hakin9

Les systèmes de détection d'intrusion vus de l'intérieur

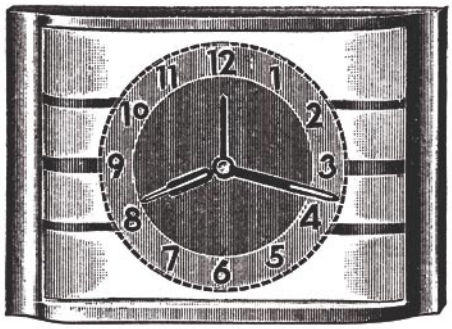
Antonio Merola

Article publié dans le numéro 4/2005 du magazine *hakin9*

Magazine *hakin9*, Software – Wydawnictwo, ul. Piaskowa 3, 01-067 Varsovie, Pologne, *fr@hakin9.org*

Les systèmes de détection d'intrusion vus de l'intérieur

Antonio Merola



L'objectif des systèmes de détection d'intrusion consiste à identifier des attaques ou des violations de sécurité issues du réseau de surveillance et des activités hébergées. Afin de bien comprendre le fonctionnement de tels systèmes, il est nécessaire de présenter la technologie qui régit les systèmes de détection d'intrusion.

Un système de détection d'intrusion peut être comparé à une alarme domestique contre les cambrioleurs – si une tentative d'intrusion malveillante est découverte, une réponse de quelque nature qu'elle soit sera alors déclenchée. Plus de capteurs sont paramétrés, meilleur est le système, puisque chaque capteur est chargé de détecter un type particulier d'activité (telle que l'ouverture des portes ou des fenêtres, une détection volumétrique etc.). Toutefois, à l'instar de tous les autres systèmes automatiques, un système de détection d'intrusion peut présenter des failles, émettre de fausses alertes ou être contourné par des techniciens hautement qualifiés sur ce genre de matériel.

Le premier système de détection d'intrusion a vu le jour au début des années 80 ; il s'agissait d'un projet de recherche mené par le gouvernement américain et certaines organisations militaires. La technologie mise au point a rapidement évolué en une dizaine d'années et à la fin des années 90, des solutions commerciales sont apparues sur le marché. Depuis lors, de nombreux produits et ressources ont fait l'objet de recherches et un nouveau poste informatique a été ainsi créé – *analyste d'intrusion*.

C'est dans le domaine de l'audit que remonte l'origine des systèmes de détection d'intrusion, comme le relate l'ouvrage parfaitement documenté intitulé *A Guide to Understanding Audit in Trusted Systems* (publié sous la collection *Rainbow Series* du ministère de la Défense des États-Unis), et également connu sous le titre de *The Tan Book*. Les auteurs de cet ouvrage définissent l'audit comme *un contrôle et une révision indépendants des activités et des*

Cet article explique...

- la nature des systèmes de détection d'intrusion,
- la façon de contourner les solutions proposées par les systèmes de détection d'intrusion,
- la façon de se protéger contre les fuites de tels systèmes.

Ce qu'il faut savoir...

- vous maîtrisez idéalement les bases du protocole HTTP,
- vous devez connaître le fonctionnement de base des protocoles TCP/IP,
- vous êtes censé savoir utiliser la couche système UNIX et Windows.

Science

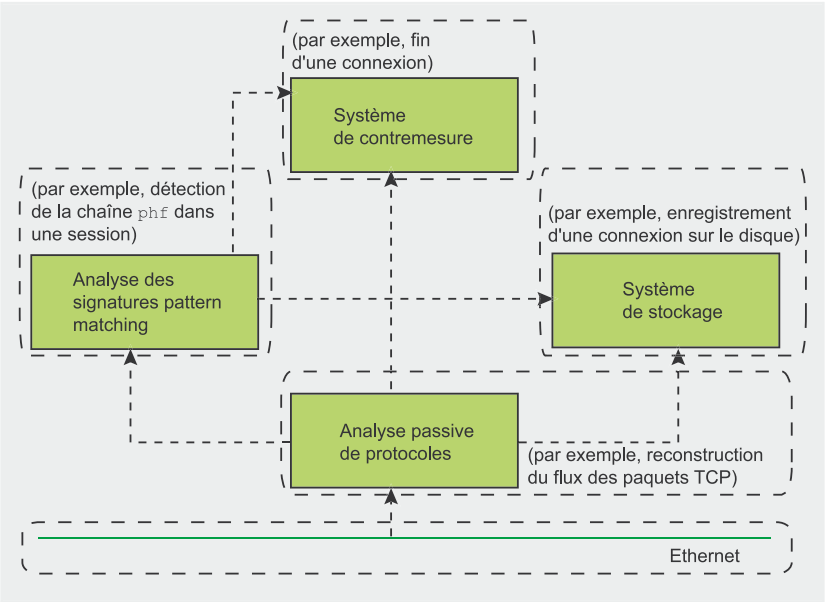


Figure 1. *Modèle CIDF (Common Intrusion Detection Framework) d'un système de détection d'intrusion au niveau du réseau*

