024 bits. La paire de clés de l'utilisateur est stockée dans la base de données de configuration RMS, afin qu'un utilisateur donné conserve la même paire de clés dans l'ensemble du système RMS.

Lorsqu'un auteur publie du contenu protégé par RMS, une application compatible RMS crée une clé de contenu symétrique et l'utilise pour chiffrer le contenu. RMS utilise *Advanced Encryption Standard* (AES) pour protéger la clé de contenu. La clé de contenu est incluse dans la licence de publication et est chiffrée avec la clé publique du serveur RMS ayant émis la licence.

Lorsque ce serveur reçoit une demande de licence d'utilisation, il déchiffre la clé de contenu avec la clé privée du serveur, puis chiffre de nouveau la clé de contenu avec la clé publique de l'utilisateur (incluse dans la demande). La clé de contenu est ensuite insérée dans la licence d'utilisation.

Un ordinateur client RMS possède une paire de clés RSA 1024 bits, qui constitue ce que l'on appelle les clés de l'ordinateur. La clé publique de l'ordinateur sert à chiffrer la clé privée du certificat de compte RM. Le certificat d'ordinateur RMS contient la clé publique de l'ordinateur. La Lockbox contient la clé privée de l'ordinateur, qui sert à déchiffrer le certificat de compte RM pour autoriser l'utilisation de la clé privée de l'utilisateur.

Un serveur RMS possède une paire de clés RSA de 1024 bits. La clé publique du serveur sert à chiffrer la clé de contenu qui figure dans une licence de publication, afin que seul le serveur RMS puisse extraire la clé de contenu et émettre des licences d'utilisation pour cette licence de publication. Le certificat de licence serveur contient la clé publique du serveur. La clé privée du serveur sert à signer tous les certificats et toutes les licences qui sont émis par le serveur.



2 *Fichier protégé par RMS*

3.2 Emploi de RMS

RMS utilise différents paramètres de sécurité pour les trois modes suivants :

- Installation : les fichiers RMS sont installés et configurés.
- Déploiement : les sites Web, les répertoires virtuels et les bases de données sont créés et configurés.
- Opérations normales : certains services, comme la certification de compte, exigent une authentification, et d'autres, comme l'activation d'un ordinateur, ne l'exigent pas.

Installation

L'administrateur qui effectue la procédure d'installation doit se connecter avec un compte d'utilisateur membre du groupe de

l'administrateur local. Le mot de passe de ce compte est la clé de voûte de toute la sécurité RMS, puisque ce compte est capable de déchiffrer tous les documents en cas de problème.

Pour plus d'information, vous trouverez le guide d'installation rapide RMS sur le site de Microsoft.

Déploiement

Les composants intervenant dans un système RMS sont le service d'inscription Microsoft, les serveurs RMS d'organisation, les ordinateurs clients et les utilisateurs du système. Chaque composant reçoit un certificat qui établit son identité dans le système. Une hiérarchie d'approbation définit la relation d'approbation entre ces certificats, et, en conséquence, les entités qui les possèdent. Elle définit également la relation d'approbation entre les entités approuvées et les licences qu'elle délivre aux autres entités approuvées.

À chaque niveau de la chaîne de certification, RMS valide la licence ou le certificat, puis vérifie qu'il ou elle est relié(e) à une racine connue de l'approbation à l'intérieur d'une chaîne d'approbation. Pour chaque certificat ou licence de la chaîne, RMS vérifie les éléments suivants :

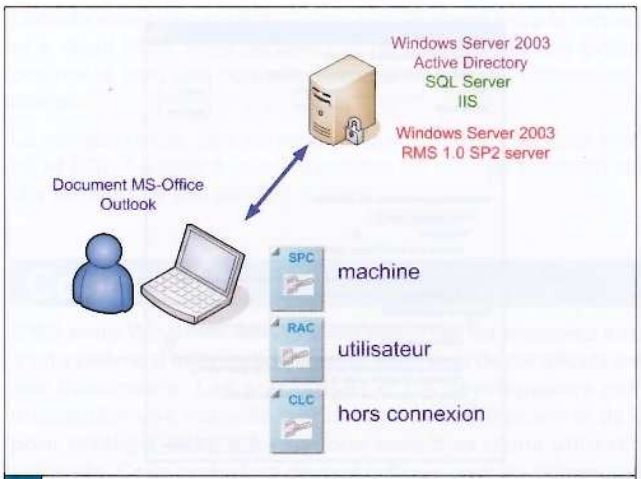
- la validité de son XrML ;
- la validité de la signature de l'émetteur ;
- la conformité de la sémantique de la licence par rapport à l'usage prévu ;
- le respect des conditions imposées (telles que les dates de validité) ;
- la révocation de la licence ;
- la concordance de la clé de la signature de la licence et de la clé de l'émetteur certifié.

L'URL de chaque service doit être publiée afin qu'elle puisse être détectée par les clients et par les autres services qui doivent l'utiliser. Si vous avez activé SSL sur votre serveur RMS, ces URL utilisent le protocole de connexion https://. Lorsqu'un client demande une activation d'ordinateur RMS, il ajoute le nom de fichier du service proxy d'activation à l'URL de la manière suivante : **http://nom_serveur/_wmcs/Certification/Activation.asmx**.

Le client RMS est un fichier Windows Installer (2.3 Mo) et peut être distribué à l'aide d'une infrastructure de distribution logicielle telle que Systems Management Server 2003. Il est également possible de distribuer le client RMS en utilisant un objet Stratégie de groupe (GPO) qui utilise un compte de service avec des droits administratifs.

Activation d'une machine

- Le client RMS génère une paire de clés RSA 1024 bits.
- La clé privée est sécurisée par les CryptoAPI.
- La clé publique est stockée dans le SPC (*Security Processor Certificate*). RMS s'appuie sur un élément externe (identification physique de l'ordinateur) pour ne pas qu'un *trojan* utilise les droits du profil de l'utilisateur pour s'attribuer des droits illicites. On utilise la Lockbox pour rajouter un élément de contrôle des clés. Par contre, rien n'empêche un virus d'utiliser Word en tant qu'objet OLE pour manipuler un fichier en mémoire.
- Le SPC est signé par le client RMS : c'est le 1er certificat créé sur la machine (1 par session utilisateur).



3 *Les certificats sur le client RMS*

Certification du compte utilisateur

- Le client RMS envoie le SPC au serveur RMS.
- L'utilisateur est authentifié par AD.
- Le serveur RMS valide le SPC et envoie le courriel stocké dans AD. L'adresse email est l'identifiant du certificat de l'utilisateur.
- Une paire de clés RSA est générée et stockée dans la base SQL Server.
- La clé privée de l'utilisateur est chiffrée avec la clé publique de la machine.
- Le RAC (*Rights Management Account Certificate*) est créé, l'email et la clé publique y sont ajoutés. Le RAC est signé par le serveur RMS, puis retourné au client.
- L'utilisateur peut maintenant consommer du contenu.

Pour MS-Office, il faut pouvoir utiliser l'information hors connexion avec un autre certificat : le CLC (*Client Licensor Certificate*) qui va contenir la clé publique du serveur. À tout instant, hors ou en ligne, un document est protégé par la clé publique du serveur RMS.

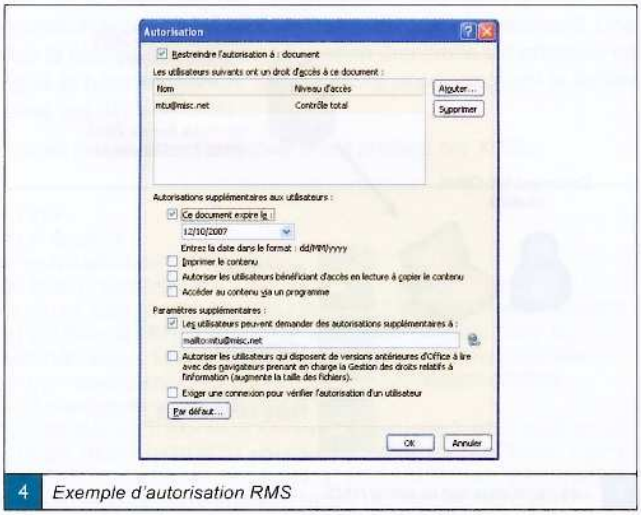
Même principe pour le RAC, il y a génération d'une bi-clé RSA pour le CLC. Sur un ordinateur, il existe donc 3 certificats : le SPC, le RAC et le CLC.

Publication d'un document

- L'utilisateur crée du contenu avec une application compatible RMS et précise les droits, les destinataires et les conditions de publication.
- Le client RMS génère une clé de session AES 128 bits, chiffre le contenu et crée une PL (licence de publication) chiffrée avec la clé publique du serveur (à travers la CLC). L'en-tête du document (Word par exemple) contiendra cette PL. Si on retire manuellement cet en-tête, le reste du document est chiffré donc illisible. Cette PL contient (en clair) l'URL du serveur RMS afin d'obtenir des droits d'accès. (Voir figure 4, page suivante.)

Révocation RMS

La révocation est un mécanisme qui révoque des informations d'identification qui ont déjà été émises, comme un certificat ou une licence. L'objectif principal de la révocation est d'empêcher les



4 *Exemple d'autorisation RMS*

entités qui ne sont plus approuvées d'intervenir dans un système RMS. Ainsi, la révocation peut être appliquée dans les scénarios suivants :

⇨ pour empêcher l'utilisation d'un contenu lorsqu'une entité figurant dans la chaîne d'approbation a été compromise, par exemple lorsqu'un employé quitte votre entreprise et qu'il ne doit plus pouvoir consulter le contenu protégé par RMS ;

⇨ pour empêcher une application compatible RMS donnée d'ouvrir un élément de contenu si l'application n'est plus approuvée ;

⇨ pour empêcher qu'un élément de contenu suspect déjà distribué et doté d'une licence autorisant son utilisation ne soit encore utilisé.

Lorsque vous l'activez, la révocation est effective chaque fois qu'un utilisateur tente d'utiliser un contenu protégé, que cet utilisateur dispose ou non d'un exemplaire stocké en local de la licence d'utilisation ou qu'il demande une nouvelle licence d'utilisation au serveur RMS au moment de son utilisation. La révocation est mise en œuvre par le biais de listes de révocation qui sont distribuées aux ordinateurs clients. Ces listes sont des fichiers XrML signés qui spécifient le contenu, les applications, les utilisateurs ou d'autres entités qui ont été révoqués par l'entité émettant la liste.

En général, il est préférable d'octroyer une licence de documents à des groupes d'utilisateurs définis dans *Active Directory* plutôt qu'à des comptes d'utilisateurs individuels. Nous vous recommandons ce choix qui vous permettra de supprimer l'utilisateur du groupe Active Directory pour qu'il ne puisse plus lire les documents envoyés au groupe s'il venait à quitter la société. Cependant, l'utilisateur pourra toujours lire les documents bénéficiant de licences d'utilisation sauf si les droits des documents ont été définis de manière à exiger de l'utilisateur qu'il obtienne une licence d'utilisation à chaque ouverture du document. Si ce droit n'est pas défini, le seul moyen d'empêcher l'utilisateur d'ouvrir les documents protégés est d'effacer le magasin de licences de l'utilisateur sur son ordinateur.

Les utilisateurs internes auront besoin d'accéder aux serveurs RMS qui émettent les certificats de comptes RM (RAC) et les licences d'utilisation. Par défaut, le serveur RMS écoute sur HTTP (port 80 TCP) ou HTTPS (port 443 TCP), selon que votre serveur est configuré pour utiliser SSL, ce qui veut dire que ces ports doivent être ouverts sur le pare-feu de la partie Internet. On notera que les messages échangés sont complexes et difficilement « filtrables »...

3.3 Test, limites et recommandations

Si vous êtes pressé...

RMS est une solution qui nécessite une architecture Microsoft de bout en bout, y compris un annuaire Active Directory pour la gestion des droits.

C'est donc une solution qui ne fonctionne qu'à l'intérieur d'un domaine de confiance (clients, serveurs, utilisateurs) de type intranet (mais toutes les autorités sont cross-certifiées par Microsoft).

Il est nécessaire de faire activer son serveur de clés par Microsoft (en ligne on non).

L'utilisateur est identifié par son email renseigné dans Active Directory. Cet email doit être joignable.

Et, enfin, il ne faut accorder qu'une confiance toute relative à ce type de solutions : si quelqu'un peut « lire » un document à l'écran, il peut probablement (avec un minimum d'effort) récupérer le document en clair dans la mémoire...

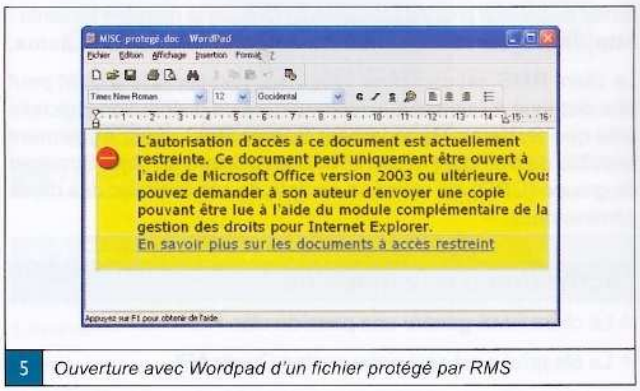
Cas particuliers

Des tests avec RMS 1.0 SP2 nous ont permis de vérifier l'application des droits sous RMS avec un résultat satisfaisant. En revanche, quelques scénarii restent intéressants :

⇨ Application de droits classiques (lecture, contrôle total, etc.) + droits RMS avec création de groupes d'utilisateurs sous AD : ce sont les droits les plus lâches qui s'appliquent.

⇨ Suppression d'un modèle de droits RMS appliqué par défaut : si le document avait déjà été ouvert, la licence avait été émise. Il n'y a pas de problème d'ouverture tant que la licence n'expire pas. Si le document n'avait jamais été ouvert, il est impossible d'ouvrir le document (sauf pour son propriétaire).

⇨ Restriction pour l'auteur du document : l'auteur peut toujours ouvrir son document, même s'il n'a pas les droits RMS de lecture, car il possède la licence d'utilisation.



5 *Ouverture avec Wordpad d'un fichier protégé par RMS*

Recommandations :

⇨ Le mot de passe du compte administrateur RMS doit être robuste, car c'est le maillon faible.

⇨ Activer le protocole SSL afin de chiffrer les communications avec le service web.

⇨ IIS ne doit héberger que ce service web (pas de mutualisation de sites web).

⇨ Sécuriser les serveurs : AD, web, bases SQL Server, notamment avec l'activation des audits journalisés (système et applications) et la mise en place d'un IDS.

⇨ Les certificats sont basés sur l'adresse email sous Active Directory : ne pas utiliser les adresses fonctionnelles (alias ou liste de diffusion), mais les adresses personnelles.

Retour en arrière

Le processus de mise hors service d'une architecture RMS est possible. Chaque application compatible RMS de l'utilisateur doit ensuite être connectée au service de mise hors service lorsqu'une fonction RMS est utilisée. Une clé de registre spécifique identifie le service de mise hors service. Une fois que cette clé est configurée, le serveur RMS octroie des licences d'utilisation qui accordent des autorisations complètes (lecture, écriture, copie, impression, modification, etc.) à l'utilisateur pour ce contenu, que ces autorisations lui aient été accordées ou non au début. Les utilisateurs doivent ensuite être dirigés pour supprimer toutes les protections de droits pour tous les documents qu'ils veulent conserver une fois le serveur RMS entièrement mis hors service. Une fois cela fait, le serveur RMS peut être mis totalement hors service et les fichiers de programme du RMS client peuvent être supprimés.

Nous vous recommandons de sauvegarder la base de données de configuration du serveur RMS, au cas où vous auriez besoin de récupérer un document protégé par RMS une fois le serveur supprimé. Sans la clé privée du serveur RMS, seul l'auteur du document pourra ouvrir le contenu protégé par RMS une fois le serveur supprimé. Vous pouvez supprimer le Lockbox de `%systemroot%\system32`.

Recouvrement

RMS prend en charge un groupe spécial de super utilisateurs qui détient un contrôle total sur tout le contenu protégé par RMS. Les membres du groupe des super utilisateurs bénéficient de droits de propriété complets pour toutes les licences d'utilisation émises par le serveur ou *cluster* RMS sur lequel le groupe des super utilisateurs est configuré. Par conséquent, les membres de ce groupe peuvent déchiffrer tous les fichiers protégés et supprimer leur protection !

Si les droits RMS sont définis de manière à rejeter la fonctionnalité de copie, [Alt]+[IMP écran] est désactivé par RMS. Un logiciel d'enregistrement des manipulations à l'écran (pour générer une vidéo) peut être utilisé et, à l'extrême, un appareil photo numérique pour capturer le contenu d'un fichier Word protégé. Cependant, dans un environnement avec des postes non gérés, un utilisateur pourrait utiliser des produits autres que Microsoft pour saisir le contenu.

Avec RMS, les administrateurs effectuent la sauvegarde, mais ne peuvent pas accéder au contenu.

Lors de l'ouverture d'un fichier, les fichiers de sauvegarde automatique et les fichiers temporaires gérés par Office 2003/2007 sont chiffrés. Les antivirus sur les serveurs de messagerie sont impuissants si le fichier (chiffré) contient un virus. Une restauration de fichier protégé par RMS est donc une opération très délicate pour l'administrateur système, car le logiciel de sauvegarde n'a pas vérifié l'intégrité des données.

La conversion de format du document est également une faille : convertir un document Word en un autre format (impression dans un fichier PDF par exemple) est une menace pour vos en-têtes (certificats).

L'écoute réseau est neutralisée par le chiffrement entre le serveur et le client RMS, mais certains cas ne peuvent pas être évités, comme le *sniff* des paquets entre l'ordinateur et l'imprimante réseau.

La menace réside, coté serveur, sur la sécurité des services Web IIS et SQL. La mise à jour des patches de sécurité Microsoft sur ces serveurs ne doit pas être oubliée...

Conclusion

RMS sous Windows Server 2003 sécurise les données sensibles d'un système d'information à partir de clés et de certificats associés aux documents. Les entreprises et les développeurs ont à leur disposition une nouvelle technologie de chiffrement et de licence pour protéger leurs informations sensibles d'une utilisation non autorisée. Cette protection devient critique pour les communications externes, notamment B2B (interentreprises). RMS est une des options natives de la suite MS-Office 2007 et de Windows Vista. Il permet aux professionnels de distribuer des informations tout en définissant comment et sous quelles conditions elles pourront être utilisées. Par exemple, il est possible de définir une date d'expiration et l'identité de l'utilisateur qui peut ouvrir, modifier, imprimer ou transmettre un document par Outlook. Cette possibilité renforce la sécurité des informations partagées en interne dans les entreprises, mais ne concerne pour l'instant que les documents bureautiques MS-Office élaborés sous le système d'exploitation Windows. C'est pourquoi cette solution ne répond pas globalement à la problématique de fuite d'informations. Il faut lui associer une politique de filtrage des flux réseau pour toutes les applications et les formats de documents manipulés dans l'entreprise. Nous avons évoqué le risque de mise hors service accidentelle d'une architecture RMS, de déni de service du serveur IIS et le risque lié aux pouvoirs de l'administrateur RMS capable à tout moment de déchiffrer, à l'insu de son propriétaire, des documents chiffrés. Le SDK de RMS ouvre les portes à la réalisation d'applications internes répondant aux critères RMS. Pour la prochaine version de Windows Server (Longhorn), RMS V2 devient le socle d'authentification et d'autorisation distribuées pour *Active Directory Federation Service*.

Références

[1 et 1a] BAREL (Marie), « Confidentialité et cryptographie en entreprise », Actes SSTIC 2007, page 221, avec notamment l'arrêt Nikon et l'arrêt du 18/10/2006 pourvoi nr04-48.025.

[2] Solution Mc Afee : **http://www.mcafee.com/fr/enterprise/products/data_loss_prevention/data_loss_prevention.html**

[3] VERICEPT : **http://www.vericept.com/solutions/products.asp**

[4] *Content Protection Suite* : **http://www.websense.com/global/fr/**

[5] Vontu 7 : **http://www.vontu.com/products/**

[6] *Digital Guardian* : **http://www.verdasys.com/digital_guardian.php**

[7] RMS, technet et webcast RMS : **http://www.microsoft.com/**

[8] XRML : **http://www.xrml.org/**

[9] Livre blanc RMS Microsoft : **http://www.microsoft.com/france/office/editions/prodinfo/livreBlanc/IRM.mspx**

Microsoft Network Access Protection

Nouveauté majeure du couple Vista/Longhorn, la technologie NAP (Network Access Protection) permet la mise en œuvre de contrôles de conformité à de multiples niveaux.

mots clés : *contrôle d'accès / contrôle de conformité / 802.1x / IPSec / DHCP*

Le contrôle de conformité lors de l'accès au réseau est une technologie actuellement sous les feux de l'actualité. De nombreux acteurs proposent des solutions, de nombreux chercheurs en sécurité les étudient et identifient les moyens de les contourner. Le sujet a déjà été traité dans son ensemble dans un article précédent [1] et nous rappellerons uniquement ici les grands principes nécessaires à la compréhension. Cet article a pour objectif de présenter comment fonctionne la solution proposée par Microsoft et quelles en sont les limites. L'ensemble des informations ci-dessous est basé sur la version finale de Vista et la version bêta 3 de Longhorn. Il est important de préciser que des modifications pourront avoir lieu, aussi bien sur le fonctionnement que sur les plateformes supportées avant la sortie officielle de Windows Server 2008.

Principaux éléments du contrôle de conformité

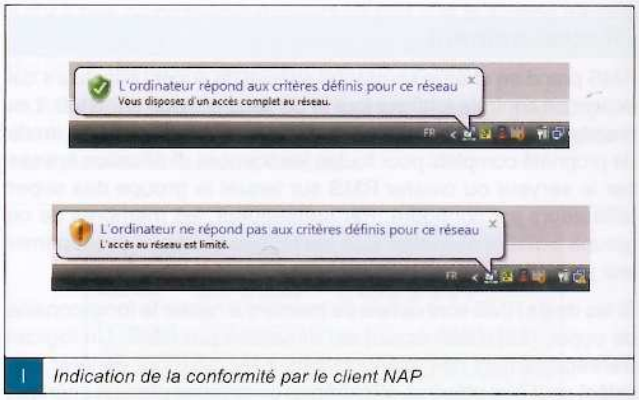
Microsoft envisage le contrôle de conformité pour protéger le réseau de la connexion de postes non conformes. Le but premier de NAP n'est pas de réaliser l'authentification des équipements se connectant, mais de vérifier leur conformité par rapport à un référentiel de sécurité. NAP permet trois usages différents : le « *monitoring* » permettant de suivre la conformité et d'enregistrer l'état des postes, l'« *enforcement* » permettant d'isoler les postes non conformes (lors de la connexion et dans le temps) et la « *remédiation* » qui peut aller jusqu'à remettre à niveau automatiquement les postes non conformes. NAP est une évolution majeure du système NAQ « *Network Access Quarantine* » historiquement présent dans le service d'accès distant sous forme de script [2].

Comme toute solution de contrôle de conformité, NAP repose sur 3 composants clés :

⇨ **Un client « NAP client »** : présent sur le poste de travail, il va collecter les informations de sécurité du socle Windows et des autres éléments de sécurité (soit par l'intermédiaire du centre de sécurité, soit par des modules additionnels). Il s'assure également du dialogue avec les points de contrôle sur le réseau. Aujourd'hui, le client NAP est inclus par défaut dans Vista et sera disponible pour Windows XP SP2. Le client Vista dispose cependant de fonctionnalités plus avancées : console locale, support de Windows Defender, SSO en 802.1x, etc.

⇨ **Un point de contrôle « NAP Enforcement Server »** : c'est le point où les échanges réseau vont être interceptés et bloqués afin de vérifier la conformité du poste. Avec NAP, ce point de contrôle peut être mis en œuvre à de nombreux niveaux sur le réseau (DHCP, 802.1x, VPN) et même au plus près des ressources (serveurs, postes de travail...) par l'usage de filtres IPSec. Afin, entre autres, d'étendre le champ des points de contrôle compatibles NAP, des accords ont été conclus avec différents constructeurs pour utiliser certains de leurs équipements, en particulier ceux réalisant des contrôles de niveau IP [3 et 4].

⇨ **Un référentiel de sécurité « Network Policy Server / NAP Health Policy Server »** : c'est lui qui prend les décisions en comparant les informations émises par les postes de travail et le référentiel de sécurité. Ce référentiel peut être soit stocké localement sur le NPS, soit faire appel à des serveurs tiers. NPS remplace le serveur IAS (*Internet Authentication Service*) de Windows 2003, son champ d'action est donc plus étendu que le simple contrôle d'accès au réseau. Sur le serveur NPS, c'est le composant *NAP Health Policy Server* qui réalise les actions propres à NAP. Dans la suite de l'article, nous utiliserons le terme « NPS/NAP » pour identifier ce composant.



1 *Indication de la conformité par le client NAP*

Protocoles et architectures utilisés par NAP

Microsoft a conçu de nouveaux protocoles pour mettre en œuvre NAP et a profondément revu l'infrastructure EAP de Windows en créant le composant EAPHost. Ces modifications étaient devenues nécessaires. Elles ont été une des causes des difficultés de déploiement du 802.1x avec Windows XP SP2. Une avancée importante est l'initialisation du client NAP au plus tôt dans le cycle de démarrage. Ceci permet la réalisation des contrôles même en l'absence d'utilisateur logué et évite les cas de blocages mutuels (j'ai besoin d'être identifié pour vérifier la conformité, mais j'ai besoin d'être conforme pour m'identifier).

Gérôme Billois
Responsable du département Sécurité des Systèmes d'Information, Solucom
gerome.billois@solucom.fr
http://www.solucom.fr

Principe des SHA/SHV et protocole d'échanges des SOH

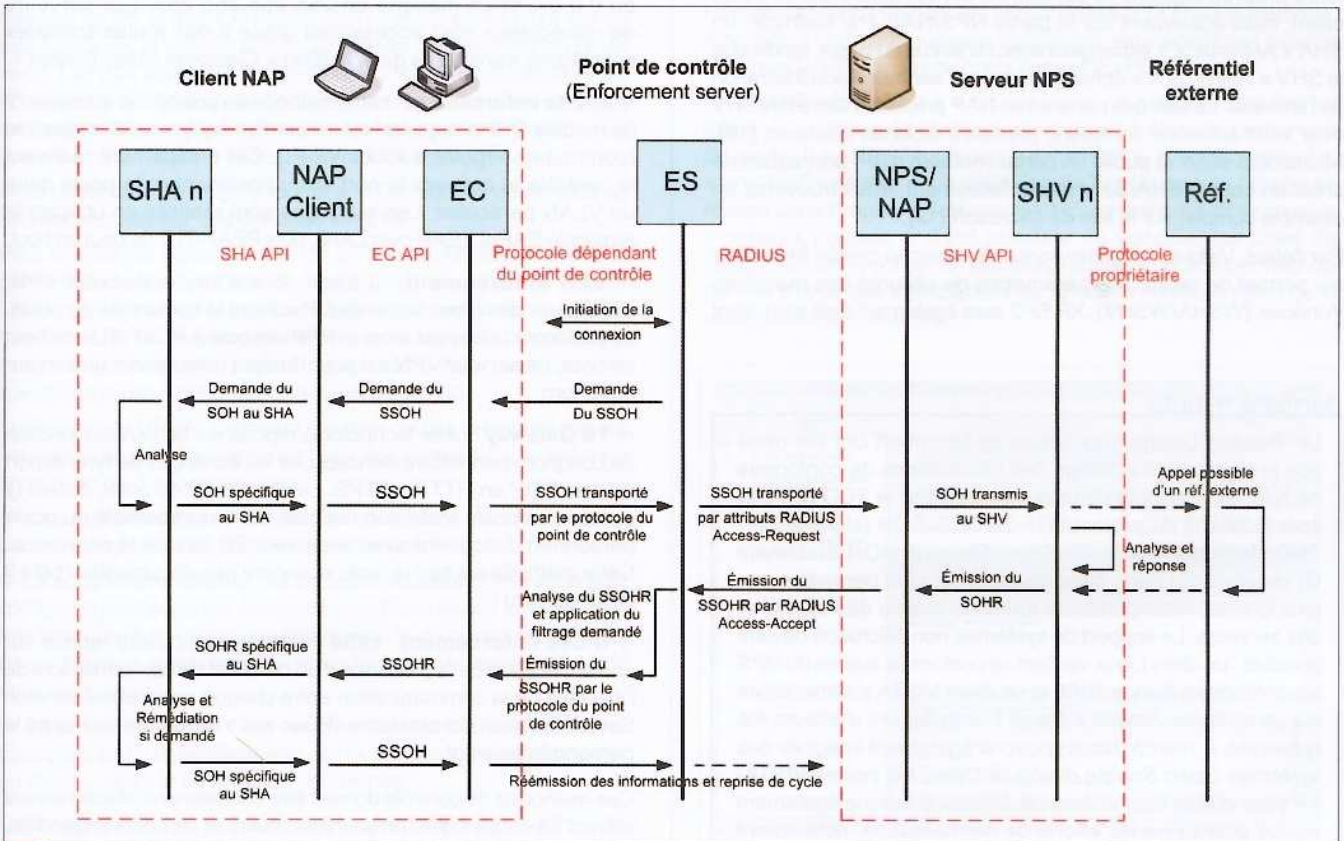
NAP, en tant que tel, définit la couche protocolaire permettant l'échange des informations de conformité et de remédiation et fournit un certain nombre d'éléments d'infrastructure propres à l'environnement Windows (point de contrôle, module de conformité).

Le client NAP est composé de plusieurs éléments :

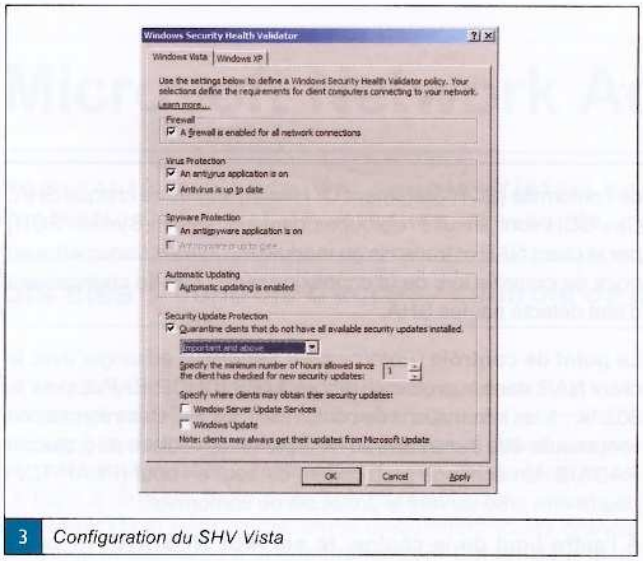
- le client NAP en tant que tel, qui centralise les messages et coordonne les échanges ;
- les modules SHA (*System Health Agent*) en charge de collecter les informations de conformité sur le système, de les transmettre au client NAP, mais également de mettre en œuvre les demandes de remédiation ;
- les modules d'enforcement (EC), qui vont réaliser les échanges protocolaires en fonction du point de contrôle rencontré (par exemple client 802.1x pour échanger avec un commutateur 802.1x, etc.). En situation de vérification, le client NAP va collecter les informations de conformité (SOH *Statement Of Health*) auprès de chaque SHA. Ces SOH sont ensuite regroupés dans un SSOH (System SOH) par le client NAP et transmis au module EC, qui les transmettra au point de contrôle lors de la connexion ou en cas de changement d'état détecté par les SHA.

Le point de contrôle (*Enforcement Server*) va échanger avec le client NAP, dans le protocole propre à l'ES (DHCP, EAPoL pour le 802.1x...), les informations de conformité (SSOH). Ces informations vont ensuite être transmises au NPS par l'intermédiaire du protocole RADIUS. Un échange protocolaire de bout en bout (PEAP-TLV) pourra être créé suivant le protocole de conformité.

À l'autre bout de la chaîne, **le serveur NPS/NAP** reçoit le SSOH et communique chaque SOH aux différents modules de validation (SHV *System Health Validator*). Les modules SHV sont les pendants des modules SHA sur le poste. Ce sont eux qui vont analyser les différents SOH provenant des SHA client et évaluer leur niveau de conformité. Chaque SHV fera état du niveau de conformité par l'intermédiaire d'un message SOHR (*Statement Of Health Response*). Le serveur NPS définira alors l'état de conformité global, les actions d'isolement, d'acceptation,



2 *Échanges protocolaires*



3 *Configuration du SHV Vista*

de remédiation à réaliser. Ces informations seront envoyées sous la forme d'un message SSOHR au point de contrôle qui l'analysera et appliquera les règles de conformité, mais également au client NAP qui le communiquera aux différents SHA pour réaliser les actions de remédiation.

Il n'y a pas de limitation au niveau des SHA et SHV. Ces modules peuvent être développés par des tiers pour prendre en compte leurs propres logiciels et serveurs d'infrastructure sur le poste client, mais également sur la partie NPS/NAP. Par exemple, un SHA « Antivirus X » échangera avec l'antivirus du poste, tandis que le SHV « Antivirus X » échangera avec le serveur d'administration de l'antivirus. La liste des partenaires NAP préparant des SHA/SHV pour leurs solutions comprend plusieurs dizaines d'acteurs [10]. Microsoft a défini et publié un certain nombre d'API permettant de créer un couple SHA/SHV assez facilement. Vous trouverez un exemple complet sur le site de Microsoft [12].

Par défaut, Vista et Longhorn sont livrés avec un couple SHA/SHV qui permet de vérifier les paramètres de sécurité des machines Windows (WSHA/WSHV). XP SP2 sera également doté d'un client NAP et d'un SHA spécifique. Les paramètres suivants peuvent être contrôlés :

- activation du pare-feu ;
- activation et mise à jour de l'antivirus ;
- activation des mises à jour automatique ;
- correctifs de sécurité à jour et date de dernière vérification ;
- pour Vista, présence et activation de Windows Defender.

Ces mesures restent très simples et manquent un peu de souplesse. Cependant, elles couvrent les cas principaux de protection d'un poste de travail.

dernière minute

Le *Trusted Computing Group* et Microsoft ont annoncé que le protocole d'échange des informations de conformité de NAP sera adopté comme standard par le TCG [5]. Les spécifications du protocole IF-TNCCS-SOH (signifiant IF-TNC : Interface TNC ; CS Client-Serveur; SOH Statement Of Health) sont donc disponibles [6] et vont permettre une plus grande interopérabilité aussi au niveau des clients et des serveurs. Le support de systèmes non-Microsoft devient possible : un client Linux vérifiant sa conformité auprès du NPS sous Windows Server 2008 et un client VISTA s'authentifiant sur un système Juniper Infranet Controller ont d'ailleurs été présentés à Interop. Nous pouvons également imaginer des systèmes *Open Source* dialoguant avec les postes VISTA/XP pour vérifier leur conformité. Microsoft indique également vouloir poursuivre les efforts de normalisation, notamment auprès de l'IETF [7].