enregistrements d'un système. En règle générale, l'audit permet de remonter à la source des événements et de découvrir des activités illégales.

James Anderson fut le premier à développer ces systèmes de détection d'intrusion naissants pour le compte de l'armée de l'air américaine. Dans sa publication scientifique, il a décrit comment restreindre le champ des audits uniquement aux conditions de sécurité pertinentes et comment discerner les activités normales des illégales. Plus tard eut lieu la démonstration publique du *Network System Monitor*, système élaboré à l'Université de Californie. Ce système était capable de détecter des intrusions, et de présenter les corrélations existantes entre les activités anormales et une mauvaise utilisation de l'ordinateur.

Les solutions de détection d'intrusion

Il existe trois approches différentes en matière de détection d'intrusion. La première est le système de détection d'intrusion au niveau du réseau ou *Network Intrusion Detection System* (NIDS), largement utilisé, chargé d'analyser de manière passive le trafic réseau, et de chercher les activités malveillantes. La deuxième approche est le système de détection d'intrusion au niveau des hôtes ou *Host Intrusion Detection System* (HIDS), fonctionnant sur un hôte surveillé pour y détecter des intrusions. Enfin, la troisième approche, et la moins utilisée, est le système de détection d'intrusion au niveau des nœuds du réseau ou *Network Node Intrusion Detection System* (NNIDS) – solution hybride regroupant les caractéristiques des NIDS mentionnés plus haut, tout en n'analysant qu'une partie (ou nœud) du trafic réseau. Les approches de détection comprennent les tâches de contrôle de l'activité du réseau en cherchant des modèles particuliers (ou signatures) afin de réaliser une analyse statistique sur cette activité pour déterminer si les données ont été modifiées ou non. La Figure 1 permet de mieux comprendre le type d'analyse réalisée.

Par *Common Intrusion Detection Framework* (CIDF), on désigne l'ensemble des composants qui définissent un système de détection d'intrusion (l'objectif consiste à créer un modèle de conception pour ces systèmes). Ces composants comprennent la génération d'événements, les modules d'analyse, les mécanismes de stockage et même les contre-mesures. La plupart des systèmes de détection d'intrusion se contentent de chercher des signes d'attaques connues, appelés signatures, comme par exemple les définitions antivirus. La principale

Snort, fer de lance de la détection d'intrusion

Snort est un outil libre développé en 1998 par Martin Roesch de l'équipe Sourcefire. Aujourd'hui, des entreprises, des universités, des agences gouvernementales etc. du monde entier ont recours à cet outil – la documentation de *Snort* est disponible en plus de 10 langues. La version la plus récente, sortie en mai 2005 est *Snort* 2.3.3, téléchargeable à partir du site *http://www.snort.org*.
Snort peut être configuré afin de fonctionner selon trois modes principaux :

- mode sniffeur,
- mode enregistreur de paquets de données,
- mode système de détection d'intrusion réseau.

Le dernier mode est de loin le plus complexe et le plus difficile à configurer. Ce logiciel est chargé d'analyser le trafic réseau selon des règles définies et de réaliser certaines actions (comme par exemple déclencher une alerte). La page consacrée au manuel d'utilisation de *Snort* ainsi que les données de sortie de la commande `snort -?` contiennent les informations nécessaires pour lancer l'outil en différents modes. Par exemple, activer le mode système de détection d'intrusion réseau revient à taper les éléments suivants :

```
# snort -dev -l ./log \
 -h 10.10.10.0/24 -c snort.conf
```

où le dernier fichier indiqué représente le nom de notre fichier de règles. Chaque paquet de données sera ainsi contrôlé afin d'y trouver une règle correspondante ; si tel est le cas, une action est alors déclenchée.
Quelques exemples de signatures sont exposés dans le Tableau 1.

Table 1. *Exemple de signatures du logiciel Snort*

Type de signature	Chaîne
Signature d'en-tête	`alert tcp any any -> $HOME_NET any ↩ (flag: SF; msg: "SYN-FIN scan")`
Signature pattern matching	`alert udp $EXTERNAL_NET any -> ↩ $HOME_NET 53 (msg: "DNS named ↩ version attempt"; content:"|07|version")`
Signature basée sur protocole	`preprocessor: http_decode 80`
Signature à base d'heuristique	`alert icmp any any -> $HOME_NET any ↩ (msg: "Large ICMP packet"; dsize > 800)`

différence réside dans les nombres ; un système de détection d'intrusion peut contrôler approximativement 200 000 signatures, contre seulement 15 000 signatures vérifiées par un système d'antivirus classique.

Comment les données sont-elles contrôlées ? Il existe des *signatures d'en-tête* moins précises dans lesquelles les systèmes de détection d'intrusion cherchent des champs particuliers contenus dans des en-têtes de paquets de données – ils cherchent par exemple dans le paquet le port de destination TCP 80 (on parle de filtrage de paquet de données sans état). Il existe également des *signatures pattern matching* plus élaborées. Dans ce cas, le système de détection d'intrusion cherche une correspondance aux chaînes du contenu sur un seul paquet de données ou sur un train de paquets de données (on parle de filtrage de paquet de données avec état).

Pour être plus précis, il existe également les éléments suivants :

- les *signatures basées sur protocole*, dans lesquelles le système de détection d'intrusion filtre les données pour contrôler le bon respect des spécifications RFC (*Request For Comment*) ;
- les *signatures à base d'heuristique*, dans lesquelles le système de détection d'intrusion filtre les données par évaluation statistique ;
- les *signatures de détection d'anomalies*, dans lesquelles le système de détection d'intrusion déclenche des alertes en cas de détection d'un trafic anormal.

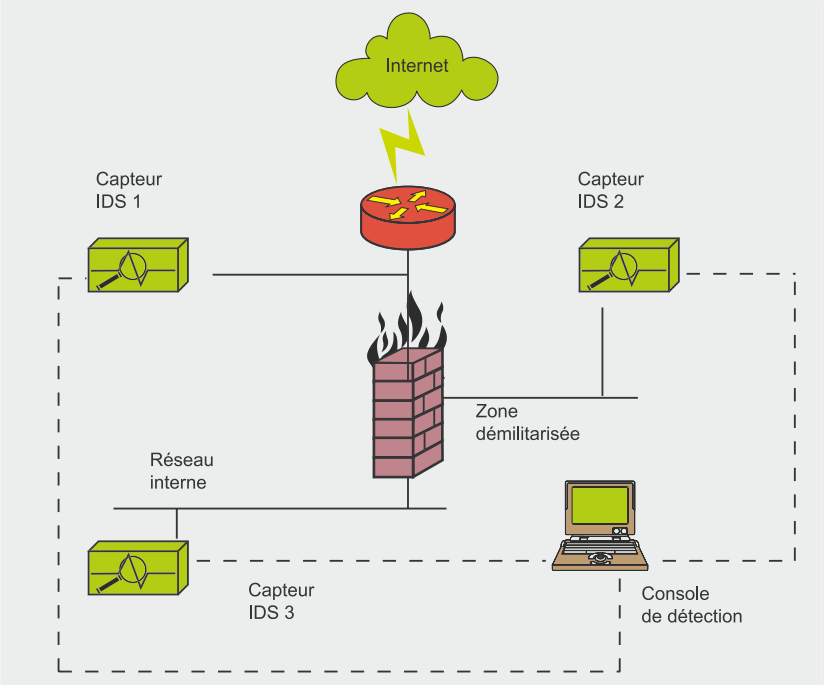


Figure 2. *Système de détection d'intrusion réseau*

Le logiciel de détection d'intrusion le plus connu s'appelle *Snort.* Grâce à des préprocesseurs et à des plug-ins activés, il est capable de traiter toutes les signatures excepté les signatures de détection d'anomalies (voir l'Encadré intitulé *Snort, fer de lance de la détection d'intrusion*).

Il est toutefois probable de rencontrer certains problèmes liés à cette activité de détection d'intrusion. Le premier concerne les alertes – un système doit être préalablement configuré pour cet environnement spécifique afin de minimiser le plus possible le nombre de fausses alertes. Il peut y avoir des fausses alertes *positives* ou *négatives*. Une fausse alerte positive survient lorsqu'un système de détection d'intrusion alerte l'utilisateur d'activités suspicieuses, mais une analyse approfondie prouve que le trafic réseau est normal. Par contre, une fausse alerte négative est déclenchée lorsqu'une activité non autorisée est découverte sans toutefois déclencher d'alerte. La Figure 2 illustre une implémentation classique d'un système de détection d'intrusion réseau.