Zoom sur les points de contrôle

Les méthodes de contrôle possibles dans NAP sont multiples. Il est possible d'utiliser une ou plusieurs méthodes simultanément sur un seul et même réseau. Le point de contrôle (ES Enforcement Server) communique dans tous les cas avec le serveur de conformité (NPS/NAP) en utilisant le protocole RADIUS et en utilisant des « *Vendor-Specific Attributes* » (VSA) dans des messages de type « Access-Request » et « Access-Accept ». Les différents points de contrôle possibles avec NAP en standard sont les suivants :

- **DHCP enforcement** : un serveur DHCP vérifiera la conformité avant de délivrer une adresse IP. Le client échange avec le point de contrôle en utilisant des attributs DHCP Vendor-Specific dans les messages DHCPDiscover, Request et Offer. En cas de non-conformité, le client sera isolé en indiquant un routeur par défaut en 0.0.0.0 et un masque en 255.255.255.255. Les serveurs de remédiation sont accessibles grâce à des routes statiques spécifiques transmises dans l'option « Classless Static Routes ».

- **802.1x enforcement** : cette méthode se positionne au niveau 2 du modèle OSI et requiert l'utilisation d'un équipement compatible (commutateur/point d'accès Wi-Fi). Cet équipement réalisera le contrôle et activera le port et/ou positionnera le poste dans un VLAN particulier. Les échanges sont réalisés en utilisant le protocole EAPoL (EAP over LAN), puis PEAP-TLV de bout en bout.

- **VPN enforcement** : il s'agit, lors d'une connexion VPN, d'appliquer des filtres au niveau IP suivant la conformité du poste. Le protocole utilisé est alors le PPP associé à PEAP-TLV de bout en bout. Le serveur VPN est pour l'instant uniquement un serveur Longhorn.

- **TS Gateway** : cette technologie repose sur la nouvelle fonction de Longhorn permettant d'encapsuler les échanges de type déport d'écran RDP en HTTP/HTTPS. Le client NAP du poste distant (y compris un poste Vista non maîtrisé de type cybercafé ou poste personnel) dialoguera avec le serveur TS lors de la connexion. Cette méthode est très récente et encore peu documentée (bêta 3 de Longhorn) ;

- **IPSec enforcement** : cette méthode particulière repose sur une vérification de la présence d'un certificat de conformité lors de l'initiation de la communication entre chaque poste et/ou serveur. Cette utilisation du protocole IPSec est traitée en détail dans le paragraphe suivant.

Ces méthodes de contrôle doivent être choisies avec discernement suivant les risques que l'on souhaite couvrir et les coûts engendrés, aussi bien en termes d'investissement que de d'administration. Elles sont en effet plus ou moins facilement contournables ou usurpables.

Par exemple, dans un bâtiment où le contrôle d'accès physique est bien maîtrisé, le déploiement de la solution de contrôle sur DHCP peut suffire à traiter 80% des incidents courants, mais pas les intrusions. Sur des périmètres très exposés (point de vente, salle de réunion...), l'usage du 802.1x peut apporter l'authentification des machines, et donc un meilleur contrôle des entrées sur le réseau.

Le cas particulier de l'IPSec

Une mise en œuvre de NAP est possible par l'intermédiaire de l'IPSec. Le contrôle de conformité ne sera alors plus réalisé au niveau d'un point de contrôle, mais par chaque poste de travail recevant une demande d'échange réseau. L'IPSec n'est pas alors utilisé en mode tunnel (cas habituel de chiffrement des échanges), mais en mode transport afin de réaliser une authentification basée sur les certificats de la machine.

Il s'agit pour le client NAP d'obtenir un certificat temporaire prouvant la conformité du poste et lui permettant ensuite de communiquer avec d'autres postes/serveurs.

Le fonctionnement de base est le suivant :

⇨ Un poste client est présent sur le réseau, il dispose d'une adresse IP. Ce poste va tenter d'échanger avec un serveur.

⇨ Le serveur, avant d'initier l'échange d'information, va demander au poste client de fournir son certificat de conformité par les mécanismes IPSec.

⇨ Le poste client va envoyer ses informations de conformité (SSOH) au serveur HRA (Health registration Authority). Les échanges ont lieu en utilisant le protocole HTTPS. Le serveur HRA dispose de IIS et des *Certificate Services*. C'est le HRA qui va dialoguer avec le serveur NPS pour évaluer la conformité du poste client. Le HRA joue alors le rôle de proxy RADIUS et transfère les échanges à destination du NPS. En cas de succès, un certificat de conformité sera alors généré et envoyé au client. Dans le cas contraire, une remise à niveau pourra être envisagée.

⇨ Le poste client peut alors fournir son certificat de conformité au serveur et échanger des données avec lui.

Pour faciliter les échanges décrits ci-dessus, Microsoft a revu la partie IKE de la pile IPSec de Vista/Longhorn. Nommé AuthIP (*Authenticated Internet Procotol*), cette nouvelle méthode d'échange de clé autorise de nombreux modes d'authentification (utilisateur, machine, asymétrique, etc.). Elle reste encore spécifique à l'environnement Vista/Longhorn **[8]**.

L'utilisation de l'IPSec permet la création de ces réseaux logiques sans toucher à l'infrastructure et aux équipements. Elle entraîne la création de 3 zones logiques :

⇨ **La zone sécurisée** : elle est composée de tous les postes/serveurs disposant de certificats de conformité.

⇨ **La zone intermédiaire** : les serveurs/postes de cette zone sont configurés pour accepter des communications issues aussi bien de machines conformes que non conformes. Ils disposent du certificat de conformité. Le serveur HRA et les serveurs de remédiation sont dans cette zone.

⇨ **La zone restreinte** : elle recense l'ensemble des postes ne disposant pas du certificat soit car ils viennent de se connecter au réseau, soit car ils sont non conformes.

Cette méthode est la plus souple proposée par NAP : elle ne nécessite pas la mise en place de points de contrôle, ni de modifications sur l'infrastructure réseau. De plus, elle permet de continuer à utiliser des services moins critiques sans créer de règles de filtrage complexes pour les isoler. La méthode de contrôle IPSec est également difficile à contourner : vérification à chaque nouvelle connexion IP, émission des certificats de conformité par le HRA, etc. Des informations détaillées ont été publiées par Microsoft **[9]** Attention cependant, cette méthode ne va pas garantir la disponibilité du réseau comme le ferait le 802.1x par une protection au plus prêt du poste de travail. Les problèmes rencontrés dans le cas des échanges IPSec en Host to Host sont encore d'actualités (adressage réseau, support des protocoles IPSec...) **[11]**. De plus, la configuration reste complexe et le système parfois difficile à dépanner.

Mise en œuvre de NAP

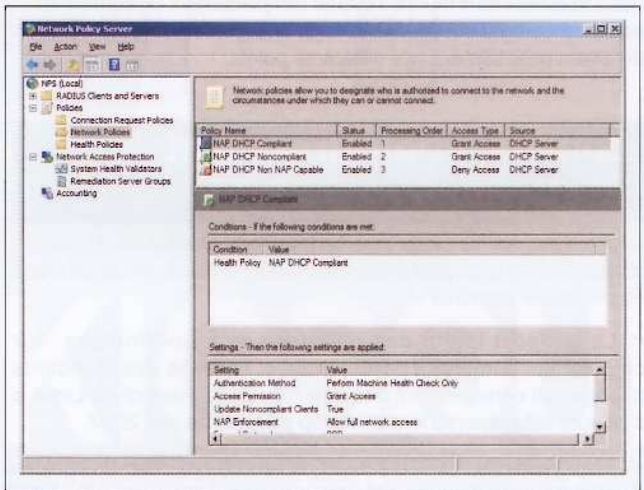
L'objectif de cet article n'est pas d'écrire un guide détaillé de mise en œuvre, nous n'en n'aurions malheureusement pas la place. De plus, Microsoft a mis à disposition des guides de mise en œuvre détaillés **[10]**. Même si ces guides évoluent rapidement du fait des versions bêta, ils représentent une aide indéniable. Nous détaillons ici les étapes clés de la configuration.

Au niveau serveur NPS

Il est nécessaire de configurer les quatre modules différents composant le NPS :

⇨ **« Connexion request »** : il s'agit ici de configurer, pour chaque méthode d'enforcement (DHCP, IPSec, TS, etc.), le comportement que doit avoir le NPS quand il reçoit une demande de connexion (RADIUS ou Proxy RADIUS). La méthode de configuration et les critères d'acceptation peuvent être spécifiés pour chaque méthode. Attention : une configuration spécifique est nécessaire pour le 802.1x et le VPN, qui requièrent l'usage du PEAP ;

⇨ **« Network policies »** : c'est dans ce module que l'on va définir la politique de réaction par rapport à l'état de conformité du poste. Il existe 3 cas type : « compliant », « non compliant », « non NAP capable ». La définition de la politique de filtrage se réalise dans l'onglet « Settings/NAP Enforcement » pour chaque « network policies ». Il est possible de spécifier les listes de filtrage, les serveurs de remédiation, l'adresse de la page Web qui permet d'avoir des informations de remise à niveau en plus de la remédiation automatique ;



4 *Configuration du NPS*

⇨ **« Health policies »** : dans ce module, nous définissons les critères de conformité requis pour chaque politique en précisant quel SHV ou couple de SHV doivent être validés pour être considérés comme conformes (par exemple : SHV Antivirus et SHV Pare-feu et SHV Windows) ;

⇨ **« System health validators »** : ce module permet de configurer les paramètres propres à chaque SHV et le comportement en cas de problème dans la chaîne de liaison (perte de communication, indisponibilité des éléments…).

Au niveau du client

Le client repose sur le service « *Network Access Protection Agent* ». Une console MMC (uniquement sous Vista) ou des GPO permettent de configurer les différents paramètres sur le client :

⇨ activation ou désactivation des NAP-EC ;

⇨ personnalisation des messages d'alerte (image, titre de fenêtre, texte de fenêtre, etc.) ;

⇨ paramètres spécifiques pour l'IPSec (Hash, CSP…).

Des bulles d'informations indiquent également l'état actuel, et peuvent fournir des informations de remise à niveau à l'utilisateur. L'outil en ligne de commande **napstat** permet de forcer l'affichage de l'état. Pour suivre les différentes opérations, il est également possible d'utiliser deux outils classiques dans l'univers Microsoft :
⇨ **Netsh nap** : l'outil netsh est doté de nombreuses commandes liées au client NAP. Plus particulièrement, la commande `netsh nap client show state` affiche l'intégralité des informations concernant le client (EC enregistré, état de conformité…).

⇨ **Journal des évènements** et en particulier la branche « *Applications and Services Logs\Microsoft\Windows\Network Access Protection* ».

Pour un suivi encore plus précis des échanges, et en particulier des échanges EAP, il est possible d'activer une trace dédiée avec l'outil Logman. [13]

Au niveau des points de contrôle

La configuration est très dépendante de la méthode de contrôle sélectionnée (IPSec, DHCP, 802.1x…) et du type d'équipement. Dans tous les cas, il faudra indiquer l'emplacement du serveur NPS agissant comme serveur RADIUS. Les guides de configuration détaillent précisément ces différents points.

Limites, contournement et attaques

NAP est une solution fortement orientée poste de travail qui souffre aujourd'hui de quelques limitations fonctionnelles. Nous pouvons en particulier identifier :

⇨ **Le support limité des plateformes non-Windows**. Sur ce point les nombreux partenariats et l'arrivée des standards apporteront certainement une solution. Le premier client Linux a d'ailleurs été présenté lors d'Interop au mois de mai 2007.

⇨ **L'obligation de disposer d'un client sur le poste**. Contrairement à d'autres solutions présentes sur le marché, NAP ne dispose pas d'un mode « *clientless* » permettant soit l'utilisation d'un client léger téléchargé dynamiquement à la connexion, soit la réalisation d'une analyse réseau (scanner de vulnérabilité) lors de l'arrivée d'un poste.

⇨ **La gestion des exceptions**. Un des problèmes récurrents du contrôle d'accès au réseau est la gestion des équipements non munis de client de conformité (imprimantes, téléphones ToIP…). Il est dans ce cas impossible de dialoguer avec le poste et de définir son niveau de conformité. Plusieurs autres solutions existent pour appliquer un contrôle minimum (liste d'adresses MAC, exception par adresse IP, etc. Vous trouverez plus de détails dans l'article [1]). Au niveau de NAP, peu de solutions de contournement sont disponibles. Sur le serveur NPS, il est possible de définir des politiques à appliquer si l'équipement est « NAP non capable ». Un déploiement à grande échelle de points de contrôle contraignants devra être murement réfléchit, en particulier pour prévoir les solutions de gestion des exceptions pratiques et difficilement utilisables pour contourner le contrôle.

⇨ **L'absence de point de contrôle de niveau 3 en standard** : NAP ne propose pas de point de contrôle de niveau 3, de type routeur ou pare-feu (hors VPN). Cette limitation fonctionnelle peut rendre certains scénarios d'implémentation difficiles. Les accords récents avec Cisco et Juniper, mais aussi la standardisation des protocoles, devraient remédier à cette limite.

Lors de l'évaluation sécurité de NAP, il est important de préciser que ce système a été conçu dans l'esprit de vérifier la conformité des équipements se connectant par rapport à un référentiel de sécurité, et pas de réaliser l'authentification des postes sur le réseau. Nous pouvons imaginer des scénarios d'attaques au niveau de l'infrastructure comme :

⇨ **Le contournement du point de contrôle** : certaines méthodes de contrôle peuvent être rapidement contournées. En particulier, la vérification de conformité réalisée par le serveur DHCP est contournable simplement par l'attribution d'une adresse IP fixe (même si des mécanismes existent dans les commutateurs pour forcer l'allocation par DHCP, ces protections restent peu déployées). Par défaut, NAP ne prévoit pas de phase d'authentification de la machine. Ceci peut être possible suivant les méthodes de contrôle (IPSec, VPN et 802.1x, et plus largement les méthodes reposant sur PEAP le permettent).

⇨ **L'attaque du point de contrôle** pour le saturer et pour entraîner soit un déni de service, soit un défaut de contrôle. Cette attaque est propre à tout système et la vulnérabilité de NAP sur ce point n'a pas encore été complètement évaluée.

⇨ **La capture des informations de conformité** lors de réalisation d'un contrôle afin de les rejouer ultérieurement, cette capture pouvant avoir lieu entre le client et le point de contrôle ou entre le point de contrôle et le serveur de conformité. NAP est cependant protégé de ce type d'attaque par l'utilisation de l'IPSec ou de PEAP en 802.1x ou VPN.

⇨ **La prise de contrôle du référentiel de sécurité** : il s'agit de l'attaque la plus désastreuse. Elle permet, d'une part, la connaissance du référentiel et de ce qui est attendu d'un poste. Donc, elle facilite l'intrusion. Pire encore, en cas de modification, il est possible de rendre conformes des PC qui ne le seraient pas (diminution des exigences) ou de réaliser un déni de service généralisé en augmentant les exigences au point que plus un seul poste de travail ne puisse être conforme. La protection et le maintien en condition de sécurité du NPS sont donc essentiels. Le NPS devient également un élément critique en termes de disponibilité.

Au niveau du client, des scénarios différents peuvent être envisagés, soit pour réaliser une intrusion, soit pour capturer des informations sur les postes :

⇨ **L'usurpation du client** et l'envoi au point de contrôle d'informations erronées, mais correspondant au niveau de sécurité attendu (attaque de type **« lying end points »**). Des chercheurs ont récemment démontré que ces attaques étaient réalisables, même si elles demandent encore un niveau d'expertise important [14].

⇨ **La fourniture d'informations erronées au client NAP officiel** : nous pouvons imaginer des modules SHA qui s'enregistrent en usurpant le SHA officiel et qui vont pouvoir émettre des informations usurpées.

⇨ Par rebond, il est également important de s'assurer de **la légitimité des points de contrôle demandant l'état de conformité**. En situation de mobilité, un point de contrôle NAP malveillant pourra engager le dialogue avec un client et connaître son état de conformité (en particulier en ce qui concerne les correctifs). Il sera alors facile de réaliser des attaques sur ce poste s'il n'est pas à jour (connaissance des failles sans même réaliser d'analyse ou de tests permettant un gain de temps important). La réalisation d'actions de remédiation malveillantes peut également être possible (déploiement d'un faux correctif, désactivation de composants de sécurité, etc.).

Plus globalement, les menaces posées par les *rootkits*, ou les machines virtuelles masquant au système les agissements exacts, restent d'actualité.

Administration

L'ensemble de l'administration repose sur les consoles habituelles chez Microsoft avec donc une centralisation des traitements. Attention cependant de prévoir de nombreuses interactions avec les équipes réseau en cas de déploiement, plus particulièrement pour le 802.1x et le cas des VPN.

La définition et le maintien à jour du seuil de conformité minimum reste également une étape sensible, nécessitant une bonne synchronisation entre les exigences de sécurité et la réalité des déploiements de correctifs et autres mécanismes de sécurité. Il s'agit d'éviter des situations de blocage où, par exemple, 90% du parc ne serait plus conforme. Des mécanismes de secours (autonomie d'un site, désactivation d'urgence...) doivent être envisagés dans les environnements critiques en termes de disponibilité.

Conclusion

Microsoft propose une fonction de sécurité innovante dans la prochaine génération des systèmes d'exploitation client et serveur. De plus, NAP est loin d'avoir été conçu comme un système minimaliste, son aspect modulaire (principe des SOH et des SHA/SHV) et la diversité des méthodes de contrôle (en particulier la méthode IPSec assez innovante) en font une solution rivalisant sans rougir avec les produits spécialisés. Même si NAP reste encore aujourd'hui en bêta et ne dispose que d'un support limité de points de contrôles et de solutions de gestion des exceptions, ce système donnera son plein potentiel avec la publication de nouveaux SHA/SHV, qui permettront un usage plus poussé que ceux possibles aujourd'hui. Cependant NAP reste perfectible sur plusieurs aspects, en particulier par le support majoritaire des technologies Microsoft. Les annonces récentes par rapport à la normalisation du TCG, ainsi que les accords de compatibilité, améliorent dès maintenant cette solution.

Références

[1] Contrôle d'accès réseau, MISC n°27.

[2] Microsoft Network Quarantine : **http://www.microsoft.com/technet/network/vpn/quarantine.mspx**

[3] Cisco et NAP : **http://www.microsoft.com/presspass/press/2006/sep06/09-06SecStandardNACNAPPR.mspx**

[4] Juniper et NAP : **http://www.microsoft.com/presspass/press/2007/may07/05-21MSJuniperPR.mspx**

[5] TCG et NAP : **http://www.microsoft.com/presspass/press/2007/may07/05-21NAPTNCPR.mspx**

[6] Spécifications du protocole IF-TNCCS-SOH : **https://www.trustedcomputinggroup.org/specs/TNC/IF-TNCCS-SOH_v1.0_r8.pdf**

[7] Groupe de travail de l'IETF sur le contrôle de conformité : **http://www.ietf.org/html.charters/nea-charter.html**

[8] AuthIP : **http://www.microsoft.com/technet/community/columns/cableguy/cg0806.mspx**

[9] NAP et IPSec dans le détail : **http://www.microsoft.com/technet/network/nap/napipsec.mspx**

[10] Site dédié à NAP : **http://www.microsoft.com/technet/network/nap/default.mspx**

[11] *Experiences with IPSec Host to Host* : **http://research.microsoft.com/users/mroe/spw2005.pdf**

[12] NAP API : **http://msdn2.microsoft.com/en-us/library/aa369706.aspx**

[13] Logman : **http://msdn2.microsoft.com/en-us/library/aa813696.aspx**

[14] NAC Attack : **http://www.ernw.de/content/e7/e181/e566/download568/ERNW_nacattack_10_dr_20070307_ger.pdf**

Open XML et MS Office 2007

Philippe Lagadec
NATO/NC3A
philippe.lagadec@nc3a.nato.int

Open XML est le nouveau format bureautique apporté par Microsoft dans la suite Office 2007, voué à remplacer à terme les formats classiques de Word, Excel et Powerpoint. Ce format est ouvert, et ses spécifications détaillées sont officiellement acceptées comme standard par l'ECMA. Comme son « concurrent » OpenDocument, il s'appuie sur des technologies bien connues comme XML et ZIP. Cet article décrit le format en détail, en se penchant particulièrement sur les caractéristiques concernant la sécurité. Il montre également quels problèmes peuvent se poser, notamment en termes de code malveillant ou de fuite d'informations.

mots clés : ***Microsoft / Open XML / Office 2007 / XML / ZIP / macros / OLE / antivirus / analyse de contenu / formats ouverts***

Un nouveau format bureautique ouvert

Open XML est le nouveau format par défaut pour les applications principales de la suite Microsoft Office 2007 : Word, Excel et PowerPoint. C'est un standard ECMA depuis fin 2006, et ses spécifications sont publiées en libre accès sur Internet **[OXSPEC]**. Une procédure de standardisation ISO est aussi en cours.

Par rapport aux formats bureautiques propriétaires, l'analyse de la sécurité de ce format est donc grandement facilitée par son caractère ouvert. Cependant, la complexité des spécifications (6045 pages !) et l'absence de documentation de certaines fonctionnalités ne rendent pas la chose si aisée.

On peut notamment remarquer que les points qui nous préoccupent particulièrement en matière de sécurité, comme les macros VBA, les objets OLE, le chiffrement ou les DRM ne sont pas abordés dans les spécifications publiées. Comme indiqué dans **[ECMA]** page 15, cette absence peut s'expliquer par le fait que Microsoft considère ces technologies comme propriétaires, optionnelles, et donc externes au format ouvert Open XML. Dans la pratique, il est cependant nécessaire de connaître ces détails pour pouvoir analyser ou convertir intégralement des documents produits par Office. On peut donc considérer que ce format n'est pas 100% ouvert. Ou alors que le standard ouvert Open XML n'est qu'un sous-ensemble du format Open XML, c'est selon.

D'autre part, la complexité du format Open XML peut s'expliquer par la volonté de Microsoft de supporter toutes les fonctionnalités des formats propriétaires antérieurs employés par sa suite bureautique.

Microsoft Office 2007 et Open XML

Les documents Open XML d'Office 2007 emploient de nouvelles extensions :

- Word : `.docx`, `.docm`, `.dotx`, `.dotm` ;
- Excel : `.xlsx`, `.xlsm`, `.xltx`, `.xltm`, `.xlsb`, `.xlam` ;
- PowerPoint : `.pptx`, `.pptm`, `.ppsx`, `.ppsm` ;
- Access : `.accdb` (nouveau format binaire, différent d'Open XML).

Office 2007 est toujours capable de lire et écrire les documents MS Office des versions précédentes (format binaire OLE2) grâce à un « mode de compatibilité ».

Il existe également un « Pack de compatibilité » pour lire et écrire des documents Open XML depuis les versions précédentes d'Office (2000, XP, 2003). La plupart des observations de cet article sont aussi valables pour une version antérieure d'Office munie de ce convertisseur.

Comme nous le verrons, certaines fonctionnalités d'Office ont été remaniées profondément, comme la sécurité des macros. Il semble que Microsoft ait prévu de faire profiter les anciennes versions d'Office de ces changements dans un avenir proche, sans doute sous forme de mises à jour.

Formats similaires

Open XML n'est ni le premier ni le dernier format bureautique ouvert constitué de fichiers XML dans une archive ZIP. Il existe au moins 3 autres formats ayant cette caractéristique :

- **OpenDocument** est le format utilisé entre autres par les suites OpenOffice.org et Sun StarOffice, normalisé par l'ISO en mai 2006. Cf. **[ODSPEC]**, **[MISC27]**. Pour une comparaison des aspects sécurité d'OpenDocument et Open XML, voir **[SSTIC07]** et **[PACSEC06]**.
- **XPS** est un autre nouveau format de Microsoft, proposant des caractéristiques similaires à PDF, pour la publication ou l'impression de documents finalisés. XPS possède la même structure interne qu'Open XML, appelée « *Open Packaging Conventions* » ou OPC. Cf. **[XPS]** et **[OXSPEC]** partie 2.
- **MARS** est un nouveau format d'Adobe en cours de définition pour prendre la suite de PDF (et bien sûr répondre au lancement d'XPS). Il reprend exactement les fonctionnalités actuelles de PDF, transcrites dans des fichiers XML et SVG à l'intérieur d'une archive ZIP. Cf. **[MARS]**.

Analyse de sécurité

Cette analyse de sécurité est basée sur les versions suivantes :

- spécifications Open XML ECMA-376, version finale publiée en décembre 2006 : **[OXSPEC]**.
- Microsoft Office 2007 version 12.0.4518.1014.
- Microsoft Office 2003 avec Open XML Converter Pack.
- Tous les tests ont été menés sur Windows XP SP2.

Les travaux ayant permis la rédaction de cet article ont été financés successivement par DGA/CELAR et par le programme de travail de recherche et développement de NATO/ACT.

Structure interne

Le format Open XML est principalement basé sur une archive ZIP contenant des fichiers XML. Cependant, la structure interne est un peu plus complexe. Open XML s'appuie en fait sur un nouveau format générique de conteneur proposé par Microsoft : OPC, pour « *Open Packaging Conventions* » (cf. **[OXSPEC]** partie 2). OPC sert de base à d'autres formats comme XPS **[XPS]**.

Voici les principaux fichiers XML à l'intérieur de l'archive ZIP, par exemple pour un document Word 2007 :

⇨ `[Content_Types].xml` : décrit chaque fichier dans l'archive ;

⇨ fichiers `.rels` : précisent les relations entre les objets des différents fichiers XML ;

⇨ `word/document.xml` : corps du document ;

⇨ `word/styles.xml` : styles ;

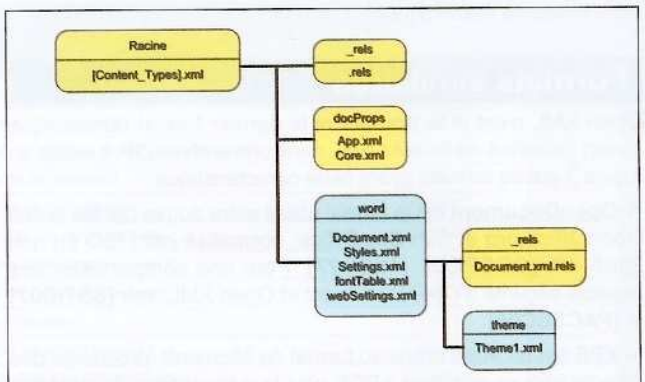
⇨ `word/settings.xml` : paramètres pour le document ;

⇨ `docProps/app.xml` et `core.xml` : méta-données (titre, auteur...).

On peut également y trouver d'autres types de fichiers :

⇨ **images** : JPEG, PNG, GIF, TIFF, WMF... ;

⇨ **fichiers binaires OLE2** : macros VBA, objets OLE, paramètres d'impression...



1 *Structure interne d'un document Word 2007 (.docx)*

Il n'y a pas de relation directe entre les différents fichiers XML. Les fichiers `.rels` établissent des relations indirectes entre les parties du document, ce qui apporte plus de flexibilité, mais rend l'analyse du document plus complexe.

Une analyse plus poussée du format montre en fait que les noms de répertoires et de fichiers dans l'archive ZIP ne servent qu'à organiser le stockage des données, et ne sont pas réellement importants pour l'application MS Office. L'organisation effective des documents repose principalement sur les relations décrites dans les fichiers `.rels`. Dans l'archive ZIP, on trouve donc 2 structures arborescentes qui cohabitent en parallèle, ce qui peut porter à confusion (au moins quand on analyse un document à la main).

Macros VBA

Il est possible d'inclure des macros en langage VBA (*Visual Basic for Applications*) dans les documents Open XML. Ces macros ont les mêmes fonctionnalités que pour les versions précédentes d'Office, et posent donc les mêmes problèmes de sécurité.

Une grande nouveauté est que les documents Open XML avec macros sont clairement distingués des documents sans macros. Leur extension est différente. Par exemple un document Word sans macro se terminera par `.docx`, et `.docm` s'il contient des macros. Si on renomme un document avec macros en `.docx`, Office refuse de l'ouvrir (en prétextant qu'il est « corrompu »).

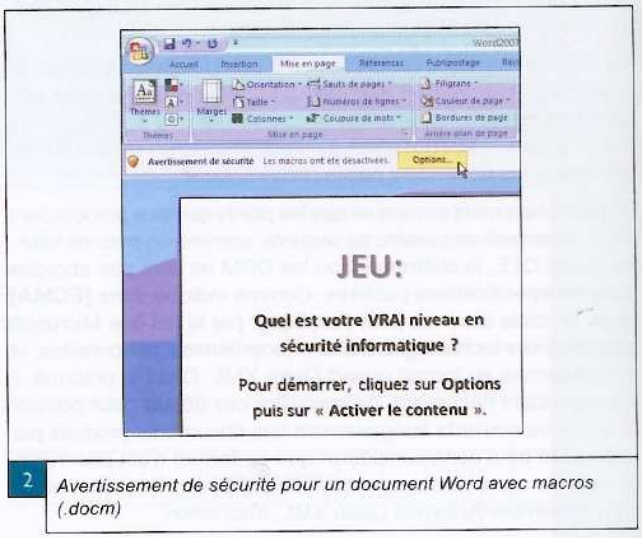
Ce changement permet donc à un utilisateur, un antivirus ou une passerelle de filtrage de reconnaître plus facilement quels documents contiennent des macros.

Niveaux de sécurité

Les niveaux de sécurité concernant les macros ont été également remaniés.

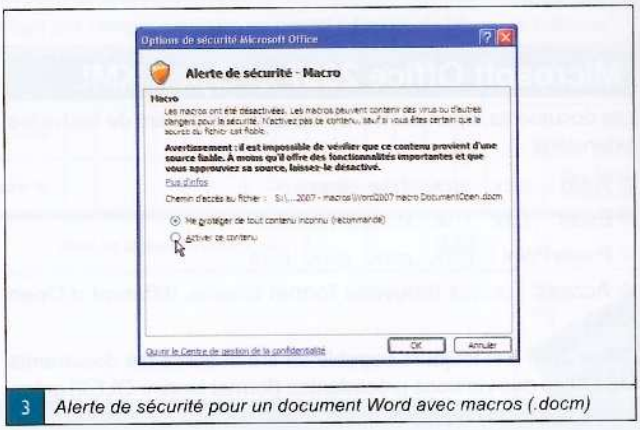
Dans les versions précédentes d'Office, le niveau par défaut était « haut », ce qui n'autorisait que l'exécution des macros signées ou provenant d'un emplacement de confiance. L'utilisateur devait modifier la configuration pour pouvoir exécuter des macros non signées.

Office 2007 fournit une nouvelle interface graphique (le « ruban »), et le nouveau niveau de sécurité par défaut est « Désactiver toutes les macros avec notification ». Lorsqu'un utilisateur ouvre un document avec une macro, un message s'affiche sous le ruban.



2 *Avertissement de sécurité pour un document Word avec macros (.docm)*

En cliquant sur ce message, il accède à une fenêtre qui lui permet d'autoriser l'exécution des macros, qu'elles soient signées ou non.



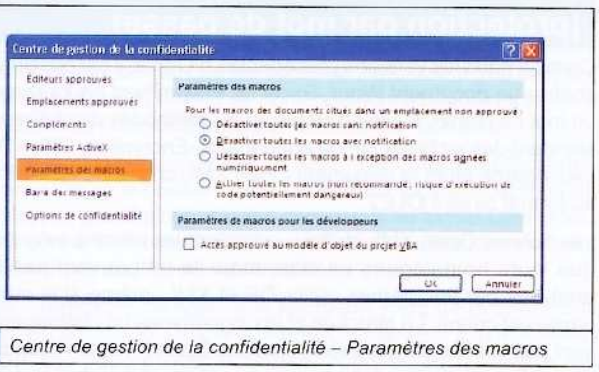
3 *Alerte de sécurité pour un document Word avec macros (.docm)*

Le nouveau mode par défaut permet donc à un utilisateur d'exécuter une macro non signée en 3 clics de souris, s'il ne lit pas consciencieusement tous les messages d'avertissement.

De plus, l'utilisateur peut lire le document avant de décider d'activer les macros, ce qui offre quelques possibilités de « *social engineering* ».

Tous les paramètres de sécurité sont maintenant regroupés dans une seule fenêtre appelée « Centre de gestion de la confidentialité » pour faciliter la configuration. On y trouve 4 niveaux de sécurité pour les macros :

- désactiver toutes les macros sans notification ;
- désactiver toutes les macros avec notification (par défaut) ;
- désactiver toutes les macros sauf les macros signées ;
- autoriser toutes les macros.



4 *Centre de gestion de la confidentialité – Paramètres des macros*

Cette fenêtre propose aussi des niveaux de sécurité pour les ActiveX.

Stockage des macros

Les macros sont stockées dans un fichier `vbaProject.bin`, dont l'emplacement à l'intérieur de l'archive varie selon les applications :

- Word : `word/vbaProject.bin` ;
- Excel : `xl/vbaProject.bin` ;
- Powerpoint : `ppt/vbaProject.bin`.

Ce fichier est au format binaire OLE2, et il n'est pas décrit dans les spécifications Open XML actuelles **[OXSPEC]**.

Si la macro a un nom spécifique, comme `Document_Open` pour Word, elle peut se déclencher automatiquement à l'ouverture du document. Dans ce cas, un fichier `vbaData.xml` est ajouté dans l'archive pour décrire ces événements.

Objets OLE

Comme dans les versions précédentes d'Office, il est possible d'inclure des objets OLE dans des documents Open XML, et on retrouve les mêmes problèmes de sécurité.

Les objets OLE permettent d'inclure dans un document des contenus provenant d'autres applications, par exemple un graphique Excel dans un document Word. Les objets OLE les plus problématiques sont les objets Package, qui peuvent contenir n'importe quel fichier, y compris un exécutable binaire, ou bien une simple ligne de commande (cf. **[SSTIC03]** pour des exemples concrets).

Les objets OLE peuvent être ouverts par un double-clic de l'utilisateur, directement sans confirmation et sans avertissement de la part d'Office. Sur un système Windows récent, c'est le gestionnaire de liaisons de Windows (`packager.exe`) qui demande confirmation à l'ouverture d'un objet OLE Package. Sur un système plus ancien (par exemple Windows 2000 SP4 non mis à jour), l'ouverture est directe sans confirmation (comportement similaire à WordPad). Heureusement, Office 2007 nécessite au minimum Windows XP SP2.

Office est du coup tributaire du système d'exploitation : par exemple en 2006, une vulnérabilité de Windows XP dans `packager.exe` permettait de camoufler la ligne de commande d'un objet OLE en faisant passer celui-ci pour un fichier inoffensif **[MS06-065]**.

Ces objets sont généralement stockés dans leur format d'origine, à des emplacements différents suivant l'application : par exemple `word/embeddings` pour Word. Un objet OLE Package est stocké au format OLE2.

Comme pour les macros, le stockage des objets OLE n'est pas décrit dans la version actuelle des spécifications Open XML.