Contournement de la détection d'intrusion

Les solutions qu'offrent les systèmes de détection d'intrusion sont très pratiques et permettent d'éliminer

À propos de l'auteur

Antonio Merola travaille en tant qu'expert confirmé de la sécurité chez Telecom Italia. Au cours de sa carrière professionnelle, il a été confronté à de nombreux aspects de la sécurité. Son statut d'indépendant lui permet de prodiguer ses services à plusieurs sociétés en tant que consultant et instructeur sur un large choix de thèmes relatifs à la sécurité. Il publie également des articles sur l'informatique dans de nombreux magazines italiens. Parmi ces centres d'intérêts récents, on peut citer les pots de miel ainsi que les solutions de sécurité proposées par les systèmes de détection et de prévention contre les intrusions.

Science

Bogue du Microsoft Indexing Server

Un des exemples les plus spectaculaires de vulnérabilité mentionnées plus haut est le bogue qui a affecté (et affecte encore) l'*Indexing Server* version 2.0 de Microsoft Windows ainsi que l'*Indexing Service* des systèmes Windows NT/2000/XP. Ces dispositifs sont nécessaires lors de l'installation du serveur Web de *Microsoft IIS*.

L'existence de ce bogue est due à l'installation de certaines bibliothèques de liens dynamiques (DLL) exigée par le processus d'installation de *Microsoft IIS*. Une de ces DLL est la bibliothèque partagée *idq.dll* chargée de fournir, entre autre chose, un support pour les scripts de nature administrative (avec une extension en *.ida*). Ce fichier contient une mémoire tampon non contrôlée dans la section chargée de gérer les adresses URL. Dans la mesure où *idq.dll* s'exécute en tant que service du système, l'intrus obtiendra un contrôle total sur le système ainsi attaqué.

Mais pire encore, il n'est pas nécessaire de lancer le processus *idq.dll* pour réussir une attaque. Le pirate n'a besoin que de demander le service de dépouillement. Établir une connexion avec le protocole HTTP puis envoyer une requête préparée en HTTP suffit à garantir le succès d'une attaque.

Un programme correctif complet a été sorti en 2002 (un an après la découverte de ce bogue), malgré l'existence de nombreux serveurs encore vulnérables. En effet, de nombreux administrateurs de Microsoft Windows n'appliquent pas les mises à jours conseillées de sécurité.

une grande majorité de menaces d'attaques. Toutefois, il est possible de contourner la détection d'intrusion par signature. Voici les techniques traditionnelles employées à cet effet :

- obfuscation de code,
- fragmentation,
- refus de service.

Ces méthodes utilisent une activité qui pousse le système de détection d'intrusion à voir un train de données différent de celui du système final, ou désactivent les systèmes de détection de données à l'aide d'attaques DoS (*Denial of Service*).

Obfuscation du code

La plupart des systèmes de détection d'intrusion identifie les attaques au moyen d'une analyse de signature. Cette analyse de signature signifie en réalité que le système de détection d'intrusion est programmé pour interpréter certaines séries de paquets de données ou bien certaines parties de données contenues dans ces paquets comme une attaque. Par exemple, un système de détection d'intrusion chargé de surveiller des serveurs Web pourrait être programmé pour chercher les paquets piégés ; une grande majorité de méthodes fait bien sûr appel au protocole HTTP, mais, toutes les données à base de texte, telles que les requêtes SQL, sont ici impliquées. Ainsi, une requête classique du type *cgi-bin*, par exemple, possède le format HTTP standard suivant :

```
GET /cgi-bin/script.cgi HTTP/1.0
```

Examinons le code suivant :

```
GET /cgi-bin/↩
 quelque_chose_de_dangereux.pl↩
 HTTP/1.0
```

Dans un environnement Web, une double période indique le répertoire parent, alors qu'une période simple représente le répertoire du moment. Le code suivant est alors identique au précédent. Toutefois, un système de détection d'intrusion par signature peut considérer ce code comme deux éléments différents :

```
GET /./././cgi-bin/./././.↩
 quelque_chose_de_dangereux.pl↩
 HTTP/1.0
```

De même, quelqu'un pourrait par exemple taper la requête suivante :

```
GET /cgi-bin/subdirectory/../↩
 quelque_chose_de_dangereux.pl↩
 HTTP/1.0
```

Dans ce cas, nous demandons un sous-répertoire puis utilisons la commande `/../` afin de retourner vers le répertoire parent pour y exécuter les scripts cibles. On appelle cette technique l'attaque par violation de répertoire (en anglais *directory traversal*), et c'est actuellement l'une des méthodes les plus utilisées.

La méthode exposée ci-dessus n'est pas la seule possibilité. L'Unicode est un mécanisme capable de supporter l'ensemble des langues du monde entier. Un serveur Web supportant l'Unicode remplacera correctement la valeur Unicode par le caractère correspondant.

Pour le serveur Web, cette exemple de chaîne de caractères :

```
../../c:\winnt\system32\cmd.exe
```

et la requête HTTP suivante :

```
%2e%2e%2f%2e%2e%2fc:↩
 \winnt\system32\cmd.exe
```

sont strictement identiques. Toutefois, il est probable qu'un système de détection d'intrusion n'interprète pas les deux déclarations comme étant les mêmes. Le ver appelé *CodeRed* exploite la vulnérabilité du débordement de tampon lors de la gestion des fichiers *.ida* (bogue présent dans le *Indexing Service* de Microsoft, voir à ce sujet l'Encadré intitulé *Bogue du Microsoft Indexing Server*) afin de s'introduire dans le système pour s'y propager. Si vous envoyez une requête codée `%u`, il est probable que vous contourniez certains contrôles du système de détection d'intrusion sur *.ida*. En effet, le caractère `a` peut être codé en tant que `U+0061` sous Unicode. Ainsi, la requête suivante :

```
GET /himom.id%u0061 HTTP/1.0
```

ne génèrera aucune alerte. Ce type de contournement des systèmes de détection d'intrusion est également connu sous le nom de *%u encoding IDS bypass vulnerability* ou vulnérabilité par codage %u pour contourner les systèmes de détection d'intrusion.

Divers outils sont disponibles pour tester les possibilités de contournement des systèmes de détec-

tion d'intrusion, mais le logiciel le plus largement utilisé en la matière est *Whisker* (voir l'Encadré intitulé *Whisker – outil anti-système de détection d'intrusion*).

Considérons maintenant les systèmes de détection d'intrusion au niveau des hôtes ou HIDS (*Host-based Intrusion Detection Systems*). Si nous disposons déjà d'un hôte fragilisé et d'un HIDS, il nous faudra réaliser quelques ajustements afin d'éviter le filtrage des signatures. Tous les systèmes d'exploitation d'aujourd'hui autorisent l'usage des alias du shell et des variables d'environnement. Pour les systèmes *NIX, un exemple assez dangereux ressemblerait à la ligne suivante :

```
# alias list_p=`more /etc/passwd`
```

L'exemple suivant, pour Windows maintenant, serait tout aussi dangereux :

```
C:\> set shell=c:\winnt\system32\cmd.exe
```

Avec un tel hôte et des alias définis correctement, taper le code `list _ p` or `%shell% /c dir c:` ne générera aucune alerte.

Fragmentation

Le problème avec le réassemblage des paquets fragmentés est que le système de détection d'intrusion doit conserver en mémoire le paquet de données puis réassembler intégralement ce paquet avant de le comparer à la chaîne. Le système de détection d'intrusion doit également comprendre comment le paquet sera réassemblé par l'hôte destinataire.

Les techniques les plus courantes de fragmentation sont les suivantes : *chevauchement de fragments* ou *fragment overlap*, *écrasement de fragments* ou *fragment overwrite* et *temporisation de fragments* ou *fragment time-out*.

Chevauchement de fragments

Le chevauchement de fragment survient lorsqu'un hôte, réassemblant une séquence de fragments, indique que l'un des paquets reçus contient un fragment qui écrase des données issues d'un fragment précédent.

Supposons que le premier fragment contienne la chaîne `GET x.idx`, et que le second contienne la chaîne `a?` (voir la Figure 3). Une fois les paquets assemblés, le second fragment écrase le dernier octet du premier fragment. Après le réassemblage des paquets sur l'hôte destinataire, la chaîne complète sera alors `GET x.ida?`. Sous le serveur *Microsoft IIS* ou sous les systèmes Windows dotés du *service d'indexation* activé et avec ce bougue, cette chaîne génèrera un débordement de tampon.