Certains cas de figure peuvent produire des résultats étonnants : par exemple, il est possible d'inclure un classeur Excel avec macros (`.xlsm`) en tant qu'objet dans un document Word sans macro (`.docx`). Il n'y a pas d'avertissement à l'ouverture du document Word. Par contre, il y en a un à l'ouverture de l'objet Excel, qui propose directement d'activer ou non les macros. Et ceci, même si le niveau de sécurité est réglé sur « désactiver toutes les macros sans notification »...

Classeurs binaires Excel 2007

Excel 2007 permet de sauvegarder un classeur dans un format Open XML hybride, appelé « classeur binaire », avec une extension `.xlsb`. Ce format reprend certaines bases d'Open XML, mais remplace une partie des données XML par des fichiers binaires pour améliorer les performances, dans un format ressemblant à BIFF8, non documenté. Les classeurs binaires peuvent contenir des macros.

Fuite d'informations

Le format Open XML peut contenir diverses informations cachées, qui créent un risque de fuite d'informations sensibles, comme cela existe dans tous les formats bureautiques propriétaires **[OSSIR03]** :

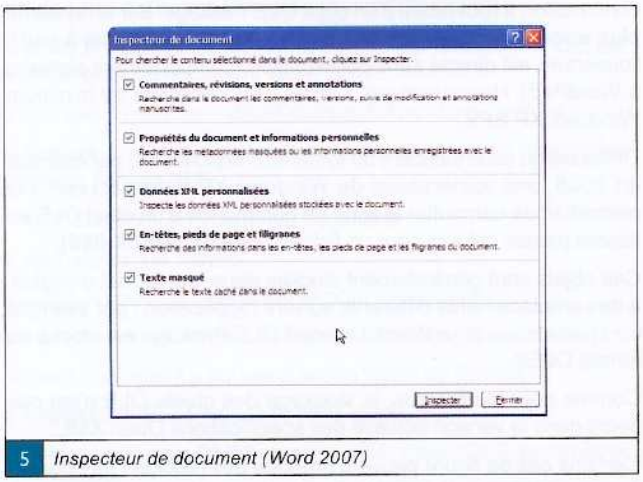
- méta-données : titre, auteur... ;
- marques de révision, suivi des modifications ;
- commentaires ;
- fichiers inclus (objets OLE) ;
- paragraphes et champs cachés ;
- formules.

Ces informations cachées sont problématiques lorsque des documents sont publiés vers l'extérieur ou bien dans le cadre de la signature électronique.

Office 2007 fournit un nouvel outil « **Inspecteur de document** » pour détecter et supprimer les différents types d'informations

cachées qui peuvent se trouver dans un document. Il s'agit d'une version améliorée de l'outil « RHDTool » qui pouvait être installé en option avec Office 2003.

C'est bien sûr une fonction extrêmement utile lorsqu'un document doit être publié, cependant les objets OLE ne sont pas considérés comme des données cachées.



5 *Inspecteur de document (Word 2007)*

Signature Numérique

Signature de document

L'intégrité et l'authenticité d'un document peuvent être assurées par l'emploi d'une signature numérique, associée au certificat X509 de l'utilisateur. Les spécifications de cette signature sont décrites dans **[OXSPEC]** partie 2. Elles sont donc communes à tous les formats basés sur OPC, comme XPS.

Ces spécifications s'appuient sur la recommandation XML-Signature du W3C **[XMLDSIG]**. C'est une signature de type *enveloppée*, c'est-à-dire stockée à l'intérieur du document lui-même.

Dans la pratique, on constate que seule une partie des fichiers est effectivement signée dans une archive Open XML. Les spécifications précisent que seuls les éléments visibles du document sont signés par l'application, et donc qu'il peut y avoir des parties modifiables.

Par exemple le fichier `[Content_Types].xml` n'est pas signé. Il est donc possible de modifier certains fichiers à l'intérieur de l'archive ou d'en ajouter sans rendre la signature invalide. On peut, par exemple, ajouter un fichier exécutable à l'intérieur de l'archive.

Il n'est cependant pas possible a priori de modifier la partie visible et utile d'un document ou d'y insérer un code malveillant qui s'activerait automatiquement.

L'intégrité complète du fichier Open XML lui-même n'est pas garantie. Il faut donc en tenir compte suivant le scénario d'emploi de la signature. S'il est nécessaire de protéger le document entier, il est préférable d'employer un mécanisme tiers de signature enveloppante ou détachée.

Signature de macros

Il est également possible de signer une macro pour assurer son authenticité et permettre son exécution sans demande de confirmation (si le certificat employé est approuvé comme éditeur de confiance).

Cette signature emploie des certificats différents de ceux utilisés pour signer les documents. Le format de la signature est lui aussi différent, non décrit dans les spécifications Open XML : un fichier binaire `vbaProjectSignature.bin` est ajouté dans l'archive, au même endroit que `vbaProject.bin`.

MS Office est fourni avec l'outil « Certificat numérique pour les projets VBA », qui permet de créer des certificats auto-signés. Un utilisateur peut ainsi signer lui-même des macros, sans besoin de PKI. Il faut cependant manipuler ces certificats avec prudence, car n'importe qui peut créer un certificat avec n'importe quel nom, au hasard « Microsoft ». Un utilisateur peut donc être abusé par une macro signée, s'il va jusqu'à accepter le certificat auto-signé dans son magasin personnel.

Chiffrement (protection par mot de passe)

Comme dans les versions précédentes d'Office, il est possible de chiffrer un document Word, Excel ou PowerPoint en fournissant un mot de passe. Au lieu d'employer les méthodes de chiffrement standard des archives ZIP ou encore XML-Encryption, Office 2007 sauvegarde alors le document Open XML chiffré dans un fichier au format binaire OLE2.

Les fichiers Open XML chiffrés emploient les mêmes extensions que leurs homologues en clair, mais ils ne peuvent pas être analysés par les mêmes outils ZIP et XML, même si le mot de passe est connu. La structure et les algorithmes de chiffrement ne sont pas publiés dans les spécifications Open XML. Un coup d'œil au contenu du fichier fait apparaître « *Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)* ».

Restrictions d'accès aux documents (DRM)

Lorsque le service WRM (*Windows Rights Management*) est installé, il est possible d'appliquer des restrictions de lecture et de modification aux documents, en suivant les mêmes principes que pour les contenus multimédias protégés par DRM (lecture limitée dans le temps, interdiction de retransmettre le contenu…).

Pour cela, les fonctionnalités IRM (*Information Rights Management*) d'Office nécessitent toutefois l'installation d'un serveur de gestion des droits sur le réseau ou bien l'utilisation d'un service gratuit sur Internet fourni par Microsoft pour évaluation.

Un autre article de ce dossier sur les technologies Microsoft décrit en détail ces fonctionnalités : cf. **[IRM]**.

Comme toute solution de DRM, le niveau de protection apporté par IRM reste faible, car les contrôles effectifs reposent toujours au final sur l'application locale. Si cette application est capable d'accéder au contenu du document à un moment donné, il suffit (en théorie) de modifier cette application sur disque ou en mémoire pour contourner les restrictions qu'elle impose.

L'exemple bien connu du format PDF qui propose des protections similaires montre également que des applications tierces (Ghostview, XPDF…) permettent parfois d'ignorer les restrictions.

Le format de ces restrictions n'est pas décrit dans les spécifications Open XML **[OXSPEC]**.

Robustesse face aux documents mal formés

Depuis quelques années les chercheurs en sécurité se sont penchés avec insistance sur MS Office et ses formats de documents binaires. De nombreuses vulnérabilités ont été découvertes **[CVEMSO]**, et certaines ont été exploitées pour diffuser des codes malveillants sur Internet. Cette relative « sensibilité aux vulnérabilités » de MS Office peut s'expliquer par la richesse des fonctionnalités de la suite bureautique et par la complexité de ses formats binaires.

Comme le nouveau format Open XML remplace la majeure partie des données binaires par des fichiers XML structurés, on peut espérer une réduction significative des vulnérabilités d'implémentation, au moins dans les fonctions de décodage de document. De plus, si la validation des données par schémas XML est systématique et suffisamment stricte, la probabilité qu'un document mal formé puisse déclencher du code malveillant devient beaucoup plus faible.

Cependant, il y a toujours un risque que certaines fonctionnalités soient mal conçues ou que les schémas XML soient trop laxistes, et Open XML peut contenir de nombreux formats binaires sujets à vulnérabilités (images bitmaps et vectorielles, OLE2...).

D'autre part, l'emploi de ZIP et XML avec des spécifications ouvertes laisse supposer que de nombreux logiciels tiers seront développés pour lire et écrire des documents au format Open XML. Étant donné la complexité du format, ces applications risquent également de souffrir de vulnérabilités.

Il est intéressant de noter que Microsoft vient de publier l'outil MOICE (*Microsoft Office Isolated Conversion Environment*) **[MOICE]**, pour protéger Office 2003 et 2007 contre les vulnérabilités exploitables via des documents dans des formats antérieurs à Open XML (OLE2, RTF...). Cet outil s'appuie sur le pack de compatibilité pour convertir systématiquement tous les documents en Open XML (et supprimer les macros) avant de les ouvrir dans la suite bureautique.

Conclusion sur la sécurité d'Open XML et MS Office 2007

Au vu de cette étude, le nouveau format Open XML pose autant de problèmes de sécurité que les anciens formats de MS Office, pour ce qui concerne le code malveillant ou la fuite d'informations. Et malgré certaines améliorations, la sécurité par défaut de MS Office 2007 n'est pas fondamentalement meilleure qu'avant. Pour certains points, comme le niveau de sécurité par défaut pour les macros, on pourrait même considérer qu'il s'agit d'une légère régression.

Même si le format Open XML est basé sur des spécifications ouvertes, il peut contenir des parties en format propriétaire non documenté (MS OLE2 ou BIFF par exemple). De plus, certaines fonctionnalités importantes, comme les macros, ne sont pas décrites dans le standard, et le format ne peut donc pas être considéré comme 100% ouvert.

Open XML est plus facile à analyser et à filtrer que les formats non ouverts. Cependant, sa structure est relativement complexe et riche, la tâche n'est donc pas si simple.

Comment se protéger ?

Pour se protéger contre l'entrée de code malveillant et la fuite d'informations par les documents bureautiques, il existe principalement deux solutions techniques, qui sont complémentaires :

- sécuriser le paramétrage des suites bureautiques ;
- filtrer les documents sur une passerelle d'interconnexion ou sur les supports amovibles.

Nous ne parlerons pas ici des solutions organisationnelles qui consistent, par exemple, à mieux informer l'utilisateur sur les dangers des codes malveillants et de la fuite d'informations cachées.

Paramétrage sécurisé d'Office 2007

Voici quelques principes généraux qui permettent d'améliorer la sécurité d'Office en fonction des besoins :

- Bien sûr, commencer par appliquer systématiquement les correctifs de sécurité et les mises à jour, si possible de façon automatique.
- Durcir le niveau de sécurité des macros et des objets ActiveX, en fonction des besoins réels des utilisateurs et de l'entreprise : choisir « désactiver toutes les macros SANS notification » si possible, ou « désactiver toutes les macros sauf signées » si la signature est employée.
- Utiliser la signature numérique des macros si besoin, si possible en s'appuyant sur une PKI (infrastructure de gestion de clés). À défaut, employer les répertoires de confiance (en les configurant finement).
- Éviter l'emploi de certificats auto-signés, sauf pour une utilisation exclusivement personnelle.
- Désactiver les notifications de la barre de messages pour bloquer les macros non signées.
- Alternativement, il est possible de désactiver totalement le moteur VBA (attention cependant aux effets secondaires sur certaines fonctionnalités...). Pour cela, positionner la clé de registre suivante :

```
HKLM\SOFTWARE\Microsoft\Office\12.0\Common\VBAOff = 1
```

- Désactiver les emplacements de confiance s'ils ne sont pas utilisés. Retirer au minimum les emplacements modifiables par les utilisateurs, à moins que les utilisateurs doivent écrire des macros sans pouvoir les signer.
- Protéger les paramètres de sécurité contre les modifications de l'utilisateur. Pour cela, la meilleure solution est d'employer les clés de registre MS Office de la ruche HKLM.
- En domaine, déployer ces paramètres de sécurité sur le parc de machines pour simplifier l'administration, par exemple à l'aide de GPO. Pour cela, chercher et télécharger « *2007 Office System Administrative Templates* » sur le site de Microsoft.
- Pour une machine hors domaine, la lecture de ces fichiers `.adm` permet de connaître l'emplacement exact des clés de registre HKLM à positionner.
- Se protéger contre les objets OLE Package, qui sont très rarement employés : interdire l'exécution de `C:\Windows\System32\Packager.exe`.

Pour plus de détails, Microsoft fournit une documentation reprenant en partie ces recommandations pour sécuriser un déploiement : **http://go.microsoft.com/fwlink/?LinkID=85671**.

Filtrage des documents

Les documents peuvent être filtrés sur une passerelle (messagerie, web, FTP...) ou sur un « sas de dépollution » pour supports amovibles. Il peut s'agir d'une simple analyse antivirus, ou bien d'un filtrage plus évolué, par exemple pour retirer tout contenu actif des documents (macros, scripts, objets OLE...) ou les informations cachées dans le sens sortant.

Open XML étant basé sur des technologies standards comme ZIP et XML, on peut imaginer qu'il est aisé d'analyser et filtrer ce format. En effet, on peut facilement localiser tous les éléments actifs comme les macros, les objets OLE, les scripts, etc.

Exemple de filtre pour Open XML

Pour retirer tout contenu actif :

- Macros : supprimer tout fichier `vbaProject.bin` et `vbaData.xml` ;
- Objets OLE : supprimer tout fichier `*.bin`.

Voici un exemple très simple de filtre en langage Python, qui se contente de supprimer les fichiers « à risque » dans un document :

```
import zipfile, sys
try:
        infile = zipfile.ZipFile(sys.argv[1], "r")
        outfile = zipfile.ZipFile(sys.argv[2], "w")
except:
        sys.exit("usage: %s infile outfile" % __file__)
for f in infile.infolist():
        fname = f.filename.lower()
        if  not fname.endswith(".bin") :
                data = infile.read(f.filename)
                outfile.writestr(f, data)
```

Contournement des filtres et antivirus

Il est tentant de filtrer ces formats ouverts avec des techniques simples, en employant par exemple un outil ou une bibliothèque permettant de manipuler les archives ZIP ou encore une recherche de texte dans les fichiers XML.

Un attaquant peut cependant camoufler les contenus malveillants de diverses façons, afin de contourner une passerelle de filtrage ou un antivirus. Pour cela, il suffit d'exploiter les fonctionnalités des suites bureautiques, d'XML et ZIP. Voici quelques exemples de camouflages possibles :

Renommage de documents Open XML avec macros

Tout d'abord il n'est pas suffisant de se fier aux extensions de fichiers : notamment même si un fichier `.docx` ne peut pas contenir de macros, il est toujours possible de renommer un fichier `.docm` en `.doc` pour faire exécuter des macros.

Par contre, la fonctionnalité étonnante qui permettait de renommer un fichier MS Office avec une extension inconnue (par exemple `.xyz`) et de toujours pouvoir l'ouvrir par double-clic n'est plus possible avec Open XML.

Renommage de macros VBA dans Open XML

À cause de la structure modulaire d'Open XML (OPC), il est possible de renommer le fichier de stockage des macros `vbaProject.bin` avec un nom quelconque. Par exemple, dans un document Word :

- renommer `vbaProject.bin` en `pasdemacrosici.txt` ;
- mettre à jour les relations dans `word/_rels/document.xml.rels` ;
- dans `[Content_Types].xml`, remplacer `bin` par `txt`.

Cette simple manipulation permet de contourner le filtre Python précédent, en conservant les macros actives. Il n'est donc pas possible de se fier aux noms des fichiers pour détecter la présence de macros dans Open XML.

Une solution plus sûre est d'employer un véritable *parseur* XML pour détecter les objets de type `vbaProject` et `vbaData` (ou `oleObject` pour les objets OLE). Une autre solution est d'analyser le contenu des fichiers à la recherche de formats binaires OLE2, avec de possibles faux-positifs.

Encodage US-ASCII et « bit de camouflage »

Comme Internet Explorer (cf. **[IEASCII]**), Office 2007 prend en charge l'encodage « US-ASCII » d'une façon bien étrange : tous les caractères dont le code ASCII est supérieur à 127 voient leur 8ème bit simplement retiré avant que le code XML soit analysé. Ce comportement permet donc un camouflage très simple pour contourner des filtres qui ne feraient pas cette vérification. Voici un exemple où les balises `<HIDDENTAG>` sont camouflées sur ce principe :

```
<?xml version="1.0" encoding="us-ascii" standalone="yes" ?>
¼HIDDENTAG¾ malware[...] ¼/HIDDENTAG¾
```

Encodage UTF-7

Sur le même principe, il est possible de camoufler des balises en employant un encodage UTF-7, et en utilisant des formes alternatives de caractères. Ceci est normalement interdit, mais Office 2007 l'autorise (tout comme Internet Explorer). Exemple :

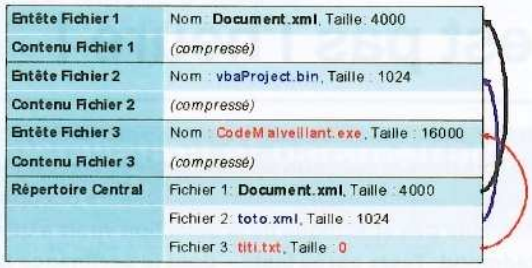
```
<?xml version="1.0" encoding="UTF-7" standalone="yes" ?>
+ADw-HIDDENTAG+AD4- malware[...] +ADw-/HIDDENTAG+AD4-
```

Pourtant les spécifications Open XML **[OXSPEC]** précisent bien que seuls les encodages UTF-8 et UTF-16 devraient être autorisés dans les fichiers XML...

Archives ZIP mal formées – noms de fichiers dupliqués

Dans une archive ZIP standard, les noms des fichiers sont dupliqués à 2 emplacements, dans le répertoire central à la fin de l'archive et dans l'en-tête de chaque fichier. C'est aussi le cas de la taille du fichier et d'autres informations. Il est possible de créer des archives ZIP mal formées en modifiant le nom d'un fichier dans un de ces emplacements uniquement. Beaucoup d'applications ne vérifient pas la cohérence de ces 2 noms, et certaines ne se fient qu'à l'un ou l'autre. Le schéma ci-après montre un exemple de fichier ZIP mal formé.

MS Office 2007 vérifie bien la cohérence entre les 2 noms. Cependant, s'il détecte une incohérence quelconque, il propose à l'utilisateur de réparer le fichier. Ce qui est plus étonnant encore, c'est que la correction se fait toujours de façon à rendre les macros exécutables, que l'on ait modifié le répertoire central ou les en-têtes !



6 *Exemple d'archive ZIP mal formée*

Cette technique permet donc de contourner tout filtre ou antivirus qui ne s'appuierait que sur les en-têtes ou le répertoire central pour son analyse.

Compression Zip64

Des améliorations du format ZIP ont été proposées ces dernières années, afin d'augmenter la taille maximale des archives ou le taux de compression. Les spécifications Open XML prévoient explicitement la possibilité d'employer le format Zip64. Il est donc nécessaire de prendre en charge ce nouveau format pour pouvoir analyser tous les documents.

Pour obtenir un filtre ou un antivirus robuste

L'analyse et le filtrage de fichiers Open XML ne doivent donc pas être pris à la légère, sous prétexte qu'il s'agit d'un format ouvert basé sur ZIP et XML.

Voici quelques points importants à vérifier pour une analyse sûre :

- Utiliser une bibliothèque ZIP robuste, capable de détecter les archives mal formées.
- Dans les fichiers ZIP, rejeter toute incohérence entre le répertoire central et les en-têtes de fichiers.
- Rejeter toute archive ZIP avec un format non supporté par la bibliothèque utilisée ou non prévu par les spécifications : Zip64, nouvel algorithme de compression, chiffrement...
- Utiliser systématiquement un parseur XML complet et robuste. Ne jamais se contenter d'une simple recherche de texte ou d'expressions régulières.
- Décoder les relations décrites dans les fichiers `.rels`, afin d'obtenir la structure réelle du document.
- Vérifier l'encodage des données XML : rejeter tout encodage non prévu dans les spécifications, et décoder les données de façon stricte pour rejeter les caractères anormaux.
- Si possible mettre à profit les schémas fournis par les éditeurs.
- Rejeter toute incohérence entre la structure interne et le nom du fichier (par exemple Open XML nommé `.doc`)

Conclusion

Le nouveau format bureautique Open XML est très séduisant, et son caractère ouvert est a priori très bénéfique en termes de sécurité. La détection de contenus actifs et d'informations cachées est notamment beaucoup plus facile qu'avant.

Cependant Open XML pose les mêmes problèmes de sécurité que les formats propriétaires utilisés jusqu'ici par Microsoft Office, et il n'y a pas lieu de se sentir plus à l'abri des contenus malveillants ou des fuites d'informations qu'auparavant.

Certaines nouveautés de ce format ou d'Office 2007 ajoutent même de nouveaux problèmes qu'il faudra prendre en compte, notamment quelques possibilités de camouflage dues aux formats ZIP et XML.

D'autre part, plusieurs fonctionnalités importantes en matière de sécurité comme les macros VBA, les objets OLE, le chiffrement de document ou les DRM ne sont pas documentées dans les spécifications ouvertes Open XML.

On peut penser qu'il faudra un certain temps avant que les antivirus et les logiciels d'analyse de contenu prennent en charge Open XML de façon satisfaisante. Cet article apporte notamment quelques pistes pour améliorer le filtrage de ce nouveau format.

Références

[OXSPEC] Ecma International, Office Open XML File Formats – Standard ECMA-376, **http://www.ecma-international.org/publications/standards/Ecma-376.htm**

[ECMA] Ecma International, National Body Comments from 30-Day Review of the Fast Track Ballot for ISO/IEC DIS 29500 (ECMA-376) « Office Open XML File Formats », Ecma/TC45/2007/006, **http://www.ecma-international.org/news/TC45_current_work/Ecma%20responses.pdf**

[ODSPEC] OASIS, Open Document Format for Office Applications (OpenDocument) v1.1, OASIS Standard, 1 Feb 2007, **http://docs.oasisopen.org/office/v1.1/OpenDocument-v1.1.pdf**

[MISC27] FILIOL (E.), FIZAINE (J.-P.), « Le risque viral sous OpenOffice 2.0.x », *MISC le journal de la sécurité informatique*, n°27, 09/2006.

[SSTIC07] LAGADEC (P.), « Sécurité des formats OpenDocument et Open XML (OpenOffice et MS Office 2007) », **http://actes.sstic.org/SSTIC07**

[PACSEC06] LAGADEC (P.), « *OpenOffice/OpenDocument and MS Open XML security, PacSec 2006 conference* », **http://pacsec.jp/psj06archive.html**

[XPS] Microsoft XML Paper Specification – XPS, **http://www.microsoft.com/whdc/xps/default.mspx**

[MARS] **http://labs.adobe.com/technologies/mars/**

[SSTIC03] LAGADEC (P.), « Formats de fichiers et code malveillant », SSTIC03, **http://actes.sstic.org/SSTIC03/Formats_de_fichiers/**

[OSSIR03] CHAMBET (P.) (EdelWeb), FILIOL (Eric) (ESAT), DETOISIEN (E.), « La fuite d'informations dans les documents propriétaires », OSSIR 6/10/2003, **http://www.ossir.org/windows/supports/2003/2003-10-06/OSSIR-Fuite%20infos.pdf**

[MS06-065] Secunia advisory for MS06-065, **http://secunia.com/advisories/20717**

[XMLDSIG] W3C, XML-Signature Syntax and Processing, **http://www.w3.org/TR/xmldsig-core/**

[IRM] MARTINEAU (T.) (ESAT), « RMS : évitez la fuite de vos informations », *MISC le journal de la sécurité informatique*, n°32, 07/2007.

[CVEMSO] *Common Vulnerabilities and Exposures*, mots-clés « Microsoft Office », **http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=microsoft+office**

[MOICE] *Microsoft Office Isolated Conversion Environment*, **http://support.microsoft.com/kb/935865/en-US**

[IEASCII] **http://www.securityfocus.com/archive/1/437948**

Évaluation de l'antivirus OneCare : quand avant l'heure ce n'est pas l'heure !

Lorsque la société Microsoft annonça, il y a près de deux ans, sa venue sur le marché de la lutte antivirale par le rachat de sociétés existantes (GeCAD), cette annonce a suscité une grande interrogation à la fois dans la communauté antivirale et dans le monde informatique. Si la maîtrise du système d'exploitation représente un indéniable avantage par rapport aux éditeurs occupant ce secteur de la sécurité, l'absence de savoir-faire et d'expérience dans le domaine très spécifique de la détection virale a quelque peu fait douter de la pertinence d'une telle orientation de la firme de Redmond. Tests après tests – dont la pertinence laisse souvent à désirer – le produit a reçu de très mauvaises notes et des jugements non moins exécrables. Au-delà du « sentiment » inspiré généralement par Microsoft, il est à ce jour difficile de démêler la justesse de ces résultats, de certains intérêts partisans d'une concurrence qui là a su, semble-t-il, s'organiser contre le nouveau venu. Cet article présente les résultats détaillés et reproductibles, pour la plupart, de l'évaluation technique de cet antivirus, menée en toute indépendance. Le jugement final est malheureusement sans appel et correspond à un avis récent de la société Microsoft elle-même sur son produit.

mots clés : ***antivirus / évaluation / schéma de détection / analyse de forme / analyse comportementale***

1. Introduction

Évaluer un antivirus n'est jamais une chose facile. La détection antivirale étant par nature un problème sans solution [1], il s'agit de déterminer à quel point un produit donné est imparfait.

En prenant le point de vue de l'attaquant – qui finalement est le seul vraiment déterminant –, il s'agit de mesurer la facilité avec laquelle ce produit peut être contourné techniquement et opérationnellement.

La plupart des tests généralement utilisés ne sont malheureusement ni reproductibles, ni pertinents. Bridés par la seule vision de la défense, ils se contentent d'étudier les produits vis-à-vis d'une menace connue. Il s'agit le plus souvent de regarder comment se comporte un produit sur une base de codes malveillants connus et d'enregistrer les taux de détection (lesquels ne sont bizarrement que très rarement à 100 %). Or la menace réelle évolue constamment et est par nature imprévisible – du moins en théorie, les attaquants faisant souvent montre d'un certain manque d'imagination, tant mieux pour nous. Cela oblige tout évaluateur à sortir des sentiers battus et à se mettre un tant soit peu dans la peau et l'esprit de l'attaquant.

En ce qui concerne la reproductibilité des tests utilisés, là encore il est pratiquement impossible de les rejouer... et cela oblige le lecteur à faire, faute de mieux, confiance à l'évaluateur. Or, un principe simple de simulabilité des tests [2] montre qu'il est très facile de choisir des échantillons de virus pour produire un résultat et un classement préalablement établis. Sans affirmer que cela est systématiquement le cas, l'absence de reproductibilité des tests incite au minimum à se poser la question, surtout lorsque d'un test à un autre des classements très différents, pour ne pas dire inverses, sont constatés.

Comme nous l'avons fait pour les autres produits antivirus, le test de OneCare s'est fait selon une méthodologie éprouvée, publiée et reproductible et pour une partie démontrée mathématiquement [3, 4, 5, 6, 7]. Outre l'étude vis-à-vis de la menace connue, nous nous sommes attachés à tester OneCare également face à une menace inconnue utilisant soit des techniques virales classiques, soit des techniques inconnues (faites « maison »), comme nous le faisons habituellement [10]. Ce sont les résultats de ces analyses qui sont présentées dans le présent article.

L'objectif n'est pas d'encenser indûment ou de démolir gratuitement un produit pour des motifs divers et variés, mais de donner sur une base rigoureuse et saine, en toute indépendance, un avis technique rationnel et constructif qui a pour but non seulement d'éclairer les utilisateurs, mais peut être également – et nous l'espérons – les développeurs de Microsoft, afin qu'ils sachent dans quelle direction éventuellement améliorer leur produit.

Il est essentiel de rappeler que OneCare n'est actuellement pas un produit d'entreprise et n'a probablement pas été pensé pour l'être. Il est dédié uniquement au marché des internautes à la maison ou éventuellement pour de petites entreprises (moins de 3 ordinateurs). Voyons le pour le moment comme une prise de marques dans un domaine où un grand éditeur ne peut rester absent, même s'il est pour le moment en retrait.

Sans que cela doive être pris comme une mesure absolue, il a été pertinent de comparer pour certains critères, les performances de OneCare avec celles d'un antivirus de référence – nous avons choisi l'antivirus Kaspersky pour son taux de détection, actuellement l'un des plus élevés du marché – et d'un antivirus grand public – Symantec Norton Antivirus. **Rappelons que tout antivirus pouvant être contourné de manière opérationnelle, ces comparaisons doivent être replacées dans leur contexte et ne sont que des instantanés sans autre réelle valeur que d'être une tendance globale.**

Nous tenons à remercier Bernard Ourghanlian et Cyril Voisin de chez Microsoft pour avoir facilité la réalisation de cette étude et apporté quand nous le souhaitions des réponses à nos questions. C'est d'autant plus important de le souligner que cette attitude n'est pas la règle dans la communauté antivirale.

disclaimer

Cet article présente des résultats de recherche et d'analyse. Ils ne représentent en rien une position officielle du ministère de la Défense dans quelque domaine que ce soit.

Eric Filiol, Philippe Evrard, Guillaume Geffard,François Guilleminot, Grégoire Jacob, Sébastien Josse, David Quenez
Laboratoire de virologie et de cryptologie
École Supérieure et d'Application des Transmissions
efiliol@esat.terre.defense.gouv.fr

2. Installation et paramétrage du produit

La version évaluée a les caractéristiques suivantes :

- version 1.5.1890.34 ;
- base de signatures 1.18.2548.9 (25 mai 2007) ;
- stratégie firewall 1.2.15.80.

Le logiciel a été fourni gratuitement par la société Microsoft.

Première constatation et souci : le produit ne peut s'installer hors connexion Internet. Un poste isolé, devant impérativement rester en permanence hors réseau, ne peut recevoir OneCare. Il n'est également pas possible de mettre à jour « à la main » (téléchargement de mises à jour moteur ou de bases de signatures et installation manuelle) comme le permettent la plupart des autres produits. Des réseaux sensibles d'entreprises devront voir ailleurs.

Il est en outre indispensable, pour activer le produit, de créer un compte « Microsoft Live ». Ne pas créer ce compte interdira cette activation.

Seconde déception : **aucun** paramétrage digne de ce nom n'est possible (types de fichiers analysés, niveaux de compression des archives, tailles des fichiers compressés....). De ce point de vue (voir figure 1), OneCare est un modèle de dépouillement.

I « Paramétrage » d'Onecare

Le produit est donc très (trop) directif et ne permet pas vraiment une administration et un paramétrage adaptés à une politique antivirale définie en local : c'est celle du produit et aucune autre. Toutefois, cela correspond à la volonté politique initiale de Microsoft. Les spécifications originelles du produit mentionnaient explicitement la volonté de simplifier au maximum l'interaction de l'utilisateur avec le produit afin de le mettre à la portée de tous. Il est vrai que pour nombre d'utilisateurs, les produits de sécurité sont dotés d'un paramétrage et d'une gestion complexes, trop complexes pour pouvoir être réellement mis à profit. Mais cela oblige alors à disposer d'un produit puissant, capable de réellement suppléer l'utilisateur. Ce n'est pas encore le cas avec OneCare.

En outre, la gestion des fichiers infectés ou suspects est quelquefois surprenante. Dans certains cas, alors que OneCare informe de la suppression d'un fichier infecté, le fichier est en réalité mis en quarantaine. Il n'y a que lors de l'analyse dynamique (à l'accès) que le fichier est toujours effectivement supprimé. Dans d'autres cas (notamment lors du traitement de certaines archives exécutables), OneCare supprime silencieusement et sans aucune alerte de l'utilisateur, les fichiers incriminés. Dans le cas de faux positifs, cela peut constituer un problème, car l'utilisateur ne pourra corriger et prévenir « manuellement » une erreur de décision de l'antivirus.

Le paramétrage du pare-feu intégré est également réduit à sa plus simple expression.

Cette première étape montre déjà une nette insuffisance du produit par rapport à la concurrence. La philosophie même du produit néglige le fait que la lutte antivirale est un couple indissociable constitué d'une politique antivirale et d'un antivirus.

3. Étude de l'analyse de forme

Rappelons que l'analyse de forme consiste à analyser un code hors de tout contexte d'exécution. Il s'agit de rechercher des structures ou méta-structures plus ou moins complexes d'octets. C'est, à l'heure actuelle, la technique encore majoritairement utilisée **[3, 4, 5]**.

3.1 Détection de codes malveillants connus

La première et triviale approche a été de scanner un certain nombre de codes malveillants très connus et réputés, de tous types. Là, une grosse déception en même temps qu'une surprise de taille pour un tel produit : le taux de détection global est mauvais.

- Sur un premier CDROM de 89 584 virus **[N0]**. La base de virus comporte un large spectre du simple macro-virus au ver polymorphe incluant des virus UNIX **[N1]**, DOS et Win16/32 (une majorité) compressés dans des archives au format ZIP.

	Virus détectés	% de détection
Microsoft Windows Live OneCare	26 291	29,35%
Kaspersky Antivirus[1]	89584	100 %
Symantec Norton Antivirus	25 593	28,57%

TI

Le produit éprouve de grandes difficultés lorsque le nombre de virus détectés est important (il arrive même que le produit n'indique

[1] *Rappelons cependant que, dans l'absolu, KAV n'atteint pas les 100%.*

aucune menace après avoir consommé une grande partie de la mémoire alors que la menace existe). Le test du scan intensif est assez difficile à réaliser avec OneCare sur une quantité importante de fichiers infectés (à quelques exceptions près, les autres produits n'ont pas ce problème).

⇨ Sur une archive connue (`Feliks1.exe`) de codes malveillants non moins connus (CIH, Back Orifice...), OneCare ne déclenche aucune alerte là où Kaspersky détecte immédiatement 168 programmes dangereux.

Les premiers résultats ont très vite indiqué que OneCare n'utilise que de la simple recherche de signatures. Cela sera confirmé tout au long de l'étude.

3.2 Extraction de schémas de détection

Afin de valider cette hypothèse, il a été procédé à une extraction du schéma de signature (motif de détection ET fonction booléenne de détection [3]). Les résultats suivants sur quelques membres de la famille Bagle résument bien la stratégie de détection utilisée par OneCare : les schémas de détection reposent effectivement sur les motifs plutôt courts (en tout cas plus que pour la plupart des concurrents) et la fonction de détection la plus faible qui soit (fonction logique ET).