Whisker – outil anti-système de détection d'intrusion

Whisker est un logiciel conçu pour pirater les serveurs Web en contournant les systèmes de détection d'intrusion. Cet outil génère de façon automatique une large variété d'attaques anti système de détection d'intrusion. Cette caractéristique lui a valu le surnom de *anti-IDS* (AIDS). D'un point de vue technique, il s'agit d'un scanner CGI chargé de trouver les vulnérabilités du Web.
Les paramètres suivants sont la cause de certaines méthodes de contournement :

- `-I 1` – mode 1 de contournement des systèmes de détection d'intrusion (codage URL),
- `-I 2` – mode 2 de contournement des systèmes de détection d'intrusion (insertion des chemin des répertoires `/./`),
- `-I 3` – mode 3 de contournement des systèmes de détection d'intrusion (fin prématurée d'une URL),
- `-I 4` – mode 4 de contournement des systèmes de détection d'intrusion (longue URL),
- `-I 5` – mode 5 de contournement des systèmes de détection d'intrusion (faux paramètre),
- `-I 6` – mode 6 de contournement des systèmes de détection d'intrusion (séparation TAB – inutilisable pour NT/IIS),
- `-I 7` – mode 7 de contournement des systèmes de détection d'intrusion (sensibilité à la casse),
- `-I 8` – mode 8 de contournement des systèmes de détection d'intrusion (délimiteur Windows),
- `-I 9` – mode 9 de contournement des systèmes de détection d'intrusion (division de session – assez lent),
- `-I 0` – mode 0 de contournement des systèmes de détection d'intrusion (méthode NULL).

Whisker dispose d'une autre méthode de contournement très pratique appelée *division de session* (*session splicing*). Cette méthode permet de diviser la chaîne en plusieurs paquets de données à la fois de manière à empêcher l'appariement des chaînes. Par exemple, si nous voulons envoyer la chaîne `GET /`, Whisker va la diviser en cinq paquets contenant respectivement : `G`, `E`, `T`, `20` (représentation hexadécimale du caractère espace) et `/`.

Afin d'éviter de se faire piéger par ces techniques, le système de détection d'intrusion devra comprendre entièrement la session, tâche difficile et lourde à gérer pour le processeur. La règle suivante de l'outil *Snort* détecte un trafic dit *Whisker* destiné au port 80 avec un ensemble de drapeaux ACK, un espace (`0x20`) dans les données utiles et un paramètre `dsize` de 1 (intercepte les deux premiers octets) :

```
alert tcp $EXTERNAL_NET any ->←
 $HTTP_SERVERS 80 (msg:←
 "WEB-MISC whisker space splice attack");←
 content:"|20|"; flags:A+; dsize:1;←
 reference:arachnids,296;←
 classtype:attempted-recon; reference
```

Toutefois, il faut bien être conscient que cette méthode est facilement modifiable dans le but de contourner un système de détection d'intrusion.

Listing 1. *Signature par défaut de Snort pour détecter le débordement de tampon .ida*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 ←
 (msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; ←
dsize:>239; flags:A+; reference:arachnids,552; ←
 classtype:web-application-attack; ←
reference:cve,CAN-2000-0071; sid:1243; rev:2;)
```

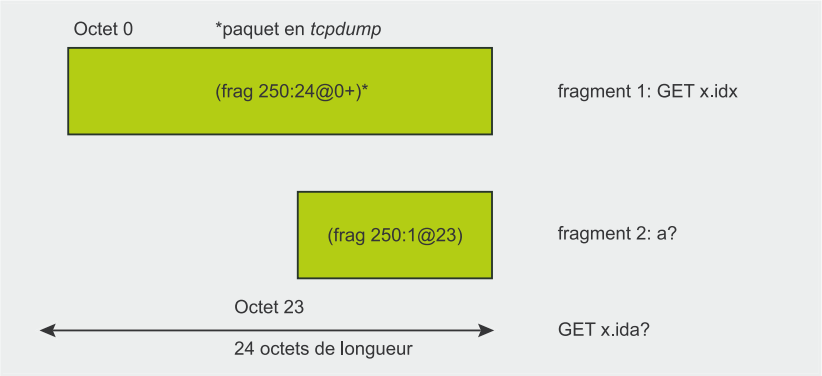


Figure 3. *Contournement d'un IDS par chevauchement de fragments*

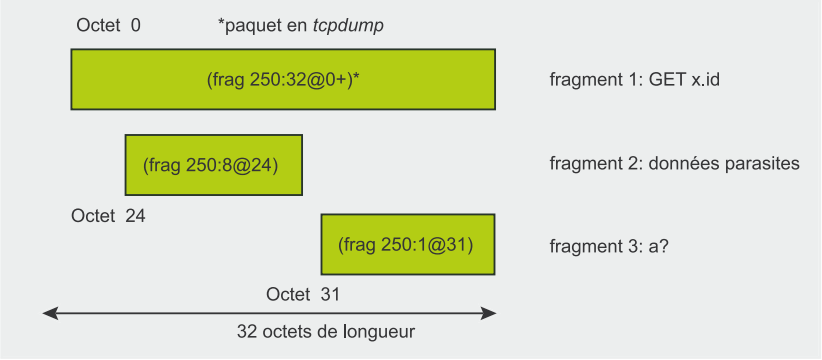


Figure 4. *Technique de l'écrasement de fragments*

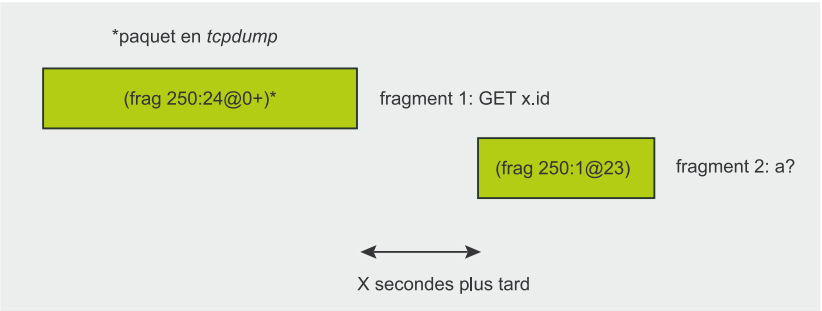


Figure 5. *Technique de temporisation de la fragmentation*

Ecrasement de fragments

La différence entre le chevauchement de fragments et l'écrasement de fragments est que dans cette dernière méthode, un fragment complet écrase le fragment précédent (voir la Figure 4). Supposons de nouveau que nous envoyions trois fragments :

- fragment 1 – chaîne `GET x.id`,
- fragment 2 – quelques éléments aléatoires,
- fragment 3 – chaîne `a?`.

Selon la manière dont l'hôte réassemble les fragments (s'il favorise par exemple les vieux ou les nouveaux fragments), il pourrait s'agir d'une tentative de débordement de tampon ou encore de quelques URL accidentelles (inexistantes).

Temporisation de la fragmentation

La temporisation dépend du temps que le système de détection d'intrusion passe à stocker les fragments dans le mémoire avant de se débarrasser du paquet de données. La plupart des systèmes temporiseront un fragment incomplet en 60 secondes. Si le système de détection d'intrusion ne gère pas le fragment en 60 secondes, il est alors possible d'envoyer des paquets de la façon suivante :

- fragment 1 – `GET x.id` avec la balise de MF (*More Fragments*),
- fragment 2 – `a?` (X secondes plus tard).

Si le système de détection d'intrusion ne conserve pas le fragment initial pendant plusieurs secondes, il est alors possible de contourner le système.

La fragmentation peut être combinée à d'autres techniques, comme par exemple mettre un terme aux valeurs de durée de vie. Si l'hôte est assez vulnérable derrière le système de détection d'intrusion, celui-ci peut voir un paquet qui expire avant d'avoir atteint l'hôte destinataire :

- paquet 1 – `GET x.id` avec une durée de vie >2,
- paquet 2 – `s_evasion.html` avec une durée de vie =1,
- paque 3 – `a?` avec une durée de vie >2.

Dans cet exemple, le système de détection d'intrusion considérera la requête comme étant `GET x.ids_evasion.html` ; toutefois, si le deuxième paquet se temporise avant son arrivée chez l'hôte, ce dernier verra `GET x.ida?`.

Il est probable que la signature par défaut de *Snort* pour le débordement de tampon lors de la gestion des fichiers *.ida* (voir le Listing 1)

n'intercepte aucune de ces techniques de fragmentation (des exceptions sont possibles si les codes de préprocesseurs comme *frag2* sont utilisés – ce genre de code étant exécuté avant le moteur de détection).

Toutefois, *Snort* dispose d'une signature pour détecter les paquets fragmentés :