Ajoutons que OneCare, du fait de ses choix technologiques, est sujet à de nombreuses fausses alarmes. Ses schémas de

Produits	Taille signature (octets)	Indice des octets
OneCare	6	0 – 1 – 60 – 200 – 201 – 206
Kaspersky	11	0 – 1 – 60 – 200 – 201 – 206 – 220 – 240 – 241 – 461 – 469
Symantec Norton	0	Fonction autre que ET

Schéma de détection Bagle.A

Produits	Taille signature (octets)	Indice des octets
OneCare	42	0 – 1 – 60 – 61 – 63 – 216...
Kaspersky	105	0 – 1 – 60 – 216 – 217 – 222 ...
Symantec Norton	0	Fonction autre que ET

Schéma de détection Bagle.E

Produits	Taille signature (octets)	Indice des octets
OneCare	75	0 – 1 – 61 – 62 – 63 – 144 – 145 – 146 – 147 – 164 ...
Kaspersky	3128	0 – 1 – 60 – 144 – 145 – 150 – 164...
Symantec Norton	8	0 – 1 – 60 – 144 – 145 – 150 – 164 – 445

Schéma de détection Bagle.J

Produits	Taille signature (octets)	Indice des octets
OneCare	80	0 – 1 – 60 – 61 – 62 – 63 – 128 – 129 – 130 – 131 ...
Kaspersky	54	0 – 1 – 60 – 128 – 129 – 548 ...
Symantec Norton	51	0 – 1 – 60 – 128 – 129 ...

Schéma de détection Bagle.N

Produits	Taille signature (octets)	Indice des octets
OneCare	84	0 – 1 60 – 61 – 62 – 63 – 128 ...
Kaspersky	59	0 – 1 – 60 – 128 – 129 – 546...
Symantec Norton	6	0 – 1 – 60 – 128 – 129 – 134

Schéma de détection Bagle.P

T2

détection ne sont de ce point de vue pas assez efficaces. Mais il faut aussi préciser que les concurrents de OneCare sont d'une part confrontés aux mêmes choix et que, globalement, tous les antivirus que nous avons testés étaient faibles vis-à-vis du test d'extraction de schémas de détection [3].

3.3 Tests techniques divers

Différents tests ont été conduits pour tester plus spécifiquement le comportement de OneCare dans le domaine de l'analyse de forme. Voyons les principaux résumés dans les tableaux suivants.

CATEGORIE	TEST	RESULTAT
Efficacité de la fonction d'analyse du système de fichier	Alternate Data Streams NTFS	Microsoft Windows Live OneCare est capable de détecter un fichier malicieux caché dans un flux NTFS
	Chemins longs NTFS	Microsoft Windows Live OneCare est capable de détecter un fichier dont le nom dépasse les 260 caractères, mais si celui-ci dépasse 5 250 caractères, le fichier ne sera pas analysé
	Analyse sur plusieurs systèmes de fichiers	Microsoft Windows Live OneCare détecte l'infection sur tous les supports (FAT, NTFS, CIFS, CDFS) et notifie l'utilisateur
	Analyse des « clusters défectueux »	Microsoft Windows Live OneCare ne détecte pas un code viral dissimulé dans un « cluster défectueux » du système de fichier NTFS

CATEGORIE	TEST	RESULTAT
Détection des rootkits	Détection de la corruption des objets du noyau (liste doublement chaînée des DRIVER_OBJECT et/ou EPROCESS, IDT, SSDT, Sysenter, DRIVER_OBJECT Major Function IRP Hook), de l'insertion d'un pilote filtre et des méthodes non conventionnelles de chargement d'un pilote.	Microsoft Windows Live OneCare ne détecte pas la manipulation directe des objets du noyau[2] et par conséquent ne notifie pas l'utilisateur
	Instrumentation des fonctions d'API en espace utilisateur par inline patching, global hook	Microsoft Windows Live OneCare ne détecte ni l'installation ni la présence de hooks sur les fonctions de l'API

CATEGORIE	TEST	RESULTAT
Analyse de la mémoire, exploitation de vulnérabilité mémoire	Exploitation d'un débordement de tampon (en local et via le réseau)	Microsoft Windows Live OneCare ne détecte pas le débordement de tampon
	Analyse de la mémoire	Microsoft Windows Live OneCare n'effectue pas d'analyse de la mémoire

T3

[2] *Utilisés par les composants noyau du système d'exploitation*

CATEGORIE	TEST	RESULTAT
Support et gestion des formats	Format d'archive, format d'archive corrompu	Microsoft Windows Live OneCare détecte le fichier sur la majorité des archives et notifie l'utilisateur. La profondeur maximale d'une archive détectable par le produit est de 62 niveaux **[N2]**
	Virus de macro	Microsoft Windows Live OneCare ne détecte ni l'exécution d'un script seul, ni l'exécution d'un script intégré à un document MS Office et par conséquent ne notifie pas l'utilisateur (script VBS, JS, VBE)

CATEGORIE	TEST	RESULTAT
Efficacité de la fonction de détection	Base virale	Taux de détection global (virus toutes plates-formes) : 29,35 %
	Exécutables packés	Microsoft Windows Live OneCare ne détecte qu'une partie des exécutables packés **[N3]**
	Exécutables polymorphes, Entry Point Obscuring	Microsoft Windows Live OneCare ne détecte pas les virus modifiés par application de techniques élémentaires d'EPO

T3 (suite)

3.4 Conclusion pour l'analyse de forme

En ce qui concerne l'analyse de forme, OneCare affiche globalement de mauvaises performances. La conclusion est qu'il est encore trop facile de contourner cet antivirus soit par de simples modifications d'un code connu, soit en utilisant des techniques virales connues peu sophistiquées. Mais si OneCare se distingue sur ce point, il faut noter d'une part que ce n'est pas le seul, et, d'autre part, que TOUS les produits peuvent ainsi être contournés.

4. Étude de l'analyse comportementale

La seconde partie de l'étude a été de tester les capacités de détection comportementale de OneCare. Rappelons qu'il s'agit de déterminer, au moment où le code est exécuté – à l'accès ou par émulation de code – si certains comportements sont douteux ou non.

4.1 Exécution de codes malveillants

4.1.1 Test de keyloggers

L'objectif est de tester la capacité du produit à détecter un *keylogger* utilisant la possibilité de l'empilement de drivers. Pour cela, la protection temps réel activée, le driver de notre keylogger est chargé via l'outil InstDriver et la réaction du produit est observée.

Nous utilisons Klog qui attache son driver à celui du clavier (`kbdclass`) et intercepte les données retournées par ce driver. Pour cela, Klog utilise le principe documenté des drivers en couche (*filter drivers*) : un driver peut être attaché à un autre et filtrer les données (*Interrupt Request Packet – IRP*) circulant depuis et vers le matériel. Klog crée le fichier `c:\klog.txt` et y stocke les touches capturées.

OneCare ne détecte pas l'installation du pilote de Klog et les touches saisies sont placées dans un fichier `klog.txt` sans aucune alerte.

4.1.2 Exécution de Back Orifice

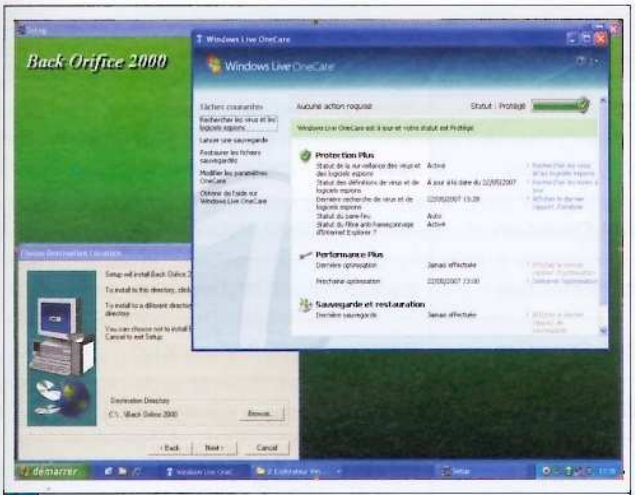
L'installation de Back Orifice 2000 se fait sans aucune alerte comme en témoigne la figure 2.

L'exécution de l'archive Félix1 se fait également sans alerte (figure 3)

L'analyse comportementale n'est donc, dans ce cas, pas utilisée et si aucune analyse de forme n'est possible, rien ne se passe alors.

4.2 Polymorphisme fonctionnel

Reprenant les travaux publiés dans **[4]**, il s'agit de faire muter le code d'un point de vue fonctionnel (changer des comportements par d'autres tout en conservant la finalité ultime du programme). Au stade actuel du projet, ces opérations sont réalisées manuellement.



2 *Installation de Back Orifice en direct*

3 *Exécution de l'archive Felix 1*

Comportement original	Référence de test
Charge finale	**CHARG**
Déni de service distribué à une date précise	CHARG_DDOS
Chargement d'une bibliothèque «backdoor»	CHARG_BDOOR
Duplication de code	**DUPLI**
Recopie dans un fichier sous un répertoire système	DUPLI_FSYS
Mise en résidence	**RESID**
Inscription sous une clé «run» du registre	RESID_RUN_KEY
Polymorphisme	**POLYM**
Chiffrement par XOR de la bibliothèque	POLYM_XOR_LIB
Chiffrement par substitution des chaînes de caractère	POLYM_SUB_STR
Test d'activité	**ACTIV**
Existence d'un mutex	ACTIV_MUTEX
Test de surinfection	**SURINF**
Existence d'une clé de registre	SURINF_REG_KEY

T4 *Comportements de référence de MyDoom (avant mutation)*

Nous avons choisi le ver *MyDoom* pour sa grande richesse fonctionnelle et ses comportements typiquement malicieux résumés dans le tableau 4.

Le tableau 5 récapitule les principales modifications fonctionnelles opérées sur MyDoom.

Le tableau 6, page suivante, résume les résultats de détection par OneCare.

Comportement modifié	Référence de test
Charge finale	**CHARG**
Cible différente et gâchette	CHARG_TRIG_TARG
Suppression de la bibliothèque «backdoor»	CHARG_NO_BDOOR
Duplication de code	**DUPLI**
Recopie dans un fichier de raccourci	DUPLI_SH_CUT
Recopie compressée sous un nom système	DUPLI_NAM_SYS
Mise en résidence	**RESID**
Inscription sous les clés services du registre	RESID_SERV_KEY
Modification de win.ini	RESID_WIN_INI
Polymorphisme	**POLYM**
Chiffrement par flot de la bibliothèque	POLYM_FLOW_LIB
Chiffrement par flot des chaînes de caractère	POLYM_FLOW_STR
Bibliothèque intégrée en clair sous un nom différent	POLYM_PLAIN_LIB
Sans chiffrement des chaînes de caractères	OLYM_PLAIN_STR
Test d'activité	**ACTIV**
Existence d'un événement	ACTIV_EVENT
Existence d'un mutex différent	ACTIV_DIFF_MUT
Test de surinfection	**SURINF**
Existence d'une clé de registre différente	SURINF_DIF_KEY
Existence d'un fichier «superhidden»	SURINF_SUP_HID
Existence d'une variable d'environnement	SURINF_ENV_VAR

T5 Références des nouvelles versions testées

Comportement	Version	Analyse statique	Protection dynamique
(aucun)	Souche originale	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	Bibliothèque shimgapi.dll	TrojanProxy:Win32/ProDoom	N/A
CHARG	CHARG_TRIG_TARG	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	CHARG_NO_BDOOR	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
DUPLI	DUPLI_SH_CUT	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	DUPLI_NAM_SYS	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
RESID	RESID_SERV_KEY	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	RESID_WIN_INI	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
POLYM	POLYM_FLOW_LIB	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	POLYM_FLOW_STR	Non detecté	Non detecté
	POLYM_PLAIN_LIB	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	POLYM_PLAIN_STR	Non detecté	Non detecté
ACTIV	ACTIV_EVENT	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	ACTIV_DIFF_MUT	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
SURINF	SURINF_DIF_KEY	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	SURINF_SUP_HID	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
	SURINF_ENV_VAR	Backdoor:Win32/MyDoomGen	Backdoor:Win32/MyDoomGen
DUPLI	DUPLI_NAM_SYS_V1	Non detecté	Non detecté
RESID	RESID_WIN_INI_V1	Non detecté	Non detecté
SURINF	SURINF_SUP_HID_V1	Non detecté	Non detecté

T6

Afin de différencier ce qui relève purement de l'analyse de forme – le paramétrage ne permet pas de séparer les deux techniques de détection –, une phase de détection statique a d'abord été opérée, suivie d'une ou plusieurs phases dynamiques.

Phase statique

L'ensemble des différentes variantes sont détectées statiquement à l'exception de celles où le chiffrement des chaînes est soit modifié à l'aide d'un autre algorithme, soit supprimé **[POLYM_FLOW_STR, POLYM_PLAIN_STR]**. Nous pouvons donc en conclure que la signature doit être localisée dans une ou plusieurs chaînes de caractères.

Phase dynamique

Les résultats sont identiques au mode statique. Ce phénomène est assez logique puisqu'une des défenses dynamiques déployées par les antivirus est l'analyse de signature préemptive. Nous allons donc nous focaliser sur la seule variante n'ayant pas été détectée **[POLYM_FLOW_STR]**.

Cette forme n'est pas détectée même dynamiquement et pourtant cette souche utilise les techniques courantes déployées par différents *malware* :

⇨ Simple duplication par l'appel successif à `GetModuleFileName` avec un paramètre `NULL` suivi d'un `CopyFile`.

➪ Tentative de mise en résidence en s'inscrivant sous une clé de `run` du registre.

➪ Chargement d'une bibliothèque « backdoor ». Pourtant, au moment de son extraction du corps du malware, la signature reste présente. Il se peut que cette dernière ne soit de nouveau qu'une chaîne de caractères du fichier.

➪ Connexion au réseau sur un port SMTP. Une remarque néanmoins concernant cette connexion. Le pare-feu émet tout de même une alerte, mais sans aucune interprétation. Il s'agit d'une simple demande de routine déclenchée par défaut permettant à l'utilisateur de configurer sa politique.

On peut dire que l'envoi de mail n'est pas malhonnête en soi et que de nombreux clients de mails observent un comportement similaire. Néanmoins, la combinaison des trois facteurs reste caractéristique des virus de mail. La combinaison de ces comportements aurait dû alarmer l'antivirus.

Deuxième phase dynamique

Dans cette deuxième phase, nous avons réutilisé la souche de chiffrement par flot des chaînes afin d'être sûr de contourner la signature **[POLYM_FLOW_STR]**. À partir de cette version, nous en avons généré trois nouvelles **[DUPLI_NAM_SYS_V1, RESID_WIN_INI_V1, SURINF_SUP_HID_V1]** en migrant les modifications apportées dans les précédentes, mais qui malheureusement conservaient la signature **[DUPLI_NAM_SYS, RESID_WIN_INI, SURINF_SUP_HID]**.

Ces trois nouvelles versions ne sont de nouveau plus détectées alors qu'elles touchent à des points sensibles, en modifiant le fichier `Win.ini`, en réutilisant un nom de fichier système (`svchost.exe`) et en se recopiant sous un des répertoires compressés de désinstallation Windows (`$NtUninstallKB904706$`) ou encore en créant des fichiers super cachés qui sont une combinaison de fichier caché avec l'attribut système d'activé.

En conclusion, ces résultats prouvent que la détection comportementale est absente ou bridée par la recherche de signatures très simples.

5. Analyse d'une attaque générique

Une attaque générique, mettant en œuvre des techniques malveillantes connues et peu sophistiquées, a été testée sur un mini réseau, en conditions opérationnelles (de la pénétration à la prise de contrôle du réseau). Cette attaque a permis de tester simultanément l'antivirus ET le pare-feu.

Pour cela, nous avons utilisé un « cheval de Troie » développé par nos soins et donc non connu des éditeurs et des laboratoires d'analyses.

Ce cheval de Troie est composé de 3 parties dont 2 constituent une application client-serveur, la troisième partie étant le vecteur de l'attaque sous forme d'un « contenu anodin à contexte d'exécution

		CLIENT	SERVEUR	Vecteur d'attaque	Remarques
ANTIVIRUS	OFFICESCAN	🟢	🟢	🟢	Infection transparente
	AVAST	🟢	🟢	🟢	Infection transparente
	KASPERSKY	🟢	🟢	🔵	Contrôle du vecteur
	NORTON AV	🟢	🟢	🟢	Infection transparente
	ONE CARE	🟢	🟢	🟢	Infection transparente
PARE-FEU	WINDOWS XP	🟢	🔵	-	
	ZONE ALARM	🔵	🔵	-	
	ONE CARE	🔴	🔵 (🔴 selon réglage)	-	Impossibilité d'autoriser le client
	NORTON AV	🟢	🔵	-	
ANTI-SPYWARE	AD-AWARE	🟢	🟢	🟢	
	SPYBOT	🟢	🟢	🟢	

🟢	Aucune réaction
🔵	Avertissement
🔴	Blocage

caché ». L'application client-serveur met en œuvre des « services » classiques d'un cheval de Troie à savoir :

- écoute du clavier ;
- récupération d'une copie écran ;
- liste des processus en cours ;
- liste des programmes installés ;
- extinction du PC ;
- redémarrage du PC ;
- verrouillage de la station ;
- exécution d'une commande *shell* à distance ;
- usurpation du mot de passe de l'utilisateur ;
- recherche des adresses mails sur les disques durs ;
- lecture, écriture, modification et suppression dans la base de registre ;
- ouverture/fermeture des lecteurs de CD-ROM ;
- envoi de messages *popup* ;
- envoi de frappes de touches à la place de l'utilisateur…

Au niveau de l'antivirus, OneCare n'est pas pire que ses concurrents dans la détection d'une attaque générique. Tous ont, de ce point de vue, montré la même faiblesse (le message de KAV n'est pas suffisamment pertinent pour permettre à la grande majorité des utilisateurs de prendre une décision avisée).

La bonne surprise a été le très bon comportement du pare-feu de OneCare, à vrai dire le plus efficace (plus que celui de XP d'ailleurs). Si les messages d'alerte laissent malgré tout l'utilisateur choisir, il y a au moins un traitement de l'alerte.

Conclusion

L'antivirus OneCare est donc en retrait par rapport à ses concurrents et, à ce jour, il représente une solution encore imparfaite pour concurrencer sérieusement les produits bien établis, même si aucun de ces derniers – rappelons-le – ne peut être considéré comme une protection parfaite.

Il convient d'ailleurs d'insister sur le fait que s'il est impératif de disposer d'un antivirus, c'est avant tout la politique de sécurité antivirale qui sera déterminante.

De ce point de vue, rappelons cependant que OneCare n'est pas seulement un antivirus : il s'agit d'un produit « tout en un » (antivirus, pare-feu, antispyware, outils d'optimisation des performances, outil de sauvegarde et de restauration). Cela replace la lutte antivirale dans la perspective plus générale qui devrait être la sienne.

L'explication a été en fait partiellement révélée par Arno Edelman, responsable produit sécurité en Europe pour Microsoft, lors du CeBit 2007 [8]. Selon lui, « OneCare est un nouveau produit. Ils n'auraient pas dû le sortir quand ils l'ont fait, mais ils règlent les problèmes maintenant […] Microsoft n'est pas une société de sécurité. La sécurité est importante, mais ne représente qu'une petite partie de Microsoft. »

Est-elle admissible et cela augure t-il une amélioration conséquente des capacités antivirales (avec la prochaine version 2.0) ? Ce n'est pas sûr. La firme de Redmond prend le train peut-être trop tard, à un moment où une crise conceptuelle majeure s'annonce dans le monde de la technologie antivirale : l'avènement de nouvelles algorithmiques virales (virus k-aires, techniques de furtivité à base de virtualisation, métamorphisme évolué… [5]) va très probablement obliger les éditeurs d'antivirus à revoir en profondeur le concept même de moteur antiviral [9]. Et l'expérience ne sera pas le moindre des avantages dans cette petite révolution… à moins que Microsoft, après GeCad, ne rachète la société Kaspersky.

Cependant, l'idée d'un éditeur de système d'exploitation produisant son propre antivirus est loin d'être absurde. La connaissance intime de son propre système représente un avantage de taille par rapport un éditeur d'antivirus, lequel devra toujours développer son produit pour une « boîte noire ». La voie est alors peut être celle d'une ouverture plus grande du système et une collaboration plus étendue entre les deux mondes. Quoi qu'il en soit, souhaitons bonne chance à OneCare et attendons ses prochaines versions.

Notes

[N0] Il s'agit d'un CD-ROM constitué de virus téléchargés sur le site **vx.netlux.org**, site, rappelons le, indépendant. Ce CD-ROM a été constitué en 2005.

[N1] La présence de codes malveillants UNIX est motivée par le fait qu'il peut être intéressant de détecter des codes pour d'autres OS lorsqu'ils sont présents sur un support amovible (clef USB par exemple). Cela peut contribuer à accroître la sécurité dans le cas de réseaux hétérogènes.

[N2] Les formats d'archives testés sont les formats ARC, ARJ, BASE64, BZIP2, GZIP, LHA, MS-CABINET, MS-COMPRESS, UPX, UPX illégal, UUE, ZIP, ZIP en-tête corrompu, ZIP imbriqué (30 niveaux), RAR, RAR imbriqué (62 niveaux max) pour lesquels il y a détection, ainsi que les formats LZO, MIME, TAR, pour lesquels il n'y a pas détection.

[N3] Ont été testés les packers Armadillo 4.05, AsProtect 1.23 RC4, Petite 2.3, Shrinker 3.4 (non détectés) et UPX 1.24w, UPX corrompu (archphase/NWC) et YC 1.2 (détectés).

Références

[1] FILIOL (Éric), *Les virus informatiques : théorie, pratique et applications*, collection IRIS, Springer France, 2004.

[2] FILIOL (Éric), « La simulabilité des tests statistiques », *MISC – Le journal de la sécurité informatique*, numéro 22, novembre 2005.

[3] FILIOL (Éric), « *Malware pattern scanning schemes secure against black-box analysis* », Special Issue EICAR 2006, V. Broucek and P. Turner eds, *Journal in Computer Virology*, 2, p. 35—50, 2006.

[4] FILIOL (Éric), JACOB (Grégoire) et LE LIARD (Mickaël), « Evaluation Methodology and Theoretical Model for Antiviral Behavioural Detection Strategies », Special Issue of WTCV 2006, G. Bonfante et J.-Y. Marion eds, *Journal in Computer Virology*, 3, p. 23—37, 2007.

[5] FILIOL (Éric), *Techniques virales avancées*, collection IRIS, Springer France, 2007.

[6] JOSSE (Sébastien), « *How to assess the effectiveness of your anti-virus ?* », Eicar 2006 Special Issue, V. Broucek and P. Turner eds, *Journal in Computer Virology*, 2 (1), p.51 – 67, 2006.

[7] JOSSE (Sébastien), « *Secure and advanced unpacking using computer emulation* », *Journal in Computer Virology*, 3 (3), à paraître, 2007.

[8] HERMANN (Vincent), « Microsoft : il manque des morceaux à Live OneCare », **http://news.zdnet.co.uk/security/0,1000000189,39286351,00.htm?r=8**

[9] FILIOL (Éric), « *Formal Model Prososal for Stealth (Malware) Programs* », *Virus Bulletin Conference 2007*, Vienne, 2007.

[10] FILIOL (Éric), « Évaluation des logiciels antivirus : quand le marketing s'oppose à la technique », *MISC – Le journal de la sécurité informatique*, numéro 21, septembre 2005.

Conception et architecture de la bibliothèque cryptographique d'OpenSSL

OpenSSL est une bibliothèque bien connue des habitués de la sécurité, en premier lieu pour son implémentation du protocole SSL, mais également en tant qu'outil cryptographique libre. Techniquement, cette bibliothèque est de bon niveau et les développeurs se sont montrés assez réactifs lors de la découverte de failles de sécurité dans le code. Là où le bât blesse, c'est la documentation.

mots clés : ***cryptographie / bibliothèque / OpenSSL***

Il est assez étonnant de trouver une bibliothèque aussi notoire et répandue, avec une documentation aussi peu garnie : outre le nombre important de fonctions non documentées, elle contient également toute une panoplie d'erreurs de mise à jour ou d'utilisation. La documentation même du site web (**http://www.openssl.org/docs**) semble n'être mise à jour qu'épisodiquement. Elle est heureusement complétée par quelques livres et articles intéressants **[IBM] [Res01]**, mais ceux-ci se limitent généralement à quelques zones typiques comme l'utilisation des commandes en ligne d'OpenSSL ou la mise en place de *sockets* SSL. De surcroît, ils sont bien souvent partiellement obsolètes.

Cet article ne prétend pas, à lui seul, résoudre tous les problèmes de documentation d'OpenSSL, mais il tente au moins de lever le voile sur une zone d'ombre de la bibliothèque : son architecture. Plus précisément, il détaille les grandes parties qui constituent la partie cryptographie de la bibliothèque (par exemple, la bibliothèque des grands nombres), comment ces parties interagissent, ainsi que leurs principes de conception et d'utilisation.

Organisation typique d'une sous-bibliothèque de cryptographie

Le répertoire de la bibliothèque de cryptographie d'OpenSSL (`./crypto`) est constitué de plusieurs sous répertoires (`aes`, `asn1`, `bf`...) qui regroupent des entités fonctionnelles. Dans cet article, ces entités sont appelées « sous-bibliothèques », OpenSSL n'y associant pas de nom particulier.

Les sous-bibliothèques de la bibliothèque de cryptographie d'OpenSSL possèdent toutes une organisation interne semblable :

⇨ Un fichier en-tête (ex. : `evp.h` pour EVP) qui regroupe toutes les fonctions exportées de la sous-bibliothèque. C'est son interface. Ce fichier est également copié dans le répertoire `./include`.

⇨ Un fichier d'en-tête `xxx_lcl.h` ou `xxx_locl.h` (ex. : `evp_locl.h`). Il s'agit d'un fichier d'en-tête interne au module (`lcl` et `locl` voulant probablement dire « local »). L'appelant n'a pas besoin de l'inclure, mais le code source y fait référence. Ce fichier contient typiquement des macros et tableaux utilisés en interne.

⇨ Un fichier source `xxx_lib.c` (ex. : `evp_lib.c`). Ce fichier implémente généralement les fonctions les plus simples de l'interface de la sous-bibliothèque, et effectue le travail d'aiguillage approprié pour les fonctions plus compliquées. Le noyau dur de l'implémentation de la sous-bibliothèque n'est généralement pas dans ce fichier.

⇨ Un fichier `xxx_err.c` (ex. : `evp_err.c`) qui contient les chaînes nécessaires pour l'affichage de messages d'erreurs explicites. Ce fichier définit généralement deux tableaux : `XXX_str_functs` qui liste le nom de toutes les fonctions qui peuvent retourner des erreurs (afin d'afficher le nom de la fonction), et `xxx_str_reasons` qui liste les messages (raisons) d'erreurs spécifiques au module. Par exemple, le module `RAND` (PRNG) contient un message spécifique explicite indiquant que l'appelant n'a pas fourni de *seed* au PRNG. Ces deux tableaux sont chargés par la fonction `ERR_load_XXX_strings` qui est la seule fonction exportée par le fichier `xxx_err.c`.

⇨ Un fichier `xxx_test.c` (ex. : `evp_test.c`), également recopié dans le répertoire `./test`, qui teste certains aspects de la sous-bibliothèque. La plupart du temps, les tests sont assez succincts (ne pas s'attendre à une batterie de tests unitaires !). Ils montrent plutôt un cas d'utilisation courante de la sous-bibliothèque. Certaines fois, le fichier test n'existe pas.

⇨ Un `Makefile`.

⇨ Et, suivant la sous-bibliothèque, un ou plusieurs autres fichiers d'implémentation.

L'interface de chaque sous-bibliothèque peut être implémentée de plusieurs manières différentes. Par exemple, la sous-bibliothèque RSA repose par défaut sur l'implémentation d'Éric Young (`./crypto/rsa/rsa_eay.c`), mais d'autres sont tout à fait envisageables comme des implémentations reposant sur un accélérateur cryptographique ou sans utilisation du *Chinese Remainder Theorem*.

Pour identifier chaque implémentation, on retrouve dans plusieurs sous-bibliothèques de cryptographie le même principe conceptuel : chaque implémentation est identifiée par une structure `xxx_METHOD` (ex. : `RAND_METHOD`, `RSA_METHOD`...), définie dans le fichier en-tête de la sous-bibliothèque. Cette structure regroupe un ensemble de pointeurs de fonctions qui réalisent les principales actions de la sous-bibliothèque. Par exemple, pour RSA, `RSA_METHOD` comprend des pointeurs de fonctions `rsa_pub_enc` (chiffrer avec une clé publique), `rsa_pub_dec` (déchiffrer avec une clé publique), `rsa_priv_enc` (chiffrer avec une clé privée), `rsa_priv_dec` (déchiffrer avec une clé privée), `rsa_mod_exp` (faire une exponentiation modulaire), etc.

L'utilisation de cette structure `xxx_METHOD` a l'avantage de clarifier ce qu'un développeur doit implémenter s'il souhaite fournir sa propre implémentation de la sous-bibliothèque. Dans le cas de RSA – évoqué ci-dessus – il lui suffira donc de coder chaque fonction référencée dans `RSA_METHOD` puis, finalement, d'instancier une structure de type `RSA_METHOD` et de renseigner chacun de ses pointeurs de fonction.

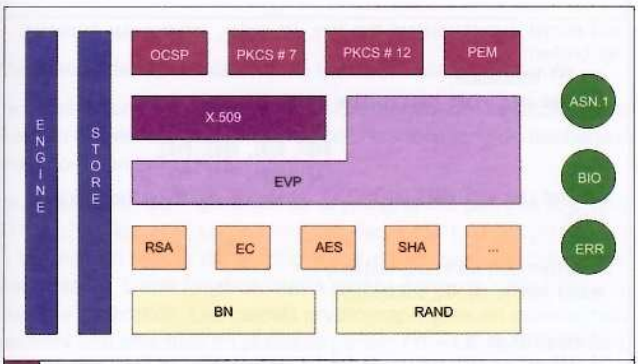
Pour l'appelant, le choix de l'implémentation à utiliser se fait sans difficulté particulière : soit il ne spécifie rien et OpenSSL se

débrouille tout seul par défaut, soit il sélectionne la méthode à utiliser à l'aide d'une fonction `xxx_set_method` :

```
RSA *rsa = RSA_new();
RSA_set_method(rsa, RSA_PKCS1_SSLeay());
...
RSA_public_encrypt(flen, from, to, rsa, padding);
...
RSA_free(rsa);
```

Les différentes couches de sous-bibliothèques de cryptographie

Maintenant que nous connaissons les grandes lignes de conduite d'une sous-bibliothèque, détaillons l'organisation de celles-ci les unes par rapport aux autres. Grosso modo, il y a trois couches différentes de sous-bibliothèques : la couche de bas niveau pour les fondements de la cryptographie, la couche intermédiaire pour le gros de l'implémentation de la cryptographie et les couches de haut niveau qui offrent des fonctionnalités cryptographiques usuelles.



1 *Organisation des différentes sous-bibliothèques « crypto » d'OpenSSL*

Sous-bibliothèques de bas niveau

Au plus bas niveau, on trouve la gestion des grands nombres (*Big Number*) et la génération de nombres aléatoires (`RAND`). Il s'agit de briques fondamentales pour la cryptographie, sans lesquelles l'implémentation des algorithmes serait impossible, inefficace et/ou non sécurisée. Par exemple, la performance de RSA tient beaucoup à une implémentation astucieuse de BN, tandis que la sous-bibliothèque `RAND` est un des piliers de la sécurité de la génération de clés.

La sous-bibliothèque BN est une bibliothèque de calculs mathématiques sur de grands entiers – des entiers qui ne tiendraient pas sur le type `int` ou `long` ! – ainsi que sur le corps GF(2^m) – c'est-à-dire le calcul sur des polynômes dont le degré est au plus de m-1 et dont tous les coefficients sont binaires (soit 0 soit 1), par exemple $x^5 + x + 1$. Ces polynômes sont en effet représentés sous forme de vecteur de bits, et leur manipulation est semblable à celle de grands entiers.

Autant l'implémentation de BN est quelque chose d'assez pointilleux et rusé, autant son utilisation s'avère assez simple : un grand entier est représenté par le type `BIGNUM`. Il doit être alloué (`BN_new()`) et initialisé (`BN_init()`) avant toute utilisation, et libéré (`BN_free()`) lorsqu'il n'est plus utile. Sa taille est « illimitée » : l'implémentation alloue dynamiquement autant de mémoire que nécessaire. Certaines opérations nécessitant beaucoup de calculs et variables intermédiaires, afin d'éviter d'effectuer sans cesse des allocations/libérations, il faut utiliser des contextes (`BN_CTX`). Les `BN_CTX` sont donc à comprendre comme des réserves de BIGNUM utilisables par l'implémentation. C'est notamment le cas de l'exponentiation modulaire (`BN_mod_exp`) réalisée ci-dessous (code n°1).

Enfin, un dernier point souvent laissé dans l'ombre : en cryptographie, une clé est une série d'octets. Il faut donc convertir cette série d'octets en un grand entier. La sous-bibliothèque BN offre pour cela la fonction `BN_bin2bn`. Elle suppose que la série d'octets est représentée en mode Big Endian, c'est-à-dire avec les octets les plus significatifs en premier (ci-dessous, 0x8d est donc l'octet de poids le plus élevé). Dans l'autre sens, la fonction à utiliser est naturellement `BN_bn2bin`. BN propose également des fonctions `BN_hex2bn`, `BN_bn2hex`, `BN_dec2bn` (etc.) pour convertir des chaînes hexadécimales (ou décimales pour `BN_dec2bn`) en grands entiers. Ces fonctions servent généralement pour facilement fournir des valeurs de test.

```
#include <openssl/bn.h>

/* valeurs de tests - enlever les '...' ! */
const unsigned char P[] = { 0x8d, 0xf2, 0xa4, 0x94, 0x49, 0x22, ... };
const char G[] = "626d0278...";
const char X[] = "2070b322...";

BIGNUM *y, *g, * x, *p;
BN_CTX *ctx;

/* allocation des grands entiers et du contexte */
y = BN_new(); g = BN_new(); x = BN_new(); p = BN_new();
ctx = BN_CTX_new();

/* initialisation */
BN_init(y); BN_init(g); BN_init(x); BN_init(p);

/* conversion des valeurs */
BN_bin2bn(P, sizeof(P), p);
BN_hex2bn(&g, G);
BN_hex2bn(&x, X);
```

```
/* calcul: y=g^x mod p */
BN_mod_exp(y, g, x, p, ctx);

/* libération */
BN_free(y); BN_free(g); BN_free(x); BN_free(p);
BN_CTX_free(ctx);
```

Pseudo-code n°1 réalisant une exponentiation modulaire

La sous-bibliothèque `RAND` d'OpenSSL a l'avantage de réellement offrir un RNG (*Random Number Generator*) et pas seulement un PRNG (*Pseudo Random Number Generator*) comme dans beaucoup d'autres bibliothèques. Plus précisément, elle offre un PRNG qui, par défaut, repose sur l'appel itératif d'une fonction de hachage et différentes manières d'initialiser ce PRNG. Sous Unix, l'initialisation se fait soit par un *Entropy Gathering Daemon* (EGD), soit par `/dev/urandom` s'il est disponible, soit par ARC4. Sous Windows, il provient par défaut soit d'un fournisseur cryptographique (CSP – *Cryptographic Service Provider*), soit de diverses informations système comme les statistiques réseau, le *handle* de la fenêtre en avant plan, la structure d'information du curseur (`CURSORINFO`), du tas (`HEAPLIST32`, `HEAPENTRY32`), de la liste des processus (`PROCESSENTRY32`), *threads* (`THREADENTRY32`) et modules (`MODULEENTRY32`), de la structure d'usage mémoire (`MEMORYSTATUS`), de la structure du processus courant. L'utilisateur est libre d'y combiner en plus des données provenant du mouvement de la souris (appel de `RAND_event()`) ou de données affichées à l'écran (appel de `RAND_screen()`).

Donc, lorsque nous utilisons l'implémentation par défaut, l'appel direct à `RAND_bytes()` ou à `RAND_pseudo_bytes()` fournit *directement* des données aléatoires de qualité raisonnable (du moins pour les systèmes sous Windows, WinCE, VMS, OS2, et les Unix disposant d'un EGD ou de `/dev/urandom`. Pour Netware et OpenBSD, l'initialisation par défaut paraît curieusement un peu plus légère et pourrait sans doute être améliorée). Il n'est alors pas strictement utile d'appeler `RAND_seed()`, `RAND_event()` ou `RAND_screen()` pour initialiser le PRNG à partir de données configurables, d'événements ou ce qui se trouve à l'écran. Il est à noter ici que la page de manuel correspondante semble incorrecte ou obsolète. Sa traduction à la volée donnerait : « OpenSSL prend garde à ce que l'état du PRNG soit unique pour chaque thread. Sur des systèmes qui fournissent un `/dev/urandom`, ce dernier est utilisé pour initialiser le PRNG de manière transparente. En revanche, pour tous les autres systèmes, l'application est responsable de l'initialisation du PRNG en appelant `RAND_add()`, `RAND_egd()` ou `RAND_load_file()`. » Dans la version 0.9.8d, ceci est faux pour l'implémentation par défaut : le PRNG est initialisé de manière transparente pour **tous** les systèmes d'exploitation, et ceci de manière acceptable pour bon nombre d'entre eux (voir plus haut). Une bonne pratique est de quand même toujours appeler `RAND_add()` ou `RAND_seed()`, afin d'ajouter encore un peu d'entropie supplémentaire, mais c'est moins important (bien entendu, en réalité, cela dépend du niveau de sécurité que vous souhaitez atteindre). Sous Windows, le code ci-dessous est donc parfaitement acceptable pour la plupart des situations :

```
const unsigned char string[] = "Une chaîne pour rajouter un petit peu d'entropie";
RAND_seed(string, sizeof(string));
```

Ce code *ajoute* un peu plus d'entropie au PRNG qui, de toute manière, a été initialisé plus sérieusement (avec des données système) à la première utilisation. À vrai dire, l'entropie ajoutée par ces deux lignes est tellement ridicule qu'il est peut-être tout aussi simple de ne pas les écrire ! En revanche, appeler `RAND_event()` ou `RAND_screen()` serait plus utile.

Les algorithmes de cryptographie

Au-dessus de ces fondements, on trouve une multitude d'algorithmes symétriques, asymétriques et fonctions de hachage (AES, DES, RSA, DSA, algorithmes à courbes elliptiques ECDH et ECDSA, Blowfish, CAST, MD5, SHA1...). Ces algorithmes sont utilisables directement (en incluant le fichier d'en-tête correspondant) ou via une interface de plus haut niveau, appelée EVP (EnVeloPe). La sous-bibliothèque de l'algorithme (ex. : `./crypto/aes`) offre une vue directe sur l'algorithme, tandis qu'EVP présente l'avantage d'en abstraire les détails pour ne laisser apparaître que ce qui est le plus souvent utile à l'appelant : chiffrer, déchiffrer, etc.

Cette différence est illustrée dans les codes sources n°2 et n°3, où, dans les deux cas, nous ne faisons que chiffrer un texte en clair, puis le déchiffrer. Le code n°2 est clairement de plus bas niveau : l'appelant doit savoir que, avant toute utilisation, la clé AES doit être transformée (routine d' « expansion ». Il doit également prendre garde au fait que l'implémentation du mode CBC travaille (et donc modifie) le vecteur d'initialisation qui lui est fourni, d'où la nécessité de travailler sur une copie du vecteur d'initialisation (`workiv`). Une petite particularité qui n'est documentée nulle part...

```
#include <openssl/aes.h>

#define NB_BLOCKS(len) (((len) + AES_BLOCK_SIZE - 1) / AES_BLOCK_SIZE)

void encrypt_decrypt(unsigned char *key, int keylen, unsigned char *plaintext,
int textlen){
  AES_KEY keyschedule;
  unsigned char iv[AES_BLOCK_SIZE] = { 0x07, 0x01, 0x02, 0x03,
                                       0x08, 0x01, 0x02, 0x03,
                                       0x09, 0x01, 0x02, 0x03,
                                       0x01, 0x01, 0x02, 0x03 };
  unsigned char workiv[AES_BLOCK_SIZE], workbuf[NB_BLOCKS(textlen)*AES_BLOCK_
  SIZE];

  memcpy(workbuf, plaintext, textlen);
  memcpy(workiv, iv, AES_BLOCK_SIZE);

  /* expansion de la clé */
  if (AES_set_encrypt_key(key, keylen * 8, &keyschedule) != 0) { /* erreur */ }

  /* chiffrement AES en mode CBC, attention, workiv est modifié */
  AES_cbc_encrypt(workbuf, workbuf, sizeof(workbuf), &keyschedule, workiv,
  AES_ENCRYPT);

  /* déchiffrement */
  memcpy(workiv, iv, AES_BLOCK_SIZE);
  if (AES_set_decrypt_key(key, keylen * 8, &keyschedule) != 0) { /* erreur */ }
  AES_cbc_encrypt(workbuf, workbuf, sizeof(workbuf), &keyschedule, workiv,
  AES_DECRYPT);
}
```

Code source n°2 effectuant un chiffrement/déchiffrement directement à l'aide de la sous-bibliothèque AES

Au contraire, avec EVP, le code source n°3 permet de s'abstraire des spécificités d'AES et offre un mode d'utilisation semblable pour tout algorithme symétrique. Si, à la place d'AES CBC,

nous souhaitons effectuer un chiffrement/déchiffrement DES ECB, il suffit de remplacer les appels `EVP_aes_128_cbc()` par `EVP_des_ecb()`, ainsi que la taille des blocs AES par ceux de DES.

```
#include <openssl/evp.h>
#include <openssl/aes.h>

#define NB_BLOCKS(len) (((len) + AES_BLOCK_SIZE - 1) / AES_BLOCK_SIZE)

void encrypt_decrypt(unsigned char *key, int keylen, unsigned char *plaintext,
int textlen){
  EVP_CIPHER_CTX ctx;
  unsigned char iv[AES_BLOCK_SIZE] = { 0x07, 0x01, 0x02, 0x03,
                                       0x08, 0x01, 0x02, 0x03,
                                       0x09, 0x01, 0x02, 0x03,
                                       0x01, 0x01, 0x02, 0x03 };
  unsigned char workbuf[NB_BLOCKS(textlen) * AES_BLOCK_SIZE];
  int len;

  EVP_CIPHER_CTX_init(&ctx);

  /* chiffrement AES en mode CBC */
  EVP_EncryptInit(&ctx, EVP_aes_128_cbc(), key, iv);
  EVP_EncryptUpdate(&ctx, workbuf, &len, plaintext, textlen);
  EVP_EncryptFinal(&ctx, &workbuf[len], &len);

  /* déchiffrement */
  EVP_DecryptInit(&ctx,  EVP_aes_128_cbc(), key, iv);
  EVP_DecryptUpdate(&ctx, workbuf, &len, workbuf, sizeof(workbuf));
  EVP_DecryptFinal(&ctx, &workbuf[len], &len);

  EVP_cleanup();
}
```

Code source n°3 effectuant un chiffrement/déchiffrement à l'aide de l'interface d'abstraction EVP

L'utilisation de l'interface EVP se fait de la manière suivante :

1▶ Initialisation d'un contexte (de chiffrement dans l'exemple, mais il existe également des contextes pour les fonctions de hachage), via la fonction `EVP_CIPHER_CTX_init`.

2▶ Initialisation de l'opération de cryptographie par une fonction `EVP_xxxInit` (ex. : `EVP_EncryptInit`, `EVP_DecryptInit`, `EVP_DigestInit`...) à laquelle on fournit un descripteur d'algorithme, une clé et des paramètres. Cette fonction met à jour le contexte utilisé pour la suite de l'opération. Les calculs cryptographiques en eux-mêmes peuvent être effectués en plusieurs parties `EVP_xxxUpdate` et doivent se terminer par un `EVP_xxxFinal`. Pour les lecteurs familiers de PKCS#11, cette façon de procéder est proche de PKCS#11 où l'on trouve des fonctions `C_xxxInit`, `C_xxxUpdate`, `C_xxxFinal`.

3▶ Déclaration de la fin de l'utilisation d'EVP par un appel à `EVP_cleanup`.

Sous-bibliothèques de « haut » niveau

Au-dessus d'EVP, nous trouvons une série de sous-bibliothèques couramment utilisées en sécurité, même si elles ne font plus strictement partie de la cryptographie. Il s'agit notamment de la gestion des certificats X.509, du support des formats PEM, PKCS#7 et PKCS#12 ou encore d'OCSP (*Online Certificate Status Protocol*). Ces sous-bibliothèques offrent généralement une API assez incomplète, peu documentée, et pas forcément très bien architecturée, mais elles ont le mérite d'exister et de faciliter la tâche du programmeur pour les traitements les plus courants.

Par exemple, dans le cas de PKCS#12 :

⇨ Les SafeBags de type `SecretBag` (transport d'une donnée secrète quelconque) ne sont pas supportés, et les CRLBag (transport de listes de révocation) sont implémentées comme des CertBag.

⇨ Les deux seules fonctions documentées sont `PKCS12_create` et `PKCS12_parse` alors que le fichier `pkcs12.h` en exporte plusieurs dizaines. En cherchant bien, dans le répertoire `./doc`, perdues au milieu d'un fichier appelé `openssl.txt`, on trouve quelques autres informations sur la sous-bibliothèque `PKCS#12`.

⇨ `PKCS12_create` et `PKCS12_parse` semblent avoir été conçues « à l'arrache » pour pallier les problèmes les plus courants du programmeur du dimanche, ce qui résulte en un prototype plutôt mal fichu. Par exemple, `PKCS12_create` confond le mot de passe PKCS#12 avec l'éventuel mot de passe PKCS#8 protégeant la clé privée, ne permet pas le choix du sel dans la dérivation du mot de passe, ne permet pas le contrôle du type de SafeBag utilisé, etc.

2

Le code n°4 (page suivante) présente une utilisation avancée de l'API PKCS#12. Son objectif est de construire un fichier PKCS#12 dont l'authenticité est protégée par un MAC, contenant un seul SafeBag chiffré, de type `KeyBag`, lui-même contenant une clé privée (en clair) au format PKCS#8. Cette manipulation n'est pas réalisable avec la fonction `PKCS12_create`. Toutes les fonctions utilisées sont présentes dans le fichier `pkcs12.h` publiquement exporté, pourtant aucune n'est documentée. Pour mettre en valeur les appels importants, le code ne traite pas les cas d'erreurs.

Tout d'abord, il faut mettre la clé privée (`pkey`) au format PKCS#8 (`EVP_PKEY2PKCS8`). Ensuite, cette clé doit être mise dans un SafeBag (conteneur PKCS#12) de type `KeyBag` (`PKCS12_MAKE_KEYBAG`). Un fichier PKCS#12 peut contenir plusieurs de ces SafeBags d'où la nécessité de l'ajouter à une liste (`sk_PKCS12_SAFEBAG_push`). Enfin, nous souhaitons chiffrer l'ensemble de ces SafeBags à l'aide de 3DES-CBC 112 bits. La clé de chiffrement est dérivée du mot de passe fourni (`encpass`) via un certain nombre d'itérations (1024 dans notre cas) de la fonction de hachage `SHA-1`. Cette façon de dériver un mot de passe en une clé secrète est standard et décrite dans PKCS#5. Elle prend également en entrée un sel, mais nous n'entrerons pas plus dans les détails. Cette étape se fait via l'appel à `PKCS12_pack_p7encdata`. Finalement, il faut construire le PKCS#12 en ajoutant notre conteneur chiffré (`PKCS12_pack_authsafes`), et en le protégeant à l'aide d'un MAC (`PKCS12_set_mac`). La clé utilisée pour le MAC est dérivée à partir d'un mot de passe (`macpass`) avec 3 itérations. Ceci n'est qu'un exemple pour illustrer que le mot de passe et le mécanisme chiffrant le conteneur n'ont aucune raison d'être les mêmes que ceux qui protègent l'authenticité du PKCS#12.

```
#include <openssl/evp.h>
#include <openssl/pkcs12.h>

PKCS12 *make_pkcs12(char *encpass, int encpasslen,
                    char *macpass, int macpasslen, EVP_PKEY *pkey) {
  PKCS12 *p12 = NULL;
  PKCS8_PRIV_KEY_INFO *p8 = NULL;
  PKCS12_SAFEBAG *bag = NULL;
  PKCS7 *p7 = NULL;
  STACK_OF(PKCS12_SAFEBAG) *bags = sk_PKCS12_SAFEBAG_new_null();
  STACK_OF(PKCS7) *p7s = sk_PKCS7_new_null();

  p8 = EVP_PKEY2PKCS8(pkey);
  bag = PKCS12_MAKE_KEYBAG(p8);  sk_PKCS12_SAFEBAG_push(bags, bag);
  p7 = PKCS12_pack_p7encdata(NID_pbe_WithSHA1And2_Key_TripleDES_CBC,
                                        encpass,
                                        encpasslen,
                                        NULL, /* salt */
                                        0,
                                        1024, /* iterations */
                                        bags);
  sk_PKCS7_push(p7s, p7);

  p12 = PKCS12_init(NID_pkcs7_data);
  PKCS12_pack_authsafes(p12, p7s);
  PKCS12_set_mac(p12, macpass, macpasslen, NULL /* salt */, 0, 3 /*
terations */, NULL);

  sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
  sk_PKCS7_pop_free(p7s, PKCS7_free);

  return p12;
}
```

Code source n°4 : création avancée d'un fichier PKCS#12.

Une fois que nous disposons d'une structure PKCS12 bien remplie, il est possible de l'écrire dans un fichier en utilisant la fonction `i2d_PKCS12_fp`. Nous pouvons alors visuellement vérifier que le fichier PKCS#12 est correct à l'aide de la commande openSSL :

```
$ openssl pkcs12 -twopass -noout -info -in mypkcs.p12
Enter MAC Password: TAPER LE MDP du MAC
Enter Import Password: TAPER LE MDP de CHIFFREMENT
MAC Iteration 3
MAC verified OK
PKCS7 Encrypted data: pbeWithSHA1And2-KeyTripleDES-CBC, Iteration 1024
Key bag
```

Les autres façons d'utiliser la bibliothèque de cryptographie

Orthogonalement à cette série de couches, deux API assez récentes existent : l'interface STORE et l'interface ENGINE. Comme son nom l'indique, l'interface STORE traite du stockage de données cryptographiques. Imaginons que nous avons généré une paire de clés : comment la conserver ? C'est de cela que STORE se préoccupe. L'interface STORE permet de stocker des objets (clés, certificats, données quelconques, etc.) sur un support donné. Pour cela, le développeur est libre d'implémenter une `STORE_METHOD` à sa convenance, dont les fonctions de base sont :

- `init`, `clean` : fonctions à appeler pour initialiser et terminer le stockage sur un support donné ;
- `generate_object` : création d'un objet directement sur le support ;
- `get_object`, `store_object`, `modify_object`, `revoke_object`, `delete_object` : gestion des objets ;
- `list_object_xxx` (`start`, `next`, `end`...) : pour lister les objets stockés ;
- `update_store` ;
- `lock_store`, `unlock_store`.

Cependant, à l'heure actuelle (version 0.9.8d), seule l'*interface* STORE est définie et il n'y a aucune implémentation par défaut vraiment utilisable. Une implémentation stockant les objets en mémoire vive a été initiée (`str_mem.c`), mais la plupart des fonctions ne sont pas encore implémentées. Il faut donc attendre un peu ou faire sa propre implémentation.
L'interface ENGINE (« moteur ») place la bibliothèque de cryptographie sous l'angle des modules cryptographiques. Un moteur est le regroupement d'un certain nombre d'implémentations cryptographiques. Cette notion peut être rapprochée de celle de « *Crypto Provider* » Java, de CSP (*Crypto Service Provider*) chez Microsoft ou de *token* cryptographique PKCS#11. OpenSSL est livré avec le code source de plusieurs moteurs (répertoire `./engines`), comme celui du coprocesseur IBM 4758 ou des interfaces CHIL (*Cryptographic Hardware Interface Library*) du matériel nCipher. Un autre moteur intéressant à signaler est celui d'OpenSC **[OpenSC]** : il permet de faire le lien avec les cartes à puce offrant une interface classique PKCS#11.

Suivant ses propres capacités, un moteur peut offrir l'accès à :

- toute structure cryptographique `xxx_METHOD` ;
- des algorithmes de chiffrement symétriques (via `EVP_CIPHER`) ;
- des fonctions de hachage (via `EVP_DIGEST`) ;
- chargement de clé publique ou privée.

Le fichier `engine.h` contient à la fois l'interface *d'utilisation* d'un moteur (comment charger, utiliser un moteur existant) et l'interface de *développement* d'un nouveau moteur, ce qui complexifie un peu sa lecture. Au final, l'utilisation du moteur reste relativement simple :

1. Charger en premier lieu le moteur désiré. À titre d'exemple, nous chargeons ci-dessous le moteur qui correspond aux implémentations par défaut d'OpenSSL (`ENGINE_load_openssl`).
2. Ensuite, récupérer un pointeur (`ENGINE *`) vers le moteur correspondant, qui permettra d'accéder à toutes ses fonctionnalités. N'ayant qu'un seul moteur de chargé, `ENGINE_get_first()` fonctionne dans notre cas.
3. Initialiser le moteur (`ENGINE_init`) – c'est important notamment pour certains accélérateurs cryptographiques.
4. Récupérer le mécanisme et l'algorithme désiré. L'exemple ci-dessous récupère la fonction de hachage SHA1 et effectue le condensé d'un texte.
5. Terminer l'utilisation du moteur en signalant que l'on ne veut plus l'utiliser (`ENGINE_finish`), puis en libérant ce qui a été alloué (`ENGINE_free`, `ENGINE_cleanup`).

```
#include <openssl/evp.h>
#include <openssl/engine.h>

void engine_digest_sample(unsigned char *text, int len) {
  ENGINE *e = NULL;
  EVP_MD *digest = NULL;
  EVP_MD_CTX ctx;
  unsigned char result[20];
  int resultlen;

  ENGINE_load_openssl();
  e = ENGINE_get_first();
  ENGINE_init(e);
  digest = ENGINE_get_digest(e, NID_sha1);

  EVP_MD_CTX_init(&ctx);
  EVP_DigestInit(&ctx, digest);  EVP_DigestUpdate(&ctx, plaintext, len);
  EVP_DigestFinal(&ctx, result, &resultlen);
  EVP_MD_CTX_cleanup(&ctx);
  EVP_cleanup();

  ENGINE_finish(e);
  ENGINE_free(e);
  ENGINE_cleanup();
}
```

Code d'exemple n°5 utilisant un moteur cryptographique

Pour implémenter un nouveau moteur, le développeur doit :

⇨ Donner un nom et un identifiant à son moteur en appelant `ENGINE_set_id` et `ENGINE_set_name`.

⇨ Renseigner toutes les structures `xxx_METHOD` qu'il souhaite que le moteur fournisse. Il est tout à fait possible de n'implémenter que certaines fonctions (mettre `NULL` pour le reste). Les fonctionnalités offertes doivent être déclarées via l'appel à des fonctions comme `ENGINE_set_RAND` (déclaration d'une structure `RAND_METHOD`), `ENGINE_set_init_function`, `ENGINE_load_privkey_function`...

⇨ Pour les moteurs qui supportent en plus leur propre jeu de commandes, fournir un tableau de `ENGINE_CMD_DEFN` – une commande par entrée (terminer par une entrée nulle).

⇨ Faire attention aux cas où RSA, DH, DSA (etc.) ne sont pas présents (flags de compilation `OPENSSL_NO_xxx`).

⇨ Fournir une fonction `ENGINE_load_nomDuMoteur()` permettant le chargement statique du moteur. Si le chargement dynamique du moteur est supporté, il faut déclarer une fonction `bind_fn` via la macro `IMPLEMENT_DYNAMIC_BIND_FN`.

Cela donne le pseudo-code suivant :

```
#include <openssl/engine.h>

#ifndef OPENSSL_NO_RSA
static RSA_METHOD myengine_rsa = {
  "RSA method pour mon moteur",
  myengine_rsa_pub_enc,
  NULL,
  ...
};
#endif /* OPENSSL_NO_RSA */
static const ENGINE_CMD_DEFN myengine_cmd_defns[] = {
  { MYENGINE_CMD1, "CMD1", "Commande n°1", ENGINE_CMD_FLAG_STRING },
  {0, NULL, NULL, 0}
};

static const char *myengine_id = "myengine";
static const char *myengine_name = "Mon moteur à moi";

static int bind_helper(ENGINE *e) {
  if(!ENGINE_set_id(e, myengine_id)
     || !ENGINE_set_name(e, myengine_name)
     ||      !ENGINE_set_cmd_defns(e, myengine_cmd_defns)
#ifndef OPENSSL_NO_RSA
     || !ENGINE_set_RSA(e, RSA_get_default_method())
#endif  /* OPENSSL_NO_RSA */

    )
    return 0;
  return 1;
}

void ENGINE_load_myengine(void) {
  ENGINE *e = ENGINE_new();
  if (e == NULL) return;
  if (! bind_helper(e)) {
    ENGINE_free(e);
  }
  ENGINE_add(e);
  ENGINE_free(e);
}

#ifndef OPENSSL_NO_DYNAMIC_ENGINE
static int bind_fn(ENGINE *e, const char *id)
{
  if (id && (strcmp(id, myengine_id) != 0))
            return 0;
  if (!bind_helper(e))
            return 0;
  return 1;
}
IMPLEMENT_DYNAMIC_CHECK_FN()
IMPLEMENT_DYNAMIC_BIND_FN(bind_fn)
#endif /* OPENSSL_NO_DYNAMIC_ENGINE */
```

Pseudo-code n°6 d'implémentation d'un nouveau moteur cryptographique OpenSSL

Les « utilitaires »

Autour de toutes ces sous-bibliothèques, gravitent quelques utilitaires (*stack*, *threads*...) dont trois particulièrement importants :

⇨ BIO (*Binary Input Output*) : gestion des entrées/sorties binaires, notamment affichage vers `stdout` ou redirection des flux vers un fichier, une *socket*... Sans entrer dans les détails, pour afficher des données, il faut créer un BIO (`BIO_new`), lui affecter un flux (ici, `BIO_set_fp`) et, à la fin, libérer la structure (`BIO_free`). Des fonctions plus avancées permettent de lire/écrire des flux encodés en Base64 (`BIO_f_base64`) ou l'affichage de grands entiers (`BN_print`), etc.

```
#include <openssl/bio.h>

BIO *bp;
bp = BIO_new(BIO_s_file());
BIO_set_fp(bp, stdout, BIO_NOCLOSE);
BIO_puts(bp, "Un exemple\n");
...
BIO_free(bp);
```

⇨ ERR : traitement des messages d'erreurs qui peuvent survenir dans les diverses sous-bibliothèques de cryptographie. Chaque sous-bibliothèque définit un ensemble de textes descriptifs pour ses fonctions (ex. : « `DH_new_method` ») et de raisons d'erreur (ex. : « `bad generator` » – mauvais générateur). Lorsque nous tombons dans un cas d'erreur, la sous-bibliothèque appelle une macro qui ajoute une erreur (gérée par ERR) en insérant automatiquement le nom de la sous-bibliothèque (ex. : `RAND`), le nom de la fonction, la ligne où s'est produite l'erreur, la description de la fonction et la raison de l'erreur. Par exemple, `Dherr(DH_F_DH_BUILTIN_GENPARAMS, DH_R_BAD_GENERATOR)` va définir une erreur pour la sous-bibliothèque Diffie-Hellman où la valeur du générateur est jugée illicite. Pour l'appelant, l'affichage d'erreur se fait tout simplement par l'appel de `ERR_print_errors_fp(stderr)`. Cependant, afin de ne pas surcharger inutilement les bibliothèques, les textes descriptifs d'erreurs ne sont pas automatiquement chargés. On obtiendra donc un message d'erreur du style suivant :

```
6578:error:06074079:lib(6):func(116):reason(121):evp_pbe.c:89:TYPE=pbeWithSHA1And2-
KeyTripleDES-CBC
```

Pour rendre le message plus parlant, il faut charger les textes au préalable par l'appel de `ERR_load_crypto_strings()`. On obtient alors :

```
6578:error:06074079:digital envelope routines:EVP_PBE_CipherInit:unknown pbe
algorithm:evp_pbe.c:89:TYPE=pbeWithSHA1And2-KeyTripleDES-CBC
```

Cette erreur fournit le thread id qui pose problème (6578), l'identifiant de l'erreur (06074079 – cet identifiant est constitué du numéro de sous-bibliothèque, de la fonction et de la raison affichés par la suite), le nom de la sous-bibliothèque « `digital envelope routines` » désigne la sous-bibliothèque EVP (identifiant n°6), le nom de la fonction concernée (`EVP_PBE_CipherInit`), la raison de l'erreur : ici, on nous signale qu'on tente d'utiliser un algorithme inconnu de dérivation de mot de passe, le fichier et la ligne où l'erreur s'est produite et des données complémentaires : ici, on explique que c'est l'algorithme `pbeWithSHA1And2-KeyTripleDES-CBC` qui est inconnu. Cet algorithme étant valide, c'est sans doute que l'appelant a oublié de charger cet algorithme à l'initialisation d'OpenSSL (voir chargement dynamique plus bas).

⇨ `ASN.1` : il s'agit d'une bibliothèque très proche de l'`ASN.1` qui vise à faciliter l'encodage et le décodage en `DER`. Les habitués de l'`ASN.1` ne devraient pas avoir trop de mal à l'utiliser, les autres devront cependant d'abord se familiariser avec l'`ASN.1` et le `DER`. L'encodage et le décodage d'un type `ASN.1` se réalisent de la manière suivante :

1▶ Déclarer le type à encoder/décoder : chaque type `ASN.1` doit être associé à une structure C, dont chaque champ est un pointeur sur un type `ASN.1`. Dans l'exemple ci-dessous, nous définissons un type bien connu, appelé « `DigestInfo` », qui consiste en une fonction de hachage (et ses paramètres éventuels) et la valeur du haché. Des précisions sur comment effectuer le principe de l'encodage sont apportées par diverses macros comme `ASN1_OPT` (champ optionnel), `ASN1_IMP` (encodage implicite), `ASN1_EXP` (encodage explicite).

2▶ Implémenter les fonctions d'allocation/libération/encodage et décodage du type. Ceci se fait *automatiquement* via la macro `IMPLEMENT_ASN1_FUNCTIONS` : elle écrit pour vous l'implémentation de ces 4 fonctions : `NomDuType_new()`, `NomDuType_free()`, `i2d_NomDuType()`, `d2i_NomDuType()`.

3▶ Renseigner les champs du type à encoder. Dans l'exemple qui suit, la fonction de hachage utilisée est SHA-1 (`NID_sha1`), et nous affectons le haché correspondant (`digest`). La fonction de hachage `SHA-1` n'ayant aucun paramètre, nous attribuons au champ `parameters` la valeur correspondant au type `ASN.1 NULL`. La présence même de paramètres étant optionnelle (`parameters` est déclaré `ASN1_OPT`), il faut noter qu'il est impératif d'allouer le pointeur `dinfo->digestAlgorithm->parameters` (`ASN1_TYPE_new()`) avant de lui affecter une valeur. Une fois les valeurs renseignées, l'encodage DER se fait par l'appel à la fonction `i2d_DigestInfo` qui a automatiquement été implémentée par la macro `IMPLEMENT_ASN1_FUNCTIONS(DigestInfo)`. Le résultat est une suite d'octets. Enfin, nous pouvons libérer en profondeur la structure `dinfo` par l'appel à `DigestInfo_free`.

4▶ Le décodage DER se fait par l'appel à la fonction `d2i_DigestInfo`, qui renseigne la variable `dinfo`. Ensuite, nous récupérons chaque champ de cette structure, suivant les spécificités de son type. La fonction `OBJ_obj2nid` convertit un OID en un NID d'OpenSSL – ce dernier pouvant être affiché textuellement via `OBJ_nid2ln()`. Les champs de type `ASN.1 OCTET STRING` sont directement accessibles via un pointeur sur leurs données (`->data`) et leur longueur (`->length`). Les champs de type `ASN.1 INTEGER` sont accessibles via la macro `ASN1_INTEGER_get`, etc.

```
#include <stdio.h>
#include <openssl/objects.h>
#include <openssl/asn1t.h>

typedef struct {
  ASN1_OBJECT *id;
  ASN1_TYPE *parameters;
} AlgorithmIdentifier;

typedef struct {
  AlgorithmIdentifier *digestAlgorithm;
  ASN1_OCTET_STRING *digest;
} DigestInfo;

ASN1_SEQUENCE(AlgorithmIdentifier) = {
  ASN1_SIMPLE(AlgorithmIdentifier, id, ASN1_OBJECT),
  ASN1_OPT(AlgorithmIdentifier, parameters, ASN1_ANY)
  } ASN1_SEQUENCE_END(AlgorithmIdentifier)
IMPLEMENT_ASN1_FUNCTIONS(AlgorithmIdentifier)

ASN1_SEQUENCE(DigestInfo) = {
  ASN1_SIMPLE(DigestInfo, digestAlgorithm, AlgorithmIdentifier),
  ASN1_SIMPLE(DigestInfo, digest, ASN1_OCTET_STRING)
  } ASN1_SEQUENCE_END(DigestInfo)
IMPLEMENT_ASN1_FUNCTIONS(DigestInfo)

int encode_digestinfo(unsigned char **der) {
  DigestInfo *dinfo = DigestInfo_new();
  unsigned char digest[20] = { 0x01, 0x02, ... }; // la valeur du ache
  int len;

  dinfo->digestAlgorithm->id = OBJ_nid2obj(NID_sha1);
  dinfo->digestAlgorithm->parameters = ASN1_TYPE_new();
  ASN1_TYPE_set(dinfo->digestAlgorithm->parameters, V_ASN1_NULL,
ULL);
  ASN1_OCTET_STRING_set(dinfo->digest, digest, sizeof(digest));

  len = i2d_DigestInfo(dinfo, der);
```

```
  DigestInfo_free(dinfo);
  return len;}

void decode_digestinfo(const unsigned char *der, int len) {
  DigestInfo *dinfo;
  int nid;

  dinfo = d2i_DigestInfo(NULL, &der, (long)len);

  nid = OBJ_obj2nid(dinfo->digestAlgorithm->id);
  printf("--> OID: (nid=%d) - %s\n", nid, OBJ_nid2ln(nid));

  if (dinfo->digestAlgorithm->parameters) {
    switch(ASN1_TYPE_get(dinfo->digestAlgorithm->parameters)) {
    case V_ASN1_NULL:
      printf("--> NULL parameters\n");
      break;
    }
  }

  printf("--> Digest: \n");
  dump_buffer(dinfo->digest->data, dinfo->digest->length);

  DigestInfo_free(dinfo);
}
```

Code d'exemple n°7 pour l'encodage/décodage DER d'une structure DigestInfo.

Compilation et link de la bibliothèque de cryptographie

Il reste un dernier point à éclaircir pour pouvoir effectivement utiliser la bibliothèque de cryptographie d'OpenSSL : la compilation et le *link* d'un programme l'utilisant.

Compilation des programmes

Pour compiler avec succès du code appelant la bibliothèque de cryptographie, il faut inclure le fichier en-tête qui correspond à la sous-bibliothèque. Tous ces fichiers en-tête sont recopiés dans le répertoire `./include/openssl`. Nous pouvons donc compiler via une commande du style :

```
gcc -I$(OPENSSLDIR)/include ...
```

et inclure dans le fichier source une directive `#include <openssl/xxxx>`. Par exemple, les codes sources vus plus haut incluent `openssl/evp.h`, `openssl/aes.h`, etc.

Paramétrages de compilation de la bibliothèque de cryptographie

Par ailleurs, le code source de la bibliothèque crypto supporte trois types de flags de compilation :

⇨ Les flags d'activation/désactivation de sous-bibliothèques : `OPENSSL_NO_xxx`. Lorsque ce flag est positionné, la sous-bibliothèque correspondante n'est pas compilée. Par exemple, `OPENSSL_NO_IDEA` ne compile pas l'algorithme IDEA (qui est d'ailleurs breveté), `OPENSSL_NO_ENGINE` désactive l'utilisation de moteurs cryptographiques (voir sous-bibliothèque `ENGINE`) etc. Par défaut, les algorithmes Camellia, MDC2 et RC5 ne sont pas compilés.

⇨ Les flags d'utilisation (ou non) de routines en assembleur sont activés par des flags `xxx_ASM`. En effet, le code source de certains algorithmes contient des portions d'assembleur qui peuvent ou pas être utilisées. Si ce flag est positionné, ces routines sont utilisées, sinon, on n'utilise que du code C. Par défaut, les algorithmes SHA1, MD5, RIPEMD-160, AES et la multiplication de grands entiers (sous-bibliothèque BN) contiennent des portions d'assembleur.

⇨ Les flags de type de plateformes : normalement, ceux-ci sont automatiquement correctement positionnés par `./configure`, on n'a pas à s'en soucier. Ils précisent l'*endianness* de la machine (Big Endian avec `B_ENDIAN`, Little Endian avec `L_ENDIAN`), le nombre de bits (`SIXTY_FOUR_BIT_LONG` pour les systèmes 64 bits).

Ces options de compilation peuvent être précisées, soit directement dans le code (pour les hackers) en utilisant le bon flag, soit au moment de la configuration d'OpenSSL. Par exemple :

```
./config no-asm no-mdc2
```

compilera une bibliothèque de cryptographie sans utilisation d'assembleur et sans l'algorithme MDC2.

Voir tableau page suivante.