```
alert ip $EXTERNAL_NET any ←
 -> $HOME_NET any (msg:"MISC ←
 Tiny Fragments"; fragbits:M; ←
 dsize: < 25; classtype:bad-unknown; ←
 sid:522)
```

De telles techniques peuvent également être mises en échec. Dans le cas d'un exploit *.ida*, l'URL requise n'est pas de grande importance. Vous auriez donc pu exécuter l'attaque frontale avec de nombreuses données parasites afin d'éviter le déclenchement d'une règle de fragment :

- paquet 1 – `GET une_longue_chaîne_rendant_impossible_la_détection.com`,
- paquet 2 – `a?`.

Dug Song a sorti l'outil *fragroute*, chargé de contrôler de nombreuses vulnérabilités liées à la fragmentation. *Snort* vient récemment d'implanter des contrôles et des méthodes afin d'intercepter un plus grand nombre de ruses au niveau du réseau. Un nouvelle sortie officielle devrait donc contenir un bon nombre de ces contrôles.

Attaques Denial of Service (DoS)

L'objectif d'une attaque DoS à surcharger le système de détection d'intrusion de manière à le faire planter. Le but est atteint en fragilisant la disponibilité des ressources du système, en sous-alimentant les processus, en épuisant les espaces consacrés au débit réseau, à la mémoire, à l'unité centrale et au disque. Si quelqu'un crée trop de trafic réseau, la capacité du système de détection d'intrusion à copier des paquets à partir d'un transfert dans le tampon et la mémoire est fragilisée. Les paquet entrants sont alors omis, bien entendu. Par ailleurs, si quelqu'un génère un trafic chaotique considérable, une grande partie de la mémoire est alors requise afin de réassembler les données, ayant pour effet de provoquer une insuffisance de mémoire pour les paquets entrants. Le trafic d'IP fragmentés requiert un grand nombre de cycles de l'unité centrale ce qui peut s'avérer coûteux également.

De toute façon, un attaque classique par flux nécessite plusieurs systèmes afin de dépasser les capacités en pleine expansion des systèmes de détection d'intrusion, alors que l'exploitation d'un bogue n'a besoin que d'un seul système (mais est plus difficile à réaliser que la première attaque). Les signatures sont toujours préparées en fonction d'un nouveau type d'attaques DoS, ce n'est qu'une question de temps.

Le combat éternel

Il est probable que certaines techniques décrites ci-dessus ne soient plus valides ou ne le sont qu'avec certains outils de détection d'intrusion particuliers, mais la logique de ces techniques reste la même. Les fausses attaques positives et négatives, les attaques DoS etc. rendent ces systèmes inoffensifs en tant que mécanismes de sécurité tant que ces derniers ne sont ni configurés ni maintenus correctement. L'ajout de pots de miel, de système de protection contre les intrusions etc. aux systèmes gérés en réseau ne manquera pas de les aguerrir contre les risques de plantage. ■

Glossaire

- IDS (*Intrusion Detection System*) – programme chargé d'identifier des attaques ou des violations de sécurité en surveillant les activités du réseaux et des hôtes.
- IPS (*Intrusion Prevention System*) – logiciel chargé de refuser l'accès à des intrusions.
- IDPS – système regroupant à la fois les IDS et IPS.
- HIDS (*Host Intrusion Detection System*) – système de détection d'intrusion fonctionnant sur l'hôte surveillé et chargé de chercher les intrus.
- NIDS (*Network Intrusion Detection System*) – système de détection d'intrusion chargé d'analyser de manière passive le trafic réseau, en cherchant les activités illégales.
- NNIDS (*Network Node Intrusion Detection System*) – solution hybride chargée de réassembler les systèmes de détection d'intrusion au niveau du réseau, en n'analysant toutefois qu'une partie (ou nœud) du trafic réseau.
- AIDS (*Anti-IDS*) – outil anti-IDS permettant de contourner la détection à base de signature des IDS.
- CIDF – ensemble de composants définissant un IDS.
- Signature – ensemble de règles permettant à un système de détection d'intrusion d'identifier une menace.
- Débordement de tampon – erreur survenant lorsqu'un programme ou un processus tente de stocker plus de données dans un tampon (zone de stockage de données temporaire) que la place prévue à cet effet.

Sur Internet

- *http://www.monkey.org/~dugsong/fragroute* – la page d'accueil de l'outil *fragroute,*
- *http://www.snort.org* – le site de *Snort* : programme, documentation et signatures,
- *http://www.wiretrip.net/rfp* – le scanner CGI *Whisker,*
- *http://sans.org/rr* – une mine de documentation technique sur les solutions des systèmes de détection d'intrusion,
- *http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx* – le bogue de débordement de tampon lors de la gestion de fichiers *.ida*.