Link

Toutes les fonctions de la bibliothèque de cryptographie sont groupées dans la bibliothèque `libcrypto.a`. Tout programme l'utilisant doit donc être linké avec cette bibliothèque :

```
gcc xxx.o -lcrypto -L$(OPENSSLDIR) -o xxx
```

De plus, il est parfois nécessaire de linker le programme avec la bibliothèque de chargement dynamique dl (`-ldl`). En effet, nous avons vu plus haut que l'option de compilation `OPENSSL_NO_xxx` désactive complètement la compilation d'un algorithme. Par exemple, si `OPENSSL_NO_RSA` est positionné, le développeur n'a plus le droit d'inclure `openssl/rsa.h` dans son code (erreur de compilation). Ceci est utile lorsque nous sommes certains de ne pas utiliser certains algorithmes et que nous recherchons à minimiser la taille de la bibliothèque de cryptographie résultante pas tel algorithme, et qu'on est soucieux de la taille de la bibliothèque de cryptographie résultante.

Il existe cependant une façon moins « brutale » de désactiver un algorithme lorsque la sous-bibliothèque EVP est utilisée. Au lieu d'accéder directement à un algorithme, il est possible de le *charger dynamiquement* – ce qui requiert la bibliothèque `dl`. Dans un premier temps, il faut alors indiquer quels algorithmes nous souhaitons charger. Soit nous utilisons les fonctions `OpenSSL_add_all_ciphers()` et `OpenSSL_add_digests()` qui chargent respectivement tous les algorithmes de chiffrement et fonctions de hachage, soit nous précisons « à la main » les algorithmes désirés : `EVP_add_cipher(EVP_algorithmeXXX())`, `EVP_add_digest(EVP_algorithme())`.

Par exemple, le code source n°4 sur PKCS#12 ne fonctionne correctement que si les algorithmes utilisés ont été chargés, c'est-à-dire :

Flag de compilation	Flag pour ./config	Description
OPENSSL_NO_IDEA, OPENSSL_NO_RC5, OPENSSL_NO_RSA...	no-idea, no-rc5, no-rsa...	Désactive la compilation d'un algorithme particulier, ici IDEA, RC5 et RSA
OPENSSL_NO_ENGINE	-D...	Désactive l'utilisation de moteurs cryptographiques
OPENSSL_NO_HW	-D...	Désactive tous les moteurs matériels (ex. : IBM 4758)
OPENSSL_NO_HW_4758_CCA, OPENSSL_NO_HW_ATALLA...	-D...	Désactive un moteur matériel particulier, ici IBM 4758 et Atalla.
OPENSSL_NO_ERR	-D...	Désactive le chargement des chaînes d'erreurs
OPENSSL_NO_ASM	no-asm	Interdit toute utilisation de routines assembleur
OPENSSL_NO_INLINE_ASM	-D...	Interdit l'utilisation de routines assembleur écrites « en ligne » sous la forme asm ou __asm__(« directives... »)
AES_ASM, SHA1_ASM...	-D...	Active l'utilisation de routines écrites en assembleur pour AES, SHA1...

TI *Tableau des options de compilation les plus courantes*

```
#include <openssl/evp.h>
#include <openssl/pkcs12.h>

PKCS12 *p12 = NULL;
EVP_PKEY *pkey = NULL;
...

EVP_PBE_alg_add(NID_pbe_WithSHA1And2_Key_TripleDES_CBC, EVP_des_ede_cbc(),
VP_sha1(), PKCS12_PBE_keyivgen);
EVP_add_digest(EVP_sha1());
...
p12 = make_pkcs12(pwd, sizeof(pwd), "tata", 4, pkey);
...
EVP_cleanup();
```

Chargement dynamique des algorithmes nécessaires pour le code exemple n°4

Conclusion

Loin de pallier un manque de documentation flagrant d'OpenSSL, cet article aura, je l'espère, éclaircit l'organisation et l'utilisation de quelques portions de la bibliothèque de cryptographie d'OpenSSL. Et si cela vous donnait l'envie de participer à la tâche (titanesque) de documentation, n'hésitez pas ;-)

Références

[IBM] *Secure programming with the OpenSSL API : three parts*

http://www-128.ibm.com/developerworks/linux/library/l-openssl.html

[OpenSC] OpenSC, Engine PKCS#11

http://www.opensc-project.org/engine_pkcs11/

[PKCS] RSA Laboratories, *Public Key Cryptography Standards* (n°1 to 15).

http://www.rsasecurity.com/rsalabs/node.asp?id=2124

[Res01] RESCORLA (Eric), « *An introduction to OpenSSL programming* », *Linux Journal*, 2001.

http://www.linuxjournal.com/article/4822

[Vin03] HEGDE (Vinayak), « *Encryption using OpenSSL's crypto libraries* », *Linux Gazette issue* 87, février 2003.

http://tldp.org/LDP/LG/issue87/vinayak.html

[VM03] VIEGA (John), MESSIER (Matt), *Secure Programming Cookbook for C and C++*, O'Reilly, juillet 2003.

Nfnetlink et la manipulation du suivi de connexions de Netfilter

Netfilter, la couche pare-feu de Linux, a connu une évolution majeure rendue publique dans le noyau 2.6.14. Il s'agit de l'infrastructure Nfnetlink qui est une refonte complète du système de communication entre le noyau et l'espace utilisateur. Elle accroît notamment les possibilités d'interrogations et de manipulations du suivi de connexions et ouvre la porte à de nouvelles applications.

mots clés : *Linux / Netfilter / pare-feu / administration / sécurité*

Netfilter et Nfnetlink

Netfilter est l'infrastructure de filtrage de paquets de Linux depuis la version 2.4.0. La principale nouveauté apportée par rapport à IPchains, son prédécesseur, est le suivi d'état appelé `conntrack`.

Conntrack

Historique et présentation

Les premiers filtres de paquets se contentaient d'autoriser ou de bloquer les paquets en étudiant leur contenu indépendamment les uns des autres. Ils réalisaient donc principalement des vérifications sur les en-têtes IP et protocolaire des paquets pour déterminer leur sort, opération très rapide et performante, mais qui pose des problèmes en termes de sécurité.

Prenons le cas d'un serveur Web protégé derrière un pare-feu. Si l'on veut se limiter à des filtres sur les en-têtes, il faut, pour l'autoriser à servir des pages webs :

⇨ autoriser les paquets à destination du port 80 du serveur venant d'internet ;

⇨ autoriser les paquets venant du port 80 du serveur à destination d'internet.

Si l'on exprime cela avec `iptables`, l'outil de génération de règles de Netfilter, on obtient :

```
iptables -A FORWARD -s 0.0.0.0/0 -d ${WEBSERVER} -p tcp --sport 1024: --dport 80 -j ACCEPT
iptables -A FORWARD -d 0.0.0.0/0 -s ${WEBSERVER} -p tcp --dport 1024: --sport 80 -j ACCEPT
```

Deux règles sont donc nécessaires, ce qui est déjà assez contraignant pour autoriser un simple flux. De plus, cela ne prend pas en compte l'enchaînement des paquets tel qu'il doit se produire :

1▶ Le client envoie un paquet (SYN) à destination du port 80.

2▶ Le serveur envoie un paquet (SYN ACK) venant du port 80.

Dans le cas d'un pare-feu « à l'ancienne », rien n'empêche de faire 2 avant 1 et donc d'initier une connexion depuis l'intérieur vers l'extérieur :

1▶ Le « serveur » envoie un paquet SYN venant du port 80.

2▶ Le « client » répond avec un SYN ACK.

Pour résoudre cette problématique, la meilleure solution consiste à introduire la notion de « session » dans le système de filtrage. Il conserve la trace des enchaînements des paquets et se sert de ces indications pour laisser ou non passer un paquet. De manière schématique, le conntrack tient à jour une table contenant les sessions en cours et fournit un filtre sur les états relatifs à un paquet à `iptables` :

⇨ `NEW` : le paquet n'appartient pas à une connexion en cours.

⇨ `ESTABLISHED` : le paquet appartient à une connexion en cours.

En termes de règles de filtrage, cela s'exprime comme suit :

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
iptables -A FORWARD -s 0.0.0.0/0 -d ${WEBSERVER} -p tcp --sport 1024: \\
--dport 80 -m state --state NEW -j ACCEPT
```

La deuxième règle accepte donc les paquets qui sont des initialisations de connexions à destination du serveur Web, comme le paquet SYN envoyé par notre surfeur anonyme. La réponse du serveur, le SYN ACK, appartient à une connexion établie et est donc acceptée par la première règle.

Les esprits grincheux objecteront sûrement que si la sécurité est augmentée par la prise en compte du sens des échanges, deux règles `iptables` sont encore nécessaires. C'est effectivement vrai, mais la première règle est universelle et permet de laisser passer toutes les connexions établies. Le nombre de règles nécessaire pour autoriser N flux est donc N+1 au lieu de 2N.

De plus, la première règle est la plus utilisée ; des chiffres récupérés sur un pare-feu en activité montrent que sur un échantillon de 73 millions de paquets, 99,61% sont pris en charge par celle-ci. Ceci permet donc de limiter le nombre de règles parcourues avant la décision.

L'implémentation se fait au moyen d'une structure assimilable à une table de *hash* ce qui garantit une recherche en temps fixe d'un paquet dans la table. Le coût en termes de performances reste cependant important comme le montre l'étude réalisée par Joseph Kadleszic (voir [2]). Le passage du mode routage pur au mode routage+`conntrack` divise en effet le nombre de paquets traités en une seconde par 3.

Fonctionnement du conntrack

Regardons maintenant de plus près le fonctionnement du suivi de connexions. Son emplacement dans le flot de traitement d'un paquet dans Netfitler est illustré par la figure suivante :

Éric Leblond
regit@inl.fr
Développeur principal de NuFW et cofondateur d'INL

encadré 1

Suivi de connexions TCP

Le cas de TCP est le plus intéressant puisqu'il s'agit d'une connexion orientée. Le protocole UDP est en effet régi simplement par la gestion de *timeout* suite à un premier paquet. Pour TCP, le cas est plus complexe. Le suivi de connexions TCP est géré dans le fichier `nf_conntrack_proto_tcp.c`.

La partie la plus difficile consiste à gérer les transitions. L'implémentation de Linux référence 10 états possibles pour une connexion qui peuvent être croisés avec les 6 états de paquets (syn, synack, fin, ack, rst, none), soit 60 transitions. Il faut également prendre en compte le sens des paquets dans la transition. Ainsi un SYN envoyé par le serveur après un SYN venant du serveur n'a aucun sens alors qu'un deuxième SYN venant du client est un simple cas de réémission. Cette duplication nous amène donc à 120 transitions.

Un des points méconnus est la définition de nouvelle connexion. Le terme « connexion » a en effet un sens différent suivant le contexte. Dans celui du protocole IP, tout est clair, car une connexion TCP s'ouvre avec un paquet SYN. Ce n'est pas vrai pour le suivi d'une connexion TCP par le conntrack, car Netfilter est paramétré pour récupérer les connexions existantes. Dans le cas du chargement du module de suivi de connexions ou encore dans le cas de la substitution d'un pare-feu par un autre, le pare-feu en activité voit passer des paquets de connexions existantes et déjà établies. Ces paquets sont vraisemblablement légitimes et des échanges de paquets réussis entre les deux extrémités de la connexion prouvent que la session TCP est active et bien réelle. Elle peut donc être autorisée à traverser le pare-feu. Par défaut, Netfilter, au bout de 3 échanges de paquets sur une connexion inconnue, considère que cette dernière est une connexion établie. Ce comportement est paramétrable grâce à l'entrée de `/proc`, `nf_conntrack_tcp_loose`, qui contient le nombre d'échanges nécessaire à la récupération d'une connexion. Le défaut est à 3 et il faut mettre la variable à 0 pour désactiver la fonctionnalité.

Netfilter propose donc un moyen de récupération à la volée de certaines connexions que l'on peut désactiver en écrivant dans `/proc`. Cela n'est pas sans compromis sur la sécurité d'un pare-feu, puisqu'il faut prendre en compte ce paramètre au moment de la rédaction du jeu de règles. Il est d'ailleurs dans la plupart des cas plus prudent de limiter les ouvertures TCP en ajoutant la vérification du bit SYN (en ajoutant `--syn` à la commande `iptables`).

Pour assurer le maximum de sécurité et lutter contre les attaques par injection de paquets, il est nécessaire de surveiller la fenêtre TCP des paquets. Cette fonctionnalité s'inspire du document *Real Stateful TCP Packet Filtering in IP Filter* de Guido van Rooij [6]. Un paquet est ainsi valide s'il est dans les bornes de l'intervalle défini par la fenêtre TCP. Les paquets de données doivent être dans la fenêtre TCP et les receveurs doivent envoyer un ACK sur des segments situés à droite de la fenêtre TCP. Si on définit :

```
td_maxend = max(sack + max(win,1)) vu dans les paquets de réponse
td_maxwin = max(max(win, 1)) + (sack - ack) vu dans les paquets envoyés
td_maxwin += seq + len - sender.td_maxend
                        si seq + len > sender.td_maxend
td_end    = max(seq + len) vu dans les paquets envoyés
```

un paquet de données sera valide si :

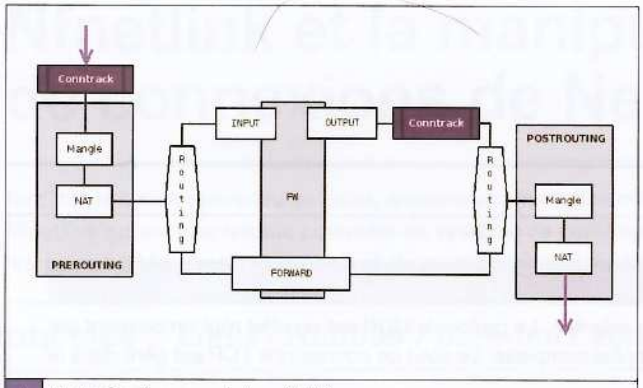
⇨ borne haute : `seq <= sender.td_maxend` ;

⇨ borne basse : `seq + len >= sender.td_end - receiver.td_maxwin`.

Pour un paquet ack, c'est :

⇨ `sack <= receiver.td_end`

⇨ `ack >= receiver.td_end - MAXACKWINDOW`



Itinéraire d'un paquet dans Netfilter

Le traitement subi par un paquet au moment de son passage dans les fonctions de `conntrack` est le suivant :

⇨ Détermination du protocole IP mis en œuvre dans le paquet.

⇨ Suivant le protocole, prise en compte des paramètres pour trouver une entrée déjà existante :

↳ Si c'est le cas, deux possibilités :

↳ Si la connexion est établie, alors on remet au maximum le délai d'expiration de la connexion.

↳ Si la connexion n'était pas encore établie, elle passe établie.

↳ Si ce n'est pas le cas, création d'une entrée classée `NEW` pour le paquet.

Dans un monde normal, cette description serait complète. Il faut cependant ajouter une notion : la « traduction d'adresse ». Un paquet peut subir des transformations d'adresse et/ou de port entre son arrivée et son départ du pare-feu. Il s'ensuit un doublement des paramètres IP relatifs à la connexion et il convient donc de moduler la vérification suivant le sens d'arrivée du paquet. Par conséquent, il faut conserver dans le suivi de connexions une paire de coordonnées IP. Une entrée dans le suivi de connexions peut donc s'afficher sous cette forme :

```
tcp      6 62 TIME_WAIT src=192.168.33.2 dst=3.6.3.9 sport=4867 dport=80
         packets=10 bytes=670 src=3.6.3.9 dst=15.12.4.5 sport=80 dport=4867
         packets=10 bytes=9633 [ASSURED] mark=0 use=1
```

Suivi de connexions applicatives

Dans un monde normal, disais-je, cette description serait complète. C'était sans compter sur le vice des développeurs de protocoles applicatifs. L'origine du mal vient sans doute de l'antique FTP. Ses concepteurs ont en effet décidé qu'il pouvait être intéressant d'effectuer les échanges de données sur des connexions séparées. Ceci permet, par exemple, le transfert site à site (FXP).

Le problème au niveau du filtrage est alors loin d'être trivial. En effet, si l'on veut filtrer ce genre de protocoles de manière efficace, il faut être capable de savoir quelles sont les connexions parallèles qui vont être ouvertes. Si nous n'y parvenons pas, il nous faudra, pour supporter le mode passif de FTP, ouvrir la moitié des ports possibles par une règle autorisant les connexions des ports hauts du client vers les ports hauts du serveur :

```
iptables -A FORWARD -s ${IP_CLIENTS} -d ${IP_SERVEUR} \
         -p tcp --sport 1024: --dport 1024:  -j ACCEPT
```

Plusieurs écoles s'opposent pour résoudre ce problème. Les développeurs d'OpenBSD ont choisi de déléguer ce genre de choses à un proxy de manière à éviter tout problème de surcharge du code noyau. Ceux de Netfilter ont fait un choix bien différent.

Les indications d'ouverture de connexions parallèles sont forcément placées dans les paquets. Lorsqu'un protocole est connu, l'étude au niveau du pare-feu des motifs qui transitent conduit à la détection des annonces des connexions parallèles. Celles-ci sont des connexions attendues (*expects*) : pendant un certain laps de temps, il est probable qu'une des deux parties connectées tente d'établir la connexion annoncée. L'algorithme traduisant ce raisonnement est mis en œuvre dans Netfilter : lorsque que l'un des modules d'analyse protocolaire et de suivi de connexions applicatives détecte la probable ouverture d'une connexion parallèle, il crée une entrée provisoire, un EXPECT. Si un paquet correspondant à l'entrée provisoire est capté par le pare-feu, le conntrack l'assimile à un paquet relatif à une connexion existante et il crée une entrée nouvelle dans la table de connexion. Le conntrack met de plus à disposition d'`iptables` un moyen de filtrer ces connexions relatives. Ainsi, on peut autoriser les paquets relatifs à une connexion et les paquets des connexions établies en remplaçant la première règle par la commande suivante :

```
iptables -R FORWARD 1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

On constate ainsi que le conntrack comporte en fait deux tables :

⇨ la table des connexions ;

⇨ la table des attentes.

Cette deuxième table est longtemps restée cachée au commun des mortels pour la simple raison que rien ne permettait de la voir, et encore moins de la modifier, depuis l'espace utilisateur.

La vision du conntrack depuis l'espace utilisateur était complètement minimaliste. Elle se limitait au fichier `/proc/net/ip_conntrack` qui fournit un moyen de lister les entrées de la table des connexions. Cette méthode comportait certains inconvénients, comme celui de bloquer les modifications du conntrack pendant la consultation du fichier et donc le routage des paquets. Un DOS était donc possible avec un simple :

```
while true; do cat /proc/net/ip_conntrack; done
```

Ce qui est d'autant plus amusant quand on sait que, jusqu'au noyau 2.6.12, l'entrée était lisible par n'importe quel utilisateur.

Nfnetlink

Les insuffisances des interactions entre Netfilter et l'espace utilisateur ont décidé les développeurs de Netfilter (Harald Welte en tête) à mettre en place un nouveau système offrant un cadre pratique et modulaire de communication entre le noyau et les utilisateurs.

encadré 2

nf_conntrack

Deux implémentations du suivi de connexions sont disponibles dans les noyaux postérieurs à 2.6.18. Une nouvelle version, `nf_conntrack`, est apparue dans Linux 2.6.18 et le choix entre les deux implémentations s'effectue lors de la compilation du noyau.

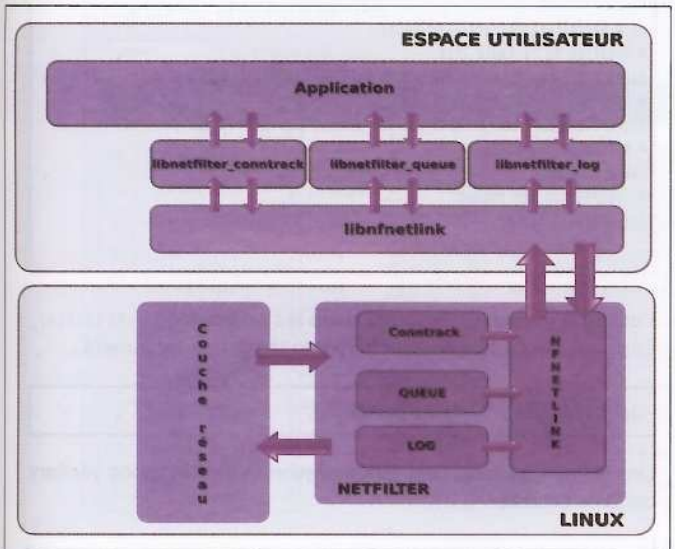
`nf_conntrack` a été développé pour pallier la dépendance à IPv4 de la version précédente `ip_conntrack`. Il a en effet été décidé de généraliser `ip_conntrack` de manière à pouvoir l'utiliser pour IPv4 et IPv6. Ceci factorise le code de manière conséquente et permet de gérer sans duplication le suivi de connexions des protocoles.

Cette évolution a offert à Linux un suivi de connexions IPv6 fonctionnel. Il y a cependant cohabitation des deux infrastructures jusqu'à ce qu'il soit possible d'ici quelques versions du noyau de retirer l'ancien code. Le travail de développement et de portage des modules de suivi de connexions applicatives vers le nouveau système est terminé depuis quelque temps et la suppression de l'ancienne implémentation ne devrait plus tarder.

Difficilement consultable, cette structure n'en est pas moins essentielle pour l'administrateur du pare-feu, puisqu'elle contient l'information sur les sessions actives. Au même titre que les journaux, elle fournit une vue de l'activité sur le réseau.

Appelé Nfnetlink, il s'agit d'un système de multiplexage dédié à Netfilter construit au-dessus de Netlink. Netlink est un système de communication par *socket* entre le noyau et l'espace utilisateur. Il est utilisé par des systèmes comme le routage ou les interfaces Tun. Nfnetlink utilise ainsi un canal de Netlink pour échanger des messages concernant plusieurs sous-systèmes :

- `nflog` : Journalisation ;
- `nfqueue` : Décision en espace utilisateur ;
- `nfconntrack` : Interaction avec le suivi de connexions.



2 *Architecture de Nfnetlink*

Nfnetlink offre un socle commun aux systèmes d'échanges de message (`conntrack`, `queue`, `log`) de Netfilter et permet d'avoir dans l'espace utilisateur une bibliothèque qui permet les communications bas niveau avec le noyau et sur laquelle s'appuie un jeu de bibliothèques de plus haut niveau :

- `libnetfilter_queue` : Gestion de la décision en espace utilisateur ;
- `libnetfilter_log` : Journalisation ;
- `libnetfilter_conntrack` : Interaction avec le conntrack.

Libnetfilter_conntrack

La bibliothèque `libnetfilter_conntrack` répond à plusieurs problématiques :

- consultation de la table des connexions et de celle des expects ;
- modification des entrées :
  - suppression ;
  - modification ;
  - création ;
- écoute des événements.

Si l'absence de résolution des deux premières problématiques se faisait cruellement sentir, la troisième est une évolution spontanée qui montre tout son intérêt lorsque l'on veut suivre l'évolution du suivi de connexions en temps réel. C'est le cas du projet de réplication de suivi de connexions, conntrackd **[11]**, ou encore du pare-feu authentifiant NuFW.

La bibliothèque `libnetfilter_conntrack` s'est vue dès l'origine utilisée par un outil, `conntrack`, dont la fonction est de manipuler le suivi de connexions et d'en extraire les informations.

On notera aussi l'ajout de la possibilité de travailler soit sur la table des connexions, soit sur la table des expects.

La sortie de `conntrack -E` est instructive, puisqu'elle nous éclaire sur la vie d'une connexion dans le conntrack. La capture d'écran suivante rassemble l'ensemble des événements relatifs à une connexion ssh (port 22) depuis 12.2.3.4 vers 15.12.4.5.

```
    [NEW] tcp      6 120 SYN_SENT src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   [UNREPLIED] src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039
 [UPDATE] tcp      6 60 SYN_RECV src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039
 [UPDATE] tcp      6 432000 ESTABLISHED src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039 [ASSURED]
 [UPDATE] tcp      6 120 FIN_WAIT src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039 [ASSURED]
 [UPDATE] tcp      6 30 LAST_ACK src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039 [ASSURED]
 [UPDATE] tcp      6 120 TIME_WAIT src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22
                   src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039 [ASSURED]
[DESTROY] tcp      6 src=12.2.3.4 dst=15.12.4.5 sport=1039 dport=22 packets=24
                   bytes=3991 src=15.12.4.5 dst=12.2.3.4 sport=22 dport=1039
                   packets=21 bytes=3523
```

Les trois événements initiaux sont déclenchés par la poignée de main en trois temps réalisant l'initialisation de la connexion TCP. Au troisième paquet, la connexion passe à « établie » et se voit attribuer un timeout de 43200 secondes, soit 5 jours. Le quatrième est déclenché par la fermeture de la connexion suite à une requête de l'utilisateur. Les derniers événements aboutissent à la suppression de la connexion du conntrack.

Interactions avec le conntrack

Consultation

L'option `-L` de `conntrack` liste les connexions en offrant une sortie comparable à celle du fichier `ip_conntrack`.

Modification

La modification d'entrée est l'un des grands apports de `libnetfilter_conntrack`. Son utilité pratique est double :

⇨ Modifier la marque pour changer la prise en compte de la connexion au niveau de la qualité de service et du routage (voir encadre 3).

⇨ Modifier le timeout d'une connexion pour déclencher son expiration à une date donnée. Cela nécessite un noyau supérieur à 2.6.18 ou l'application d'un patch (voir [4]) développé pour NuFW qui l'utilise pour implémenter des règles horaires strictes.

Ainsi, la marque d'une connexion est modifiable avec l'utilitaire `conntrack` par la commande : `conntrack -U -m 20 -s 15.131.19.16 -d 192.168.1.2 -p tcp --orig-port-src 22 --orig-port-dst 44014`

encadré 3

Linux et le marquage de paquets

La structure interne de stockage de paquet de Linux, le `sk_buff` comporte un grand nombre de champs parmi lesquels un entier appelé « mark » qui est modifiable et utilisable par Netfilter. Ce champ est consulté par d'autres systèmes tels que le routage et la qualité de service qui l'utilisent pour aiguiller les paquets sur certaines routes ou dans certaines files de qualité de service.

Bien évidemment interne, cette marque meurt avec le `sk_buff` et les informations stockées dans cette marque disparaissent avec elle. Cela peut être problématique dans la mesure où bon nombre de traitements s'effectuent uniquement sur le paquet d'initialisation. Aussi, une marque liée à la connexion, connmark, a été développée et introduite par Henrik Nordström vers 2.4.18. Cette marque persiste tout au long de la vie de la connexion pour garantir un traitement similaire à tous les paquets de la connexion. Un mécanisme de transfert de la marque de connexion vers la marque de paquet (voir [7]), grâce à des règles `iptables`, assure la compatibilité de ce système avec les outils existants (tel `tc` ou `ip`).

Jamais à court d'imagination, les développeurs du noyau ont décidé de parfaire le tableau en ajoutant une troisième marque, secmark (voir [8]). SELinux butait sur un problème de classification des paquets entrants et n'offrait qu'une solution incomplète. Les développeurs de SELinux ont alors décidé d'utiliser `iptables` pour sélectionner et étiqueter les paquets. Il revient ensuite à SELinux d'utiliser cette marque pour appliquer la politique de sécurité.

Mais c'est sans doute la modification la plus extrême, la destruction, qui a un des plus grands intérêts. Les flux indésirables peuvent ainsi être bloqués par l'administrateur de manière automatique et sûre. La syntaxe de `conntrack` est la suivante :

```
conntrack -D -s 15.131.19.16 -d 192.168.1.2 -p tcp --orig-port-src 22 --orig-
        port-dst 44014
```

Création d'entrée

Parallèlement à ces fonctions attendues, un nouveau champ de possibilités s'ouvre grâce à la création d'entrées de type `expect` ou `connexion`. Le but avoué est simple : permettre la gestion des protocoles complexes en espace utilisateur.

L'apparition de certains protocoles et notamment ceux liés à la Voix sur IP, comme SIP ou H323, a complexifié le travail des modules de suivi de connexions chargés de renseigner les expects en détectant et analysant le protocole. Les développeurs ont donc souhaité pouvoir faire passer cette tâche dans l'espace utilisateur pour bénéficier de plus de souplesses.

Malheureusement, par manque de temps, l'infrastructure nécessaire à cette évolution n'est pas encore en place.

Pynetfilter_conntrack

Un binding Python pour libnetfilter_conntrack

Comme nous avons pu le voir dans la section précédente, la syntaxe de l'utilitaire `conntrack` est loin d'être triviale. Du côté des développeurs, c'est encore pire au niveau de `libnetfilter_conntrack`. Une couche d'abstraction dans un langage de plus haut niveau était donc nécessaire pour rendre le système utilisable par le plus grand nombre. La société INL a donc développé un *binding* Python pour `libnetfilter_conntrack` qu'elle distribue sous licence GPL.

Un outil pour l'administrateur

Le code suivant permet de lister et supprimer les entrées de la table de connexions à destination du port 80 :

```
#!/usr/bin/python
from pynetfilter_conntrack import NetfilterConntrack, CONNTRACK, \
    L3PROTONUM_REVERSE_NAMES, IPPROTO_REVERSE_NAMES
from optparse import OptionGroup, OptionParser
from IPy import IP
import sys

nf = NetfilterConntrack(CONNTRACK)
# Creation de la table IPv4
table = nf.create_table(L3PROTONUM_REVERSE_NAMES["ipv4"])
# Filtrage sur le port 80 et TCP
table = table.filter(IPPROTO_REVERSE_NAMES["tcp"], orig_dst_port=80)
# Affichage des entrées
table.display()
# Suppression des entrées
for entry in table:
        nf.delete_conntrack(entry)
```

L'utilitaire `conntrack.py` fourni dans les sources de `pynetfilter_conntrack` permet de réaliser la même tâche très facilement :

```
conntrack.py delete --orig-dst-port=80
```

Une autre application est l'énumération des connexions vérifiant certains critères :

```
conntrack.py list -s 192.168.1.3 --orig-dst-port=80
```

pynetfilter_conntrack est donc un outil intéressant pour construire des briques plus évoluées de gestion du suivi de connexions.

Pyctd

L'idée de Pyctd est de s'appuyer sur XML-RPC pour exporter les fonctions de gestion du suivi de connexions sur d'autres systèmes et de pouvoir le modifier à distance. Un *frontend* PHP a été développé pour fournir une couche de présentation agréable. Également développé par INL, Pyctd est disponible sous licence GPL.

Grace à Pyctd, l'administrateur du pare-feu peut modifier dynamiquement certains paramètres comme le timeout ou la marque de connexion. La modification du timeout permet de définir la date de fin d'une connexion sélectionnée. Celle de la marque de connexion module le comportement des outils de qualité de service et de routage par rapport à cette connexion. L'affichage est d'ailleurs paramétrable pour indiquer la qualité de service correspondant à la marque et proposer sa modification grâce à un menu déroulant.

Pyctd peut s'appuyer sur le pare-feu authentifiant NuFW (voir [9]) pour afficher l'utilisateur à l'origine de la connexion. NuFW maintient en effet une base SQL contenant l'ensemble des connexions authentifiées actives. Lors d'une requête cliente, Pyctd enrichit l'affichage en couplant l'interrogation du suivi de connexions Netfilter avec une requête sur cette base.

Conclusion

L'apparition dans le noyau Linux 2.6.14 de l'infrastructure Nfnetlink a augmenté les capacités d'interaction avec le suivi de connexions. L'action directe de l'utilisateur sur cet élément important de sécurité offre à l'administrateur un nouveau champ de contrôle. De nouveaux outils exploitant ces fonctionnalités concrétisent cette potentialité et donnent à l'administrateur de pare-feu de nouveaux moyens de connaître et de contrôler l'activité de son réseau et donc d'en renforcer la sécurité.

Liens

[1] Netfilter : **http://www.netfilter.org/**

[2] *Netfilter performance testing* : **http://people.netfilter.org/kadlec/nftest.pdf**

[3] pynetfilter_conntrack : **http://software.inl.fr/trac/trac.cgi/wiki/pynetfilter_conntrack**

[4] Pyctd : **http://software.inl.fr/trac/trac.cgi/wiki/pyctd**

[5] Patch Fixed timeout : **http://www.nufw.org/**

[6] *Real Stateful TCP Packet Filtering in IP Filter* : **http://www.iae.nl/users/guido/papers/tcp_filtering.ps.gz**

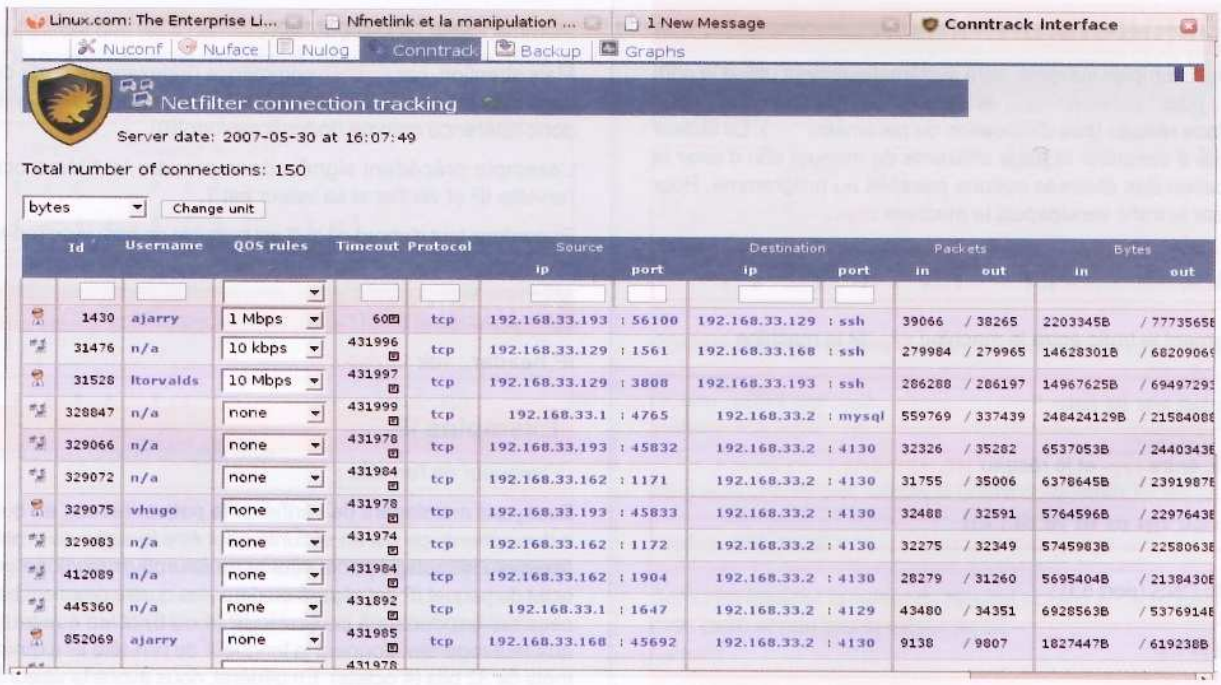
[7] Netfilter connmark : **http://home.regit.org/?page_id=7**

[8] *New secmark-based network controls for SELinux* : **http://james-morris.livejournal.com/11010.html**

[9] Site de NuFW : **http://www.nufw.org/**

[10] INL : **http://www.inl.fr/**

[11] conntrackd : **http://people.netfilter.org/pablo/conntrackd/**



3 *Le frontend PHP de Pyctd*

Utilisation avancée de TCPDUMP

mots clés : *TCP/IP / filtre BPF / réseau / sniffer / paquets*

Préambule

`tcpdump` est considéré comme un outil d'analyse réseau et est en général disponible de façon standard sous Unix (Linux/*BSD/...), ainsi que sous Windows sous le nom de Windump [1]. Si une utilisation basique est bien souvent suffisante pour régler tel ou tel problème, il en va tout autrement lors d'analyses plus pointues. C'est à ce moment que `tcpdump` va dévoiler sa richesse et ses fonctionnalités étendues au prix d'une certaine gymnastique intellectuelle mêlant connaissance des principaux en-têtes réseau et algèbre booléenne (les ET, OU et NON logiques).

Rappels sur les filtres tcpdump

- [x:y] : débute à l'octet 'x' et lit 'y' octets ;
- p[x:y] : la même chose, mais utilise l'en-tête « p » ;
- p[x:y] & z = 0 : aucun bit sélectionné en appliquant le masque 'z' sur p[x:y] ;
- p[x:y] & z != 0 : certains bits sont sélectionnés en appliquant le masque 'z' sur p[x:y] ;
- p[x:y] & z = z : tous les bits sélectionnés en appliquant le masque 'z' sur p[x:y] sont présents ;
- p[x:y] = z : p[x:y] a uniquement les bits sélectionnés en appliquant le masque 'z'.

Utilisation simple

Dans les exemples suivants, sera systématiquement utilisé le nom `tcpdump` pour `tcpdump`/`windump` et sera fait abstraction de la notion d'interface réseau (pas d'utilisation du paramètre `'-i'`). Le lecteur est invité à consulter la page afférente du manuel afin d'avoir la signification des diverses options passées au programme. Pour visualiser le trafic vers/depuis la machine `mygw` :

```
tcpdump host mygw
```

Uniquement le trafic entre la machine `mygw` et la machine `zephyr` :

```
tcpdump host mygw and zephyr
```

Le trafic entre `mygw` et le réseau `192.168.1.0/24` :

```
tcpdump host mygw and net 192.168.1.0/24
```

Le trafic DNS (port 53) :

```
tcpdump port 53
```

Il est possible d'utiliser la négation pour éviter certains flux, ici tous sauf ceux de la machine `orion` :

```
tcpdump not host "orion"
```

`tcpdump` est bien sûr capable d'utiliser certains mots clés pour sélectionner les paquets adéquats, ici les paquets `icmp` :

```
tcpdump icmp
```

Utilisation avancée

Dans la « vraie vie », il est souvent nécessaire pour régler un problème réseau, qu'il soit fonctionnel ou bien de sécurité, de « d'entrer » dans les détails et, dans ce cas de figure, `tcpdump` se révèle un parfait allié.

Nous avons montré ci-dessus la requête pour ne capturer que les paquets `icmp`, mais la même peut aussi s'écrire :

```
tcpdump 'ip[9] = 1'
```

La compréhension de ce type de notation est essentielle pour la suite des exemples, car à la base de l'utilisation avancée de `tcpdump`.

- 'ip' signifie prendre l'en-tête ip;
- '[9]' prendre l'octet numéroté 9;
- '=1' la valeur de comparaison.

Mais attention, car `tcpdump` commence l'indexation à partir de 0 et, dans tous les exemples suivants, le premier octet de l'en-tête sera donc référencé comme (ip/tcp/icmp/udp)[0].

L'exemple précédent signifie donc prendre le 10ème octet de l'en-tête IP et vérifier si sa valeur est 1.

Regardons tout d'abord ce qu'il est possible de faire avec l'en-tête IP.

Filtres IP

IP header : voir tableau 1.

Exemples IP

- longueur de l'en-tête ;

Essayons maintenant de vérifier si le paquet réseau en question a des options, ce qui en sécurité peut être le signe de problèmes (exemple le routage par la source ou *source routing*). Le premier octet du paquet IP est séparé en deux, les quatre premiers bits sont ceux correspondant à la version IP (4 ou 6) et les 4 autres, ceux qu'il faut regarder, donnent la longueur de l'en-tête IP exprimée en mots de 32 bits (4 octets). En général, nous avons la valeur '5', car un en-tête IP « standard » fait 20 octets ou 5 mots de 32 bits, une valeur supérieure serait le signe de présence d'options.

Julien Desfossez
julien.desfossez@revolutionlinux.com

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

00 01 02 03	04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Version	Internet Header Length	Type of Service	Total Length
Identification		R DF MF	Fragment offset
Time to Live		Protocol	Header checksum
Source IP address			
Destination IP address			
Options and padding			

T1 *IP header*

00 01 02 03	04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Data Offset	reserved	ECN U A P R S F	Window
Checksum			Urgent Pointer
Options and padding			
Data			

T2 *TCP header*

Il ne faut garder que les 4 derniers bits du premier octet : comment faire ? Tout simplement en utilisant un masque « binaire ». Voici la représentation binaire de cet octet :

```
      [Version]     [Longueur en-tête]
Valeur 8  4  2  1    8  4  2  1
       0  1  0  0    0  1  0  1      4 et 5
```

On ne veut garder que la deuxième partie. Masquons la première (la fonction « ET booléenne ») avec des « 0 » et la seconde avec des « 1 »

```
       0  0  0  0    1  1  1  1      (0x0f en hexadécimal)
```

Ce qui donne comme résultat final :

```
       0  0  0  0    0  1  0  1
```

Le résultat du masquage retourne uniquement les quatre derniers bits, ce que nous cherchions. La commande `tcpdump` pour savoir si la longueur de l'en-tête IP est supérieure à 20 octets est donc :

```
tcpdump 'ip[0] & 0x0f > 5'
```

⇨ détection de la fragmentation.

Toujours dans l'en-tête IP, détectons si fragmentation il y a. Dans ce cas, tous les en-têtes IP (sauf le dernier fragment) ont le bit « MF » positionné. L'information est dans le septième octet (troisième bit) de l'en-tête.

```
                Septième octet
Bit               MF
Valeur 8  4  2  1    8  4  2  1
       0  0  1  0    0  0  0  0      0x20 en hexa, 32 en décimal
```

Plusieurs façons de faire existent pour obtenir la valeur voulue :

```
tcpdump 'ip[6] & 0x20' = 32
tcpdump 'ip[6] & 0x20' !=0
```

Mais pourquoi ne pas utiliser la notation suivante ?

```
tcpdump 'ip[6] & 0x20' = 1
```

Car ici on testerait la valeur du résultat obtenu en masquant l'octet avec 0x20 et non pas la valeur du bit.

Filtres TCP

TCP header : voir tableau 2.

➪ Les paquets TCP avec le drapeau 'SYN'

Le bit qui indique que ce drapeau est présent est dans le quatorzième octet à la septième position.

```
          Quatorzième octet
 R  R  U  A    P  R  S  F
2e3 2e2 2e1 2e0  2e3 2e2 2e1 2e0
 0  0  0  0    0  0  1  0       0x02 en hexa, 2 en décimal
```

La commande `tcpdump` pour capturer tous les paquets avec le seul drapeau « SYN » est donc :

```
tcpdump 'tcp[13] & 0xff = 2'
```

On pourrait imaginer utiliser la commande suivante pour obtenir le même résultat, sauf qu'en masquant avec des zéros nous récupérerions aussi bien les paquets ayant initialement les drapeaux « SYN » que « SYN/ACK », etc.

```
tcpdump 'tcp[13] & 0x02 = 2'
```

➪ paquets TCP avec drapeaux SYN/FIN, combinaison anormale :

```
tcpdump 'tcp[13]=3'
```

➪ paquets TCP avec drapeau ACK et une valeur de ACK de 0 :

note

Il s'agit ici d'une combinaison peu courante, car tous les segments TCP ont normalement une valeur minimale de 1 pour les ACK.

```
tcpdump 'tcp[13] = 0x10 and tcp[8:4] = 0'
```

➪ paquets TCP avec un port de destination inférieur à 1024 :

```
tcpdump 'tcp[2:2] < 1024'
```

➪ attaque Winnuke:

```
tcpdump '(tcp[2:2] = 139) && (tcp[13] & 0x20 != 0) && (tcp[19] & 0x01 = 1)'
```

Il suffisait d'envoyer une donnée spéciale et non traitée par certaines piles IP qui mettait en échec le système d'exploitation. La donnée spéciale était le flag « Urgent Pointer » de l'en-tête TCP.

➪ attaque Land :

```
tcpdump '(tcp[0:2] = tcp[2:2]) && (ip[12:4] = ip[16:4])'
```

L'attaque éponyme consiste à envoyer un paquet IP (usurpé) dont l'adresse IP source est identique à l'adresse IP de destination.

Filtres UDP

UDP header : voir tableau 3.

Exemple UDP

➪ Traceroute avec recherche de TTL à '1' :

```
tcpdump 'udp[2:2] >= 33000 and udp[2:2] < 34000 and ip[8]=1'
```

Filtres ICMP

ICMP header : voir tableau 4.

Il est bien sûr possible de travailler aussi sur l'en-tête ICMP. Voici quelques exemples :

Exemples ICMP

➪ ICMP type 3 code 4 (*Need to Fragment but bit DF is set*) :

```
tcpdump '((icmp[0] = 3) and (icmp[1] = 4))'
```

Ce message ICMP indique que la taille du paquet est trop importante et qu'il faut le fragmenter alors que le bit DF est positionné.

note

Ce message fait partie du mécanisme de découverte du MTU (*Path MTU Discovery*) et il est recommandé de ne pas le bloquer sur les *firewalls* afin d'éviter de voir disparaître nos paquets dans des « trous noirs » réseau.

➪ Loki :

```
tcpdump '(((icmp[0] = 0) or (icmp[0] = 8)) and ((icmp[6:2] = 0xf001)
or (icmp[6:2] = 0x01f0))'
```

Loki est une *backdoor* permettant d'établir un canal caché en faisant passer des données dans des paquets ICMP *echo request* (valeur 8) et *echo reply* (valeur 0). Il est possible de le repérer en cherchant, dans les octets 6 et 7 de l'en-tête ICMP, les valeurs 0xf001 ou 0x01f0.

tcpdump et les autres couches

`tcpdump` n'est pas uniquement réservé aux couches 3 et 4. Il est également capable de travailler plus bas. Par exemple, pour analyser le trafic Ethernet *multicast*, il est possible d'utiliser le filtre suivant :

```
tcpdump 'ether[0]  &  1 != 0'
```

Il est aussi possible de descendre encore plus bas en changeant le *data link type* (le média de la couche liaison). Lorsque c'est

00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port		Destination Port
Length		Checksum
Data		

T3 *UDP header*

00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Type	Code	ICMP header checksum
Data		

T4 *ICMP header*

possible, `tcpdump` utilise par défaut EN10MB (correspond à Ethernet), mais les amis des réseaux 802.11 seront ravis de le changer pour IEEE802_11 ou IEEE802_11_RADIO.

remarque

L'option `-L` permet d'afficher les data link types supportés par le pilote de l'interface réseau sélectionnée.

Dans le cas présenté, le pilote supporte trois DLT différents. Nous allons choisir le type « IEEE802_11_RADIO » afin de récupérer les données « brutes » et les en-têtes `radiotap` ajoutées par le pilote.

```
# tcpdump -i ath0 -L

3 link types supported:
        IEEE802_11_RADIO
        IEEE802_11
        EN10MB

# tcpdump -i ath0 -y IEEE802_11_RADIO -v
tcpdump: listening on ath0, link-type IEEE802_11_RADIO
22:02:48.398610 802.11: beacon, timestamp 1451206041987, interval 100,
caps=401<ESS,SHORT_SLOTTIME>, ssid (OpenWrt), \
            rates 1M 2M 5M 11M 18M 24M 36M 54M, ds (chan 1), tim 0x00030000,
            erp 0x00, 47:1 0x00, xrates 6M 9M 12M 48M, \
            vendor 0x001018020000, <radiotap v0, 1Mbit/s, chan 1, 11b, antenna
            0, rssi 58/64>
22:02:48.450056 802.11: authentication request, <radiotap v0, 1Mbit/s, chan 1,
11b, txpower 30dBm, antenna 0>
22:02:48.451503 802.11: authentication response, <radiotap v0, 1Mbit/s, chan 1,
11b, antenna 0, rssi 58/64>
22:02:48.451525 802.11: association request, <radiotap v0, 1Mbit/s, chan 1,
11b, txpower 30dBm, antenna 1>
22:02:48.453944 802.11: association response, <radiotap v0, 1Mbit/s, chan 1,
11b, antenna 0, rssi 58/64>
```

Conclusion

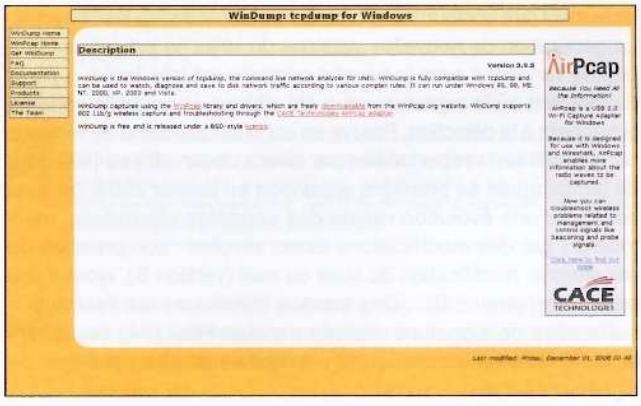
Nous avons vu ici une présentation des possibilités offertes par l'analyseur réseau `tcpdump`. Le but de cette fiche technique est de donner un aperçu des capacités de filtrage de cet outil vital pour les administrateurs réseau et sécurité. En guise d'exercice pratique, voici un filtre qui devrait vous occuper :

```
tcpdump '((ip[6] & 0x20 = 0) and (ip[6:2] & 0x1fff != 0)) and ((65535 <
(ip[2:2] + 8 * (ip[6:2] & 0x1fff))))'
```

On s'intéresse ici au dernier fragment d'un paquet. Le champ `total length` est au maximum et la longueur totale (exprimée en mots de 8 octets) des données est supérieure à 65535. Un cas typique de « ping de la mort », encore nommé « *ping of death* », attaque bien connue de la fin des années 90 !

Référence

[1] WinDump : tcpdump for Windows : **www.winpcap.org/windump/**



Détection comportementale de malwares

Il s'agit d'un état de fait : certains antivirus sont capables de détecter la quasi-totalité des malwares connus, affichant des taux de réussite proches de 95%. Ces résultats satisfaisants pour les meilleurs du marché restent beaucoup plus mitigés pour d'autres pourtant réputés phares [01]. De telles disparités ne devraient pas avoir lieu d'être et la différence d'efficacité devrait donc se faire sur leur aptitude à détecter les menaces inconnues. Quand on connaît la rapidité de propagation des attaques récentes, il s'agit d'un enjeu majeur. Pour y faire face, la plupart des éditeurs du marché mettent en avant des techniques de détection comportementale sans pour autant détailler leur fonctionnement. L'objet de cet article n'est pas de réaliser un comparatif des différents produits, mais, simplement, de présenter un état de l'art des techniques comportementales dont on a pu dire tout et son contraire.

mots clés : ***virus informatiques / détection antivirale / analyse comportementale / polymorphisme / schéma de détection / stratégie de détection***

1. Les techniques de scanning vouées à l'échec

Historiquement, les techniques de *scanning* ont été les premières utilisées pour la détection de *malwares* et restent encore actuellement la base des antivirus traditionnels. Le fonctionnement des scanners repose sur la recherche dans les fichiers suspects de patterns d'octets répertoriés dans une base. Ces patterns appelés « signatures » doivent présenter un caractère à la fois discriminant et non incriminant pour les logiciels légitimes [2, p. 147]. De fait, ces techniques fondées sur la forme se limitent uniquement à la détection des infections connues.

Si elles présentent des avantages indéniables : taux de faux positifs quasi nul et, surtout, une complexité algorithmique maîtrisée et acceptable, ces techniques restent néanmoins dépassées par l'évolution rapide des attaques virales.

La première limitation est liée à l'extraction bien souvent manuelle de ces signatures, requérant une analyse du code virale extrêmement coûteuse en temps. Reste alors le problème de leur distribution auprès des utilisateurs dont la taille des bases ne cesse d'augmenter. Dans un contexte où des vers tels que `Sapphire` sont capables d'infecter en moins de 10 minutes plus de 90% des machines vulnérables, menace et protection ne se placent plus sur la même échelle de temps.

De plus, les techniques de scanning peuvent, malheureusement, facilement être contournées comme l'ont bien compris les *copycats*. La génération de nouvelles versions du virus ne présentant que peu de variations avec la souche originale est très facile. À partir du moment où la signature en a été altérée, la nouvelle variante échappera à la détection. Preuve en est la cinquantaine de versions du ver mail `Bagle` répertoriées par divers observatoires [03]. Peu de temps après sa première apparition en janvier 2004, on a pu constater une évolution rapide des variantes répandues, mais toujours par des modifications assez simples : compression du ver attaché, modification du sujet du mail (version B), ajout d'une *backdoor* (version C)... Des travaux théoriques sur l'extraction boîte noire de signature réalisés par Eric Filiol [04] soulignent la facilité de telles pratiques permises par des schémas de signatures faibles.

Que se passe-t-il dès lors que ces modifications deviennent automatiques, à savoir que le malware devient capable de mutation au cours de sa propagation ? La première génération significative de mutation a vu le jour avec le polymorphisme [5, p. 140]. Les virus polymorphes ont leur code entièrement chiffré afin de dissimuler les signatures potentielles. Une simple variation de la clé de chiffrement entraîne alors une modification totale de la forme du virus. En contrepartie, l'ajout au code d'une routine de déchiffrement devient nécessaire au risque que cette dernière ne devienne elle-même une signature. Si le polymorphisme a pu être contourné par une simple émulation pour accéder au code en clair, la recherche de signature devient bien plus complexe dans le cadre du métamorphisme. Le malware n'est plus simplement chiffré, mais son corps complet subit un certain nombre de modifications affectant sa forme tout en conservant sa fonctionnalité [5, p. 148]. Le processus de mutation du virus commence toujours par un désassemblage de son code qui subit alors une série de transformations d'*obfuscation* avant d'être réassemblé. Une simple analyse syntaxique ne suffit alors plus à le détecter, Diomidis Spinellis ayant démontré que la détection de virus mutants par signature est NP-complet [06].

Toutes ces limitations, bien connues du monde antiviral, ont motivé la recherche d'autres solutions. L'approche comportementale est une de ces solutions de seconde génération qui devrait faciliter la détection de virus inconnus sous certaines conditions.

2. Principe général de la détection comportementale

La particularité de la détection comportementale réside dans l'identification des actions réalisées par le malware remplaçant la simple recherche de marqueur. Elle est donc théoriquement capable de détecter à partir de leur mode opératoire des menaces inconnues utilisant des techniques virales connues. Dans le cas de malwares totalement novateurs, le résultat reste malheureusement incertain. En contrepartie, il n'est plus possible d'identifier précisément la menace, ce qui n'est pas sans poser de problèmes lorsqu'il faut déterminer la contremesure à appliquer. De plus, certaines techniques, comme le chiffrement ou l'obfuscation souvent associées à un caractère malicieux,

Éric Filiol[1], **Grégoire Jacob**[1/2], **Hervé Debar**[2]
[1] ESAT, Laboratoire de virologie et de cryptologie
efiliol@esat.terre.defense.gouv.fr
[2] France Télécom R&D, MAPS/NSS
{gregoire.jacob,herve.debar}@orange-ftgroup.com

sont également utilisées dans le cadre de la protection logicielle. Cet exemple illustre la ligne très fine qui sépare une action malicieuse d'une action légitime. Le flou de cette distinction est à même de provoquer des alertes injustifiées expliquant les taux de faux positifs pour la détection comportementale supérieurs à ceux de l'analyse par signatures.

L'approche comportementale pour détecter les virus n'est pas à proprement parler nouvelle. Fred Cohen, dans ses premiers travaux de formalisation, y faisait déjà référence. Il y explique que la détection comportementale se résume à définir ce qu'est ou n'est pas un usage légitime des services d'un système [07]. Il a pu notamment démontrer qu'au même titre que l'analyse de forme, cette méthode de détection était indécidable. Pour bien comprendre le principe développé ici, il est important de bien faire la distinction entre virologie et détection d'intrusion, car ceux-ci reposent sur deux modélisations des comportements contraires. En intrusion, la détection comportementale ou détection d'anomalie consiste à mesurer une déviance par rapport à un modèle de comportements légitimes [08]. En virologie, la modélisation s'appuie à l'opposé sur les comportements suspects caractéristiques des malwares. Le choix de cette approche est très facile à comprendre si l'on considère la quantité pharaonique d'applications existantes, toutes de natures très différentes. Le comportement d'un client mail présentant une activité réseau intense n'aura vraisemblablement aucune ressemblance avec celui d'un lecteur multimédia qui, depuis un fichier, décode des données avant de les restituer, d'où la complexité d'établir de tels profils. La formalisation de ces comportements suspects reste, pour la majorité des travaux, issue de la virologie abstraite. Ils utilisent notamment la théorie des fonctions récursives afin de modéliser les principaux comportements infectieux tels que l'autoreproduction, la propagation, le polymorphisme ou la furtivité. Si ces modèles ont permis d'établir l'indécidabilité de la détection, leur modélisation s'avère difficilement utilisable de manière directe dans le cadre de la détection comportementale. Il n'existe malheureusement pas de formalisation opérationnelle globale, des modèles comportementaux spécifiques étant redéfinis d'un système à l'autre. L'établissement d'une modélisation de référence reste donc un problème ouvert.

Traditionnellement, l'analyse comportementale est associée à un mode d'exécution dynamique où l'antivirus est résident en mémoire. Néanmoins, il faut garder à l'esprit que l'identification des comportements reste possible de manière statique. Chaque mode possède ses propres caractéristiques qui peuvent se révéler dans certains cas, avantageuses et, dans d'autres, problématiques. Dans cet article, nous avons voulu adopter une vision la plus large possible de comportements en englobant ces deux modes.

3. Surveillance dynamique des comportements

La détection en mode dynamique est à rapprocher des méthodes de test de programmes par simulation. L'approche utilisée est de type boîte noire avec la surveillance de signes extérieurs d'infection au cours de l'exécution du programme. Le processus de détection se décompose toujours en trois étapes successives décrites dans les prochaines sous parties, à savoir (1) la capture des données, (2) leur analyse afin d'atteindre un niveau d'abstraction supplémentaire permettant (3) la prise de décision finale.

3.1 Capture des données

La détection d'un malware au cours de son exécution repose sur des éléments observables par un agent extérieur. Selon le principe du physicien Schrödinger, la capture de ces éléments doit introduire le moins possible de perturbations dans le fonctionnement du système. Inversement, elle doit aussi être capable de résister à d'éventuelles perturbations générées volontairement par le malware. La première source d'information sur les actions d'un virus a d'abord été l'interception des interruptions, remplacée ensuite par les appels système avec l'apparition des systèmes 32 bits. On appellera la séquence des appels réalisés par un processus « sa trace », un exemple étant détaillé à la figure 1 (page suivante). L'intérêt majeur de cette source de données réside dans le fait qu'elle constitue un point de passage obligatoire pour accéder aux objets du noyau et aux ressources matérielles depuis le monde utilisateur. Les choix réalisés au cours de cette étape sont primordiaux. Travaillant de manière connexe sur la modélisation des comportements légitimes en intrusion, Stéphanie Forrest et al. soulignent à juste titre l'importance du choix des données capturées et de leur modélisation [09]. Selon la nature de ces informations et la représentation choisie (en séquence ou en spectre de fréquence par exemple), l'analyse et la prise de décision seront fortement influencées. Dans le cadre de la surveillance comportementale, une représentation en séquence est majoritairement employée, l'analyse spectrale étant une technique de détection bien distincte.

```
Process Id: 2884 "Word.exe"
Time: 16/01/2007 1:53:34:536
\\ZwReadFile//
hFile = C:\document.doc:0x24E6B0
lpBuffer = 0x13E67C
nNumberOfBytesToRead = 10
nByteOffset = 0
```

1 *Capture d'un appel système. Lors de la capture, les informations temporelles et les paramètres constituent des informations additionnelles intéressantes. Selon le processus surveillé, les appels sont alors filtrés à l'aide de l'identifiant.*

En plus de la nature des données, les conditions de capture constituent un second facteur important. Ces conditions influencent de manière importante les techniques de collecte utilisées, la performance et les ressources nécessaires.

De manière évidente, les conditions temps réel permettent de surveiller la progression des programmes en direct sur le système. Dans ce cadre, la capture des appels système se fait alors bien souvent à l'aide d'un driver système utilisant des techniques de *hooking*. La surcharge engendrée par l'interception et le traitement des appels peut être perceptible par l'utilisateur, mais reste inférieure aux ressources nécessaires à d'autres conditions de capture. Le temps réel est bien souvent critiqué, car les actions malveillantes sont exécutées de manière effective sur le système. Il est donc primordial que la décision du caractère infectieux d'un programme soit prise suffisamment tôt avant que le point critique ne soit atteint. Pour pallier le problème, un mécanisme d'enregistrement des actions du malware et des états intermédiaires du système peut être déployé [10]. Ce compromis offre la possibilité de rétablir le système dans un état sain sous condition que le mécanisme et les enregistrements ne soient pas corrompus lors de l'attaque.

L'utilisation d'une *sandbox*, restreignant l'exécution du malware à un environnement confiné, garantit des conditions de capture plus sûres. Cette technique popularisée avec Java permet d'étudier l'exécution d'un code malveillant dans un espace mémoire étanche avec des privilèges et accès aux services réduits afin d'éviter l'infection du système. L'accès total à l'espace mémoire utilisé, combiné avec le déroulement contrôlé de l'exécution, offre des facilités d'analyse supérieures aux conditions réelles. Les ressources requises sont en contrepartie plus élevées, puisqu'on ajoute une couche intermédiaire de contrôle entre le programme et son environnement. Pour diminuer la surcharge, seules quelques portions de code jugées suspectes sont la plupart du temps analysées. Les sandboxes n'offrent malheureusement pas les mêmes possibilités qu'un système complet et peuvent être facilement détectées par des techniques d'anti-débogage, comme la surveillance du temps d'exécution ou l'utilisation de structures de gestion d'erreur. S'il décèle une tentative d'analyse, le malware pourra adapter son exécution pour échapper à la détection.

Les machines virtuelles, émulant un environnement complet, offrent le meilleur contrôle sur les conditions de capture. La machine hôte régulant tous les points d'accès au matériel de l'environnement virtuel, de nombreuses techniques de capture sont envisageables sans risques de dommage. Considérant une machine purement logicielle, la capture des appels système peut se faire au niveau du processeur émulé par reconnaissance d'instructions spécifiques (Intel `Sysenter` ou AMD `Syscall`). Le traitement est alors réalisé par le système hôte sans aucune trace visible pour l'environnement virtuel [11]. Par rapport aux sandboxes, les machines virtuelles offrent la possibilité de simuler des ressources fictives matérielles (connexions réseau) ou logicielles (clients messagerie ou P2P). Ces ressources sont souvent utilisées de manière détournée par les malwares pour se propager ou accéder à certaines informations. Ces interactions pourront donc être observées, mais les ressources nécessaires seront très importantes, ce qui rend ces environnements difficiles à déployer dans un contexte opérationnel à moins d'une machine dédiée. Ils restent pour l'instant utilisés par les experts et chercheurs dans une optique d'analyse ou de classification et non plus de détection. Bien que moins nombreuses et plus complexes, les machines virtuelles sont, elles aussi, sensibles à des techniques particulières de détection.

3.2 Analyse et prise de décision

De manière générale, un moteur de détection embarque un ensemble de connaissances sur les malwares afin d'appuyer sa décision. Dans le cadre de la détection comportementale, cette connaissance prend la forme d'une base de comportements modélisés selon la nature de ce moteur. Lors de l'analyse, les données capturées doivent être interprétées et transformées dans l'optique de faciliter leur confrontation aux comportements contenus dans la base. La nature du moteur détermine donc la modélisation des comportements et des captures ainsi que la méthode de confrontation. Nous avons donc choisi de faire une synthèse présentant l'analyse et la prise de décision ensemble, classées selon trois catégories principales.

3.2.1 Systèmes Experts

Ce type de moteur fonctionne sur une base de règles au cas par cas modélisant l'expertise d'un analyste. De la même manière qu'à la figure 2, une règle est définie pour chaque comportement reconnu comme hautement suspect. Chaque action isolée d'un programme y sera alors confrontée [12]. La cible de l'action et le niveau de privilège de l'appelant sont souvent des facteurs primordiaux, car ils facilitent la distinction entre une action légitime et une attaque.

```
Deny   WriteRun Registry Key
Deny   WriteWin.ini File
Deny   Terminate        Antivirus Process
```

2 *Règles de décision. Une règle spécifie toujours le type d'action concernée (lecture, écriture, ouverture, terminaison), la cible de l'action ainsi que la décision à prendre (permission, refus). En cas d'absence de règle pour une action, elle est autorisée par défaut.*

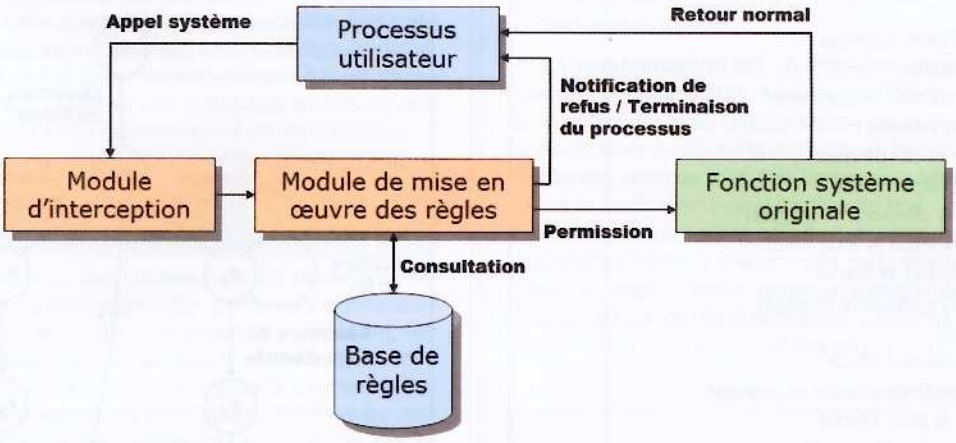
La décision de la nature malicieuse d'un comportement doit donc être prise de manière préemptive. Des systèmes tels que celui de la figure 3 permettent de capturer toute tentative d'action et de réagir en conséquence avant même sa résolution. Cette propriété a valu à ces systèmes proactifs la désignation de *behavioral blockers*.

De manière générale, ce type de moteur est enclin à des taux de faux positifs relativement élevés, car il est très difficile de juger de la nature légitime d'une action isolée sans corrélation. Une même action peut être malveillante dans le cadre de la mise en résidence pour qu'un malware s'exécute automatiquement, mais intervenir de manière tout à fait légitime lors de l'installation d'un logiciel par exemple.

3.2.2 Moteurs heuristiques

Historiquement, ils ont été les premiers moteurs utilisés pour la détection de comportements. Contrairement aux systèmes précédents, les actions capturées ne sont plus considérées de manière isolée, mais séquentiellement. Les moteurs heuristiques fonctionnent souvent couplés avec une sandbox afin de capturer les appels système ou interruptions, ainsi que leurs paramètres au niveau des instructions.

Il est important de bien faire la distinction entre les moteurs heuristiques en tant que technique utilisée pour la détection comportementale et la détection heuristique à proprement parler.



3 *Déploiement des règles. Pour chaque appel système capturé, un parcours des règles concernées est effectué. Si l'action est permise, le moteur rend la main à la fonction appelée. Dans le cas contraire, le processus reçoit une notification d'interdiction et peut même être terminé selon la contremesure établie.*

Cette dernière appuie sa prise de décision uniquement sur la présence d'anomalies dans la structure des exécutables (PE ou ELF) : tailles et noms de sections ou encore la localisation du point d'entrée. Un détecteur comportemental reposant sur un moteur heuristique comprendra toujours trois parties [13] :

⇨ Un **mécanisme d'association** étiquetant les différents comportements atomiques d'un malware. Ces comportements correspondent à l'interprétation fonctionnelle d'une ou plusieurs instructions comme l'illustre la figure 4. Deux séquences d'instructions équivalentes se verront toujours attribuer la même étiquette. Il existe alors deux types d'étiquetage. Le système pondéré utilise des poids obtenus par expérimentation pour quantifier la nocivité d'une action. Le système labellisé associe à chaque action capturée un *flag* représentant son interprétation fonctionnelle. La figure 5, page suivante, donne un exemple d'étiquetage fonctionnel sachant que chacun des comportements répertoriés correspond en réalité à plusieurs séquences d'instructions équivalentes.

```
Terminaison d'un programme
1.  MOV AX, ??4Ch
    INT 21          ;B8??4CCD21
2.  MOV AH, 4Ch
    INT 21          ;B44CCD21
3.  MOV AH, 4Ch
    MOV AL, ??h
    INT 21          ;B44CB0??CD21
4.  MOV AL, ??h
    MOV AH, 4Ch
    INT 21          ;B0??B44CCD21
```

4 *Comportements atomiques. Cet exemple tiré de la documentation du moteur heuristique de Symantec illustre l'association entre plusieurs séquences d'instructions équivalentes et une seule et même action atomique.*

⇨ Une **base de règles** qui définit les critères de la détection. Dans le cas d'un étiquetage par poids, la base contient une unique règle consistant en un seuil à partir duquel l'accumulation des comportements capturés trahit le caractère malicieux du malware. Dans le cas d'un étiquetage fonctionnel, un arbre de détection représentant les actions associées aux différents malwares est construit à l'aide des règles comme illustré figure 5, page suivante.

⇨ Enfin, une **stratégie** qui influence le déroulement de la détection par rapport aux règles. Dans le cas d'un étiquetage par poids, il s'agit de la fonction d'accumulation choisie à l'évaluation de chaque capture. Dans le cas contraire, la stratégie détermine le parcours algorithmique de l'arbre de recherche. Il en existe plusieurs types : glouton comme illustré figure 5, page suivante, tabou ou bien recuit simulé. Le choix de la stratégie est déterminant afin de trouver une solution approchée, mais néanmoins acceptable pour des problèmes de complexité rédhibitoire [5, p. 67].

3.2.3 Machines à états

De même que pour le moteur précédent, les machines à états modélisent des séquences d'appels système. Les comportements sont alors représentés comme des automates finis déterministes de la manière suivante [14] :

⇨ les états S de l'automate correspondent aux états internes du malware ;

⇨ l'alphabet Σ est constitué des données capturées à savoir les appels système ;

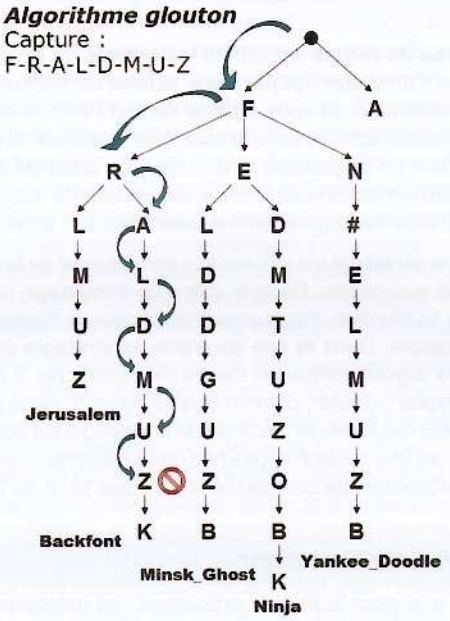
⇨ les fonctions de transition T représentent les séquences d'appels reconnues comme suspicieuses ;

⇨ l'état initial s_0 correspond au début de l'analyse ;

⇨ les états d'acceptation A signalent la détection du comportement surveillé.

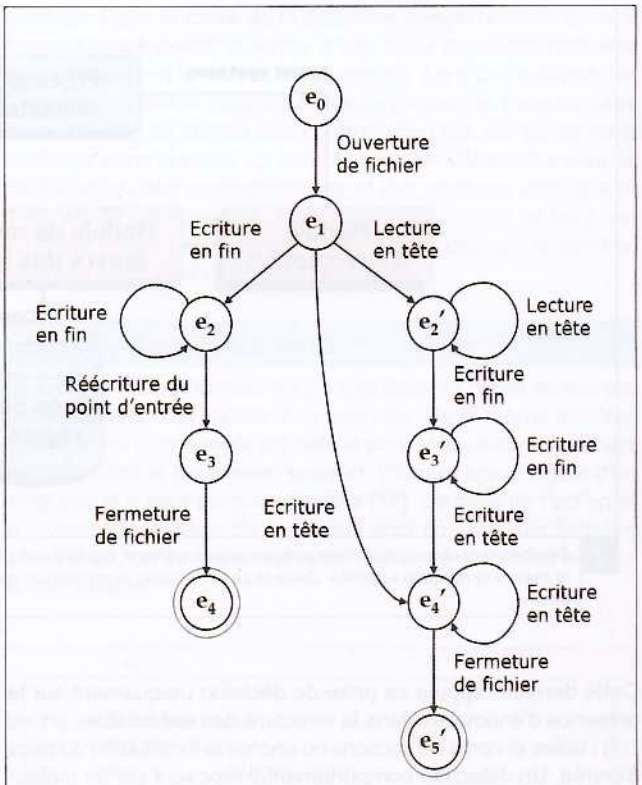
Depuis un état initial, l'automate progresse au fur et à mesure qu'il évalue les données capturées. Le passage à un état d'acceptation trahit la présence d'un comportement malicieux. Dans le cas contraire

F = Accès fichier suspicieux
R = Relocalisation
A = Allocation mémoire suspicieuse
N = Mauvaise extension
# = Routine de déchiffrement
E = Point d'entrée flexible
L = Trappe au chargement de programme
D = Ecriture disque en accès direct
M = Code résident en mémoire
T = Estampille temporelle incorrecte
G = Code mort
Z = Détermination d'EXE/COM
O = Ecrasement/relocalisation de programme
B = Retour au point d'entrée
K = Structure de pile suspicieuse



5 *Règles et stratégie de détection. La construction de l'arbre de recherche repose sur les règles de détection de cinq souches virales contenues dans la base du moteur TBScan [12]. La stratégie choisie est un simple algorithme glouton, à savoir que le chemin pris est toujours le premier possible sans retour en arrière. Dans cet exemple, la détection de Backfont échoue, car le comportement K est manquant. Une stratégie autorisant des retours aurait permis de détecter le virus Jérusalem en ignorant le flag A.*

où l'automate progresse de manière ininterrompue jusqu'à la fin de la séquence capturée ou bien rencontre une transition non supportée, le comportement est supposé légitime. La figure 6 fournit un exemple de modélisation du mécanisme d'infection.



6 *Automate du mécanisme d'infection. Les deux principaux types d'infections de fichier sont représentés ici. Celles de type append sont modélisées par la branche gauche de l'automate où le malware est recopié en fin et le point d'entrée du programme est modifié pour prendre la main à l'exécution. Celles de type prepend, modélisées par la branche droite, recoupent deux cas de figure : une forme non destructive recopiant d'abord le code original en fin de fichier et une forme destructive avec le saut direct à l'écriture du code malveillant.*

3.3 Génération des bases comportementales

Au cours des deux parties précédentes, nous nous sommes focalisés sur les étapes du processus de détection, mais un dernier aspect transversal reste à aborder. La détection repose finalement sur une connaissance approfondie du mode opératoire des malwares. Il est donc crucial de définir son processus de génération.

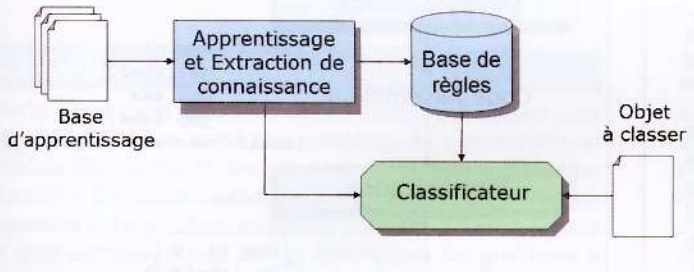
3.3.1 Génération manuelle

Que ce soit les règles d'un système expert, l'arbre d'un moteur heuristique ou encore les transitions d'un automate, ces modèles comportementaux restent, dans la majorité des cas, générés manuellement. L'avantage par rapport à un schéma de signature classique réside dans la généricité des modèles générés. Ils sont plus résistants aux tentatives de modifications syntaxiques qui ne modifient finalement pas l'utilisation des services du système. La base requiert alors des mises à jour moins régulières, car l'introduction de comportements viraux novateurs est assez rare. Cette mise à jour reste malgré tout nécessaire contrairement à ce que peuvent proclamer certains discours marketing.

Cette génération peut se faire à partir de deux sources. Soit grâce à un expert qui, fort de son expérience, définit ses règles de manière opaque et générique dans une optique d'interopérabilité. Soit cette responsabilité est déléguée à l'utilisateur. Il est alors libre de définir sa propre politique et d'adapter de manière plus pointue les règles à son environnement. En contrepartie, il doit être sensibilisé aux problèmes de sécurité et surtout posséder une certaine expérience afin d'appréhender les répercussions possibles de ses choix.

3.3.2 Apprentissage automatique

La génération automatique des modèles comportementaux est une amélioration critique pour augmenter de manière significative la réactivité des produits antiviraux. Jusqu'à maintenant, les techniques d'apprentissage n'ont pu être combinées avec tous les moteurs existants, étant donné la complexité des structures manipulées, mais le principe est prometteur. Techniquement, un mécanisme d'apprentissage repose sur la mise en œuvre de classifieurs combinés avec des techniques de fouille de donnée. Quel qu'en soit le type, la procédure générale reste la même que celle exposée schématiquement à la figure 7. Le système est dans un premier temps confronté à un panel constitué de différents malwares et de programmes sains préalablement étiquetés. Durant son apprentissage, le classifieur extrait, parmi les données capturées, des propriétés communes aux différentes classes de programme. Le choix d'un panel suffisamment conséquent, représentatif et non biaisé est donc critique. Dans le cadre de la détection comportementale, la détermination de ces propriétés communes reposent principalement sur trois paradigmes :



7 *Système d'apprentissage. Les connaissances sont extraites du panel d'apprentissage, puis intégrées à une base de règles testée par le classifieur. Selon la pertinence des résultats obtenus par rapport à ceux attendus, le processus est répété en réinjectant ces informations. Lorsque la base de règles est stable, le classifieur est opérationnel.*

⇨ Le premier paradigme appelé **induction de règles** détermine des conditions d'appartenance aux différentes classes [15], pouvant s'exprimer sous la forme d'expressions booléennes illustrées figure 8. Au fur et à mesure que le classifieur reçoit les échantillons, il intègre ou supprime certaines caractéristiques des conditions pour en préserver la cohérence.

⇨ Le second paradigme **probabiliste** sert de base à des classifieurs tels que les réseaux bayésiens [15]. Pour chaque caractéristique, la probabilité de sa capture dans une classe de malware donnée est calculée avec des résultats similaires à ceux de la figure 9. Au final, pour traduire un comportement, on ne retiendra que ceux ayant le pouvoir de différenciation le plus important. Un critère de choix sera donc le recouvrement minimal entre les différentes classes, le cas idéal étant bien évidemment l'existence d'une caractéristique présente avec une probabilité de 100% pour une unique classe et une probabilité nulle pour n'importe quelle autre.

⇨ La **clusterisation** est un troisième paradigme reposant sur des cas prédéfinis [16]. Au cours de l'apprentissage, on dresse un profil moyen pour chaque classe de malware. La classification consiste alors à calculer la distance, en termes de différences et de similarités, entre un profil et le programme testé. Il sera alors classé selon le profil avec lequel il présente une distance minimale. La méthode utilisée pour le calcul peut varier d'un système à l'autre, mais reste un facteur prédominant sur la précision du classifieur. La figure 10, page suivante, donne un exemple de méthode de calcul, fonction du nombre de substitutions, d'insertions et de suppressions nécessaires pour passer d'une séquence d'appels à une autre.

$$Ver\ mail ::= Call.Connect() \wedge Call.Send() \wedge (\neg Call.Receive()) \wedge String.HELLO \wedge String.MAIL \wedge String.RCPT$$

8 *Condition d'appartenance à la classe des vers de mails. La règle spécifie les appels système réalisés par cette classe, ainsi que les chaînes de caractères utilisées. La différence avec un client mail légitime réside dans l'absence de réception de données, le ver ne s'intéressant ni à la réception de mail, ni à la vérification de l'envoi des mails infectés.*

$$P(OpenFile \mid Programme\ bénin) = 95\%$$
$$P(OpenFile \mid Infecteur\ de\ fichier) = 100\%$$
$$P(GetModuleHandle \mid Programme\ bénin) = 20\%$$
$$P(GetModuleHandle \mid Infecteur\ de\ fichier) = 70\%$$

9 *Probabilités d'appels système. Ces résultats, donnés à titre d'exemple, illustrent la prépondérance de certaines caractéristiques. Une ouverture de fichier seule ne permet en aucun cas de caractériser un infecteur de fichier. Par contre, l'utilisation de GetModuleHandle, pour accéder à l'image en cours d'exécution dans l'optique d'une recopie, est beaucoup plus significative.*

3.4 Limitations du mode dynamique

Si la surveillance dynamique des comportements présente des avantages certains tels que sa résistance aux techniques de mutations syntaxiques, elle présente néanmoins des limitations de plus en plus évidentes au vu des malwares récents.

Il faut bien voir que le recours aux appels système n'est pas toujours obligatoire. Certains comportements, tels que le chiffrement, n'utilisent pas les ressources du système pour des raisons de furtivité. Les virus vont parfois jusqu'à redéfinir complètement des primitives système susceptibles d'être surveillées, rendant ainsi la capture inefficace.

Un autre phénomène est la migration des malwares, en particulier les *rootkits*, vers le noyau du système. Elle s'explique par le fait que les

Opération	Profil	Capture	Coût
Insérer		RegWriteKey	1,5
*	WriteFile	WriteFile	0
Supprimer	CreateProcess		0,7
*	RegWriteKey	RegWriteKey	0
Remplacer	RegWriteKey	RegWriteKey	0,1
Remplacer	Send	Receive	2,0
Distance			4,3

10 *Distance entre deux traces. Le tableau permet de comparer les séquences d'appels d'un profil et d'une capture. La distance correspond à l'addition des coûts associés à chaque opération nécessaire.*

malwares s'exécutant dans un contexte noyau et non plus utilisateur possèdent des privilèges égaux à ceux des antivirus permettant ainsi des techniques anti-antivirales avancées [5, p. 188]. À ce niveau d'exécution, les malwares peuvent interagir directement avec le matériel et les objets système sans passer par les appels système surveillés.

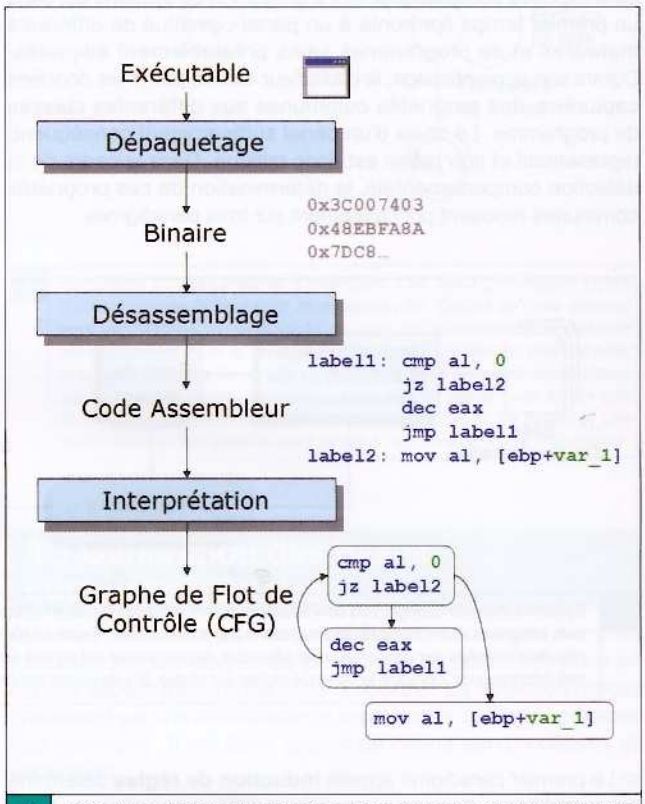
La dernière limitation n'est plus simplement liée à des problèmes techniques relatifs à la capture, mais liée à des propriétés inhérentes aux approches par simulation. En effet, la procédure explore uniquement le chemin d'exécution courant, ce qui n'est pas sans poser des problèmes de couverture de la détection. Le risque est de manquer certains comportements conditionnés. Ce que nous entendons par « conditionnés », ce sont les actions exécutées sous certaines conditions particulières, telles que l'absence d'un environnement émulé ou selon le résultat d'un générateur d'aléas. Une solution possible serait l'émulation de code avec branchements forcés, mais on ne se place alors plus totalement dans une approche boite noire.

4. Particularités du mode statique

De manière plus récente, certains chercheurs ont commencé à envisager l'étude statique du comportement des malwares. Bien que ces travaux ne soient pas officiellement rattachés à la détection comportementale, nous avons pris le parti de les présenter, étant donné qu'ils reposent également sur l'identification des actions, abordant ainsi la notion de « comportement ». Contrairement au mode dynamique, l'approche est plutôt celle d'une analyse boîte blanche, à mettre en parallèle avec la vérification formelle de programme. La détection devient typiquement un problème de bisimulation, à savoir vérifier l'équivalence entre la description d'un programme et la spécification d'un comportement malicieux. Nous allons maintenant décrire les particularités de ce processus de détection qui reste découpé selon les trois mêmes étapes.

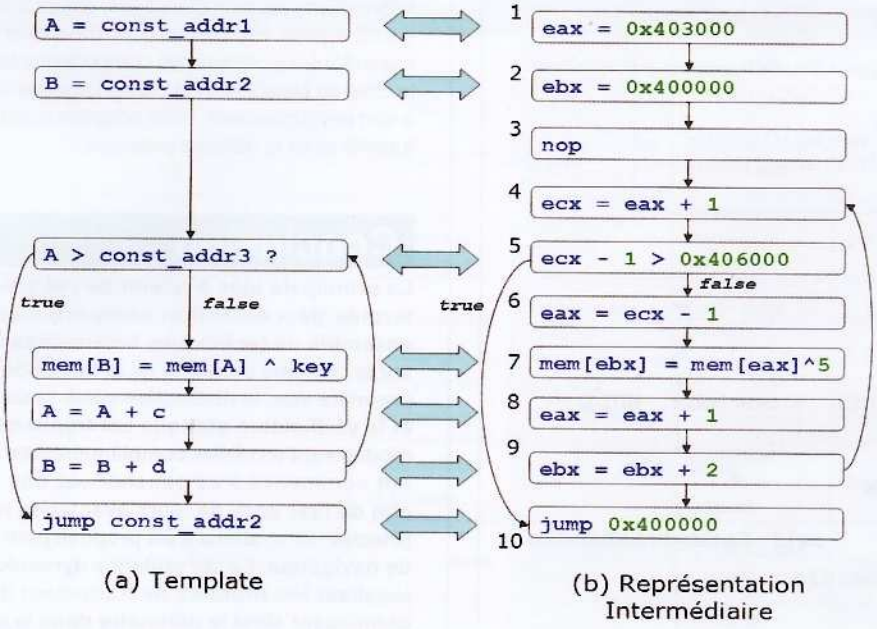
4.1 Prétraitement et extraction des données

La nature des données en entrée de l'analyse statique diffère de l'analyse dynamique par sa plus grande richesse. Il n'est en effet plus nécessaire de se limiter aux événements observables. En contrepartie, l'extraction des données est plus complexe nécessitant plusieurs phases de traitement pour obtenir la représentation intermédiaire d'un programme. La figure 11 illustre l'extraction d'un graphe de flot de contrôle (CFG) depuis un fichier exécutable. Ce type de description est majoritairement utilisé, car il permet de représenter facilement les différents chemins d'exécution potentiels pour un programme.



11 *Extraction statique de données. Ce schéma illustre les différents traitements appliqués au programme. La première phase n'est nécessaire que lorsque le programme a été packagé en utilisant des outils tels qu'UPX. Le binaire est ensuite désassemblé et traduit vers une représentation intermédiaire.*

Pour que cette extraction reste possible, il faut que le désassemblage le soit également, ce qui n'est pas toujours évident. Certaines techniques de protection logicielle peuvent fausser le désassemblage par perte de l'alignement des instructions. S'ils n'interdisent pas la construction de cette représentation, les techniques d'obfuscation peuvent également être déployées pour la complexifier.



12 *Équivalence par isomorphisme. Le template (a) représente ici de manière générique un simple chiffrement par XOR entre deux adresses. Lors de la vérification, on associe chaque nœud du template (a) avec un nœud potentiellement équivalent de l'instance (b). Une fois la correspondance établie, on vérifie la préservation des variables. Dans l'exemple présent, il s'agit de vérifier que la valeur du registre eax liée à A au nœud 1, moment de sa définition, est identique à celle de ecx au nœud 5, moment de son utilisation.*

4.2 Détection par vérification

Dans le cadre de la détection, la vérification est utilisée pour tester l'équivalence entre des modèles de comportement prédéfinis plus ou moins abstraits et la description extraite d'un programme. De manière générale, prédire le comportement d'un programme est équivalent au *halting problem*, à savoir, prédire son arrêt connaissant sa simple description. Ce problème a malheureusement été prouvé indécidable pour la première fois par Alan Turing en 1936. Néanmoins, selon les conditions définissant l'équivalence, il existe des algorithmes permettant de résoudre en partie le problème.

4.2.1 Isomorphisme de graphes annotés sémantiquement

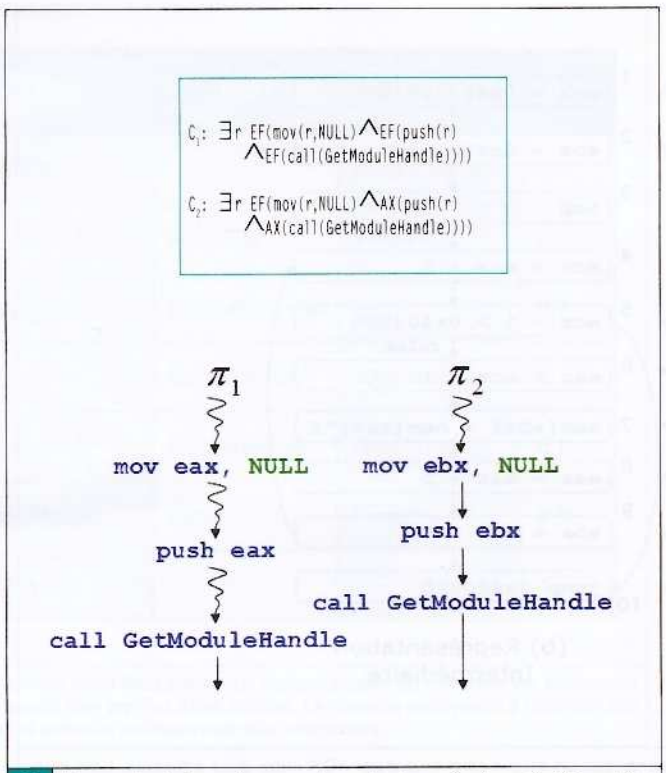
Ce type d'analyse, comme son nom le laisse présupposer, repose sur les CFG mentionnés précédemment. Concernant l'annotation, les instructions contenues dans les nœuds sont traduites dans une représentation intermédiaire. Cette représentation repose sur une sémantique particulière utilisant bien souvent des constantes et variables symboliques pour des raisons de généricité et de résistance aux mutations. Un comportement est alors spécifié par un *template* qui n'est autre qu'un graphe annoté dans la sémantique choisie [17], comme le montre l'aperçu de la figure 12 (a).

La vérification consiste alors en un problème d'isomorphisme de sous-graphe. L'algorithme cherche, dans le graphe représentant le programme, un sous-graphe dont les sommets peuvent être associés avec ceux du template. Une contrainte supplémentaire étant que cette association n'est possible que si les annotations sont cohérentes comme l'illustre la figure 12. Dans un deuxième temps, pour une meilleure précision, il est possible de vérifier que l'exécution de ces deux graphes présente un effet identique sur la mémoire. L'association établie entre les variables, constantes symboliques et les éléments réels comme les registres ou cases mémoire, permet de vérifier la préservation des valeurs depuis leur définition jusqu'à leur utilisation. Cette deuxième étape n'est pas toujours déployée, car elle augmente dramatiquement la complexité.

4.2.2 Vérification de modèles logiques

Pour ce type de vérification, un comportement sera spécifié par une formule de logique temporelle proche des modèles Booléens [18]. Un exemple succinct de formule est détaillé à la figure 13, page suivante ; pour plus d'information, il peut être intéressant de se référer à la littérature traitant du *model checking*. Comme précédemment, les registres, constantes et variables sont désignés sous des valeurs symboliques afin de faire preuve de généricité.

Classiquement, un algorithme de vérification prend en entrée un CFG ainsi qu'une ou plusieurs formules logiques et renvoie l'ensemble des états intermédiaires dont les chemins d'exécution satisfont la formule. Ce type d'algorithme est fortement récursif, puisqu'il explore de manière énumérative les différents chemins d'exécution possibles. Au cours de son déroulement, il est capable de lier les valeurs génériques à des registres ou variables réels et d'en conserver la trace.



13 *Expression logique de comportements. Les opérateurs A et E sont des quantifieurs de chemin tandis qu'X et F sont des opérateurs temporels. Par exemple, la combinaison EF(p) signifie qu'il existe un chemin d'exécution vérifiant le prédicat p lors d'un état futur indéterminé. La condition C_1 signifie donc qu'il existe un chemin où l'on observera l'affectation de la valeur NULL à un registre, sa poussée en sommet de pile ainsi qu'un appel à GetModuleHandle sans pour autant que ces actions ne soient nécessairement contiguës. En remplaçant les opérateurs EF par AX dans la condition C_2, on impose que pour tout chemin (et non plus au minimum un) présentant une affectation de NULL à un registre, les actions la suivant immédiatement seront obligatoirement la poussée de cette valeur sur la pile et l'appel.*

4.3 Complémentarité du mode statique

Contrairement au mode dynamique, le mode statique reste très sensible aux techniques de mutations et, en particulier, au métamorphisme qui utilise des techniques évoluées d'obfuscation. Ces techniques ont pour effet de complexifier fortement les représentations extraites des programmes, atténuant ainsi les similitudes avec la spécification. Néanmoins, les deux techniques que nous avons développées, démontrent une certaine résistance à ces manipulations de code. L'utilisation préalable de techniques d'optimisation, utilisées habituellement en compilation, permet autant que possible de réduire la structure du CFG extrait en supprimant le code mort inséré, les permutations et les branchements par prédicats opaques. En outre, la traduction vers une représentation sémantique intermédiaire, possible également avec les modèles logiques, bien que ce n'ait pas été présenté, résout les substitutions d'instructions les plus simples. En particulier, l'utilisation de variables symboliques insensibilise le détecteur aux réaffectations de registres. Dans le cas de substitutions de fonctionnalités plus complexes, ces mesures ne sont, par contre, plus suffisantes.

En contrepartie, la vérification statique ne souffre pas des limitations propres à la surveillance dynamique. Cette méthode apporte les informations les plus complètes, car tous les chemins d'exécution y sont visibles. Elle permet donc d'identifier l'ensemble des actions malveillantes y compris les comportements conditionnés. De plus, la cible de l'analyse n'est plus en mesure d'adapter son exécution à son environnement, cette adaptation étant une des bases de la furtivité et de la défense proactive.

Conclusion

La principale idée à retenir de cet article est que, sous les termes de « détection comportementale », se cache un ensemble de techniques hétéroclites reposant malgré tout sur le principe commun de la détection de fonctionnalités. À première vue, la distinction entre la surveillance dynamique et la vérification statique est frappante. Pourtant ces deux modes sont en réalité complémentaires. Plusieurs chercheurs ont commencé à se pencher sur une utilisation combinée afin de tirer parti de leurs avantages respectifs [19]. Sur ce principe, un système a été proposé pour détecter les spywares de navigateur. La surveillance dynamique y est utilisée pour localiser les routines de traitement des événements web, définissant ainsi le périmètre dans le code où déployer une analyse statique.

Un second point qu'il est important de rappeler est que si la détection comportementale permet de détecter certains virus inconnus n'utilisant pas de concepts novateurs, elle reste néanmoins sujette à des taux d'erreurs importants. Récemment, une méthodologie de test fondée sur la mutation manuelle de fonctionnalités a été proposée afin d'évaluer la couverture des moteurs comportementaux [20]. Les résultats ont montré que pour pallier les erreurs et identifier les menaces, la plupart des produits antiviraux confrontent l'analyse comportementale aux scanners de signature. Si cette mesure diminue ostensiblement le nombre de faux positifs, elle bride en contrepartie bien souvent la détection comportementale, empêchant la détection de nouvelles souches aux fonctionnalités très proches.

Avant de terminer, le tableau 1 donne un panorama intéressant de l'avancée des différentes techniques et des tendances actuelles. Sont intégrés à cette table non seulement des prototypes de recherche, mais aussi certains moteurs commerciaux, d'après les informations communiquées. Deux principales tendances s'y dégagent avec, d'un côté, les moteurs heuristiques et, de l'autre, les systèmes experts en temps réel. Concernant ces derniers, il est intéressant de noter que la distinction entre les antivirus les déployant et les Systèmes de Prévention d'Intrusion Hôte (HIPS) s'avère aujourd'hui impossible. Les autres techniques présentées ne s'avèrent malheureusement pas suffisamment opérationnelles pour être déployées commercialement.

Références

(suite, page suivante)

[01] Évaluation Virus.gr de produits antivirus : **http://www.virus.gr/english/fullxml/default.asp?id=82&mnu=82**.

Nom (Origine)	Sortie	Mode d'action	Données en entrée	Cible	Type de moteur	Utilisation	Envment.
TBScan N/C	1994	Dynamique (SB)	Interruptions	Infecteurs de fichier	Heuristique (flags)	Détection	Ms DOS
VIDES Univ. Namur & Hambourg	1995	Dynamique (RT)	Interruptions	Infecteurs COM et EXE	Automates finis déterministe	Dét./Class.	Ms DOS
N/C Univ. Columbia & N.Y.	2003	Statique	Fonctions importées et chaînes	Tout type de malware	Fouille de données et classifieur	Détection	Win
GateKeeper Institut Technique Floride	2004	Dynamique (RT*)	Appels système	Tout type de malware	Heuristique (poids)	Détection	Win
N/C Univ. Camegie et al.	2005	Statique	Graphe de flot de contrôle	Vers mail polymorphes	Isomorphisme	Détection	Win
N/C Univ. Munich	2005	Statique	Graphe de flot de contrôle	Vers	Vérification de modèles logiques	Détection	Win
TTAnalyze Univ. Technique Vienne	2006	Dynamique (VM)	Appels système	Tout type de Malware	Simple rapport d'activité	Classification	Win
N/C Microsoft Corp.	2006	Dynamique (VM)	Appels système	Tout type de Malware	Fouille de données et classifieur	Classification	Win
N/C Univ. Californie & Vienne	N/C	Dynam./Statique	Appels COM et système	Spywares de navigateu	Système expert	Détection	Internet Explorer
ThreatSense (NOD32) Eset	N/C	Dynamique (SB)	Actions composées d'instructions	Tout type de Malware	Heuristique	Détection	Win/Linux/ FreeBSD
AVG Anti-Virus Grisoft	N/C	Dynamique (SB)	Actions composées d'instructions	Tout type de Malware	Heuristique	Détection	Win/Linux/ FreeBSD
ViGUARD Softed	N/C	Dynamique (RT)	Appels système	Tout type de Malware	Système expert (décision de l'utilisateur)	Détection	Win
B-HAVE (Bit Defender) Softwin	N/C	Dynamique (SB)	Actions composées d'instructions	Tout type de Malware	Heuristique	Détection	Win/Linux/ FreeBSD
Bloodhound (Norton AV) Symantec	1997	Dynamique (SB)	Actions composées d'instructions	Infecteurs de Fichier	Heuristique	Détection	Win
Entercept Mc Affee	2004	Dynamique (RT)	Appels système	Tout type de Malware	Système expert (politique prédéfinie)	Détection	Win/Linux
Safe'n'Sec Antivirus Safen Soft	2004	Dynamique (RT)	Appels système	Tout type de Malware	Système expert (politique prédéfinie)	Détection	Win/Linux/ FreeBSD
TruPrevent Panda Software	2006	Dynamique (RT)	Appels système	Tout type de Malware	Heuristique	Détection	Win/Linux
Virus Keeper AxBa	2007	Dynamique (RT)	Appels système	Tout type de Malware	Système expert (décision de l'utilisateur)	Détection	Win

RT = Temps Réel, VM = Machine Virtuelle, SB = Sandbox

* Capture des actions pour annulation

TI *Panorama de la détection comportementale. La première partie du tableau recoupe les différents travaux de recherche tandis que la deuxième propose un aperçu des techniques actuellement déployées opérationnellement.*

Références

[02] FILIOL (E.), *Les Virus Informatiques : Théorie, Pratique et Applications*, Éditions Springer, collection IRIS, ISBN: 2-287-20297-8 – 2003.

[03] Observatoire Fortinet : **http://www.fortinet.com/FortiGuardCenter/antivirus/wildlist.html**.

[04] FILIOL (E.), « *Malware Pattern Scanning Schemes Secure Against Black-box Analysis* », *Journal in Computer Virology*, Volume 2, Issue 1, p. 35-50, Springer, 2006.

[05] FILIOL (E.), *Techniques Virales Avancées*, Éditions Springer, collection IRIS, ISBN : 2-287-33887-8, 2007.

[06] SPINELLIS (D.), « *Reliable Identification of Bounded-length Viruses is NP-complete*, IEEE Transactions on Information Theory », Volume 49, Issue 1, p. 280-284, 2003.

[07] COHEN (F.), « *Computer Viruses: Theory and Experiments* », *Computers & Security*, Volume 6, Issue 1, p. 22-35, 1987.

[08] DEBAR (H.), DACIER (M.), WESPI (A.), « *Towards a Taxonomy of Intrusion-detection Systems* », *Computers Networks*, Volume 31, Issue 9, p. 805-822, 1999.

[09] WARRENDER (C.), FORREST (S.), PEARLMUTTER (B.), « *Detecting Intrusion Using System Calls: Alternative Data Models* », Proceedings of IEEE Symposium on Security and Privacy, ISBN: 0-7695-0176-1, p. 133-145,1999.

[10] WAGNER (M. E.), *Behavior Oriented Detection of Malicious Code at Run-Time*, Thesis for the Master of Science Degree, Florida Institute of Technology, 2004.

[11] BAYER (U.), (KRUEGEL C.), KIRDA (E.), « *TTAnalyze: A Tool for Analyzing Malware* », Proceedings of EICAR, 2006.

[12] DEBBABI (M.), « *Dynamic Monitoring of Malicious Activity in Software Systems* », Symposium on Requirements Engineering for Information Security (SREIS'01), USA, 2001.

[13] VELDMAN (F.), « *Heuristic Anti-Virus Technology* », Proceedings of the International Virus Protection and Information Security Council, 1994.

[14] LE CHARLIER (B.), MOUNJI (A.), SWIMMER (M.), « *Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns* », Proceedings of the 5th Virus Bulletin Conference, 1995.

[15] SCHULTZ (M. G.), ESKIN (E.), ZADOK (E.), « *Data Mining Methods for Detection of New Malicious Executables* », Proceedings of IEEE Symposium on Security and Privacy, ISBN : 0-769-51046-9, p. 38-49, 2001.

[16] LEE (T.), MODY (J.), « *Behavioral classification* », Proceedings of EICAR, 2006.

[17] CHRISTODORESCU (M.), JHA (S.), SESHIA (S. A.), SONG (D.), BRYANT (R. E.), « *Semantic-Aware Malware Detection* », Proceedings of IEEE Symposium on Security and Privacy, p. 32-46, 2005.

[18] KINDER (J.), KATZENBEISSER (S.), SCHALLHART (C.), VEITH (H.), « *Detecting Malicious Code by Model Checking* », Lecture Notes in *Computer Science*, Volume 3548, p. 174-187, Springer, 2005.

[19] KIRDA (E.), KRUEGEL (C.), BANKS (G.), VIGNA (G.), KEMMERER (R.), « *Behavior-based Spyware Detection* », Proceedings of the 15th USENIX Security Symposium, 2006.

[20] FILIOL (E.), JACOB (G.), LE LIARD (M.), « *Evaluation methodology and theoretical model for antiviral behavioural detection strategies* », *Journal in Computer Virology*, Volume 3, Issue 1, Springer, 2007.

MISC
est édité par Diamond Editions
B.P. 20142 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : miscabo@ed-diamond.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler

Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor

Conception graphique :
Kathrin Troeger

Secrétaire de rédaction :
Véronique Wilhelm

Relecteurs :
Dominique Grosse
Cédric Blancher - sid@rstack.org
Thierry Martineau - thierrymartineau@yahoo.fr

Responsable publicité :
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression : I. D. S. Impression (Sélestat)

Distribution :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

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

Dépôt légal : 2e Trimestre 2001
N° ISSN : 1631-9036
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 8 euros

Imprimé en France
Printed in France

CHARTE

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.