Prattique

Système IPS basé sur Snort

Michał Piotrowski

Degré de difficulté

Pour se protéger contre les attaques sur les systèmes informatiques, le plus souvent on utilise les pare-feux ; et pour surveiller les attaques – les systèmes de détection d'intrusions. Pourtant, la détection des intrus ne suffit point. Cela ne servira à rien de détecter une attaque, si l'on n'est pas capable de la faire échouer. La solution est de construire un système de prévention des attaques – cet article explique comment construire un tel système et comment le maintenir.

Les outils les plus populaires servant à protéger les réseaux informatiques contre les attaques des cyberpirates sont les pare-feux et les systèmes de détection d'intrusions (IDS, en anglais *Intrusion Detection Systems*). Tandis que le fonctionnement des pare-feux consiste à contrôler les paquets arrivants vers le réseau, les systèmes de détection d'intrusions analysent le contenu de ces paquets et au moment où une irrégularité ou une information caractéristique pour l'attaque est détectée, ils donnent l'alerte.

Pourtant, le niveau de sécurité atteint par le biais de cette technique n'est pas satisfaisant. Tout d'abord, le pare-feu doit laisser passer une partie du trafic – si non, la connexion d'un réseau protégé avec le reste du monde n'a aucun sens, et l'attaque peut avoir lieu justement sur le service qui est accessible. Évidemment, le système IDS est capable de détecter une attaque qu'un pare-feu a laissé passé par, mais tout en étant un simple observateur, il n'est pas capable de la faire échouer, donc sa présence n'aura qu'une valeur informative.

Bien sûr, il est possible de coupler le système IDS avec un pare-feu pour bloquer en temps réel les tentatives de pénétration ou le configurer de façon à ce qu'il interrompe les connexions suspectes. Hélas, une telle solution a beaucoup de défauts. Premièrement, beaucoup d'attaques consistent à envoyer seulement un ou quelques paquets. Dans la plupart des cas, les attaques de type DoS sur un programme ou un système plantant après la réception des données spécialement préparées à cet effet, ou les attaques de débordement de tampon contraignant le système attaqué à établir une connexion de retour avec l'ordinateur de l'intrus réussiront, même si le système IDS envoie au pare-feu l'information

Cet article explique...

- ce-qu'est un système de prévention des attaques,
- comment installer, configurer et maintenir un système IPS basé sur le programme Snort.

Ce qu'il faut savoir...

- notions de base de l'administration du système Linux,
- notions de base du fonctionnement du réseau TCP/IP.

Netfilter

Le mécanisme netfilter est un sous-système du noyau de Linux permettant le filtrage, la modification des paquets et la traduction des adresses réseau (en anglais *Network Address Translation* – NAT). Il est apparu dans les noyaux de la série 2.4 et est toujours développé dans la série 2.6.

Pour configurer les règles de filtrage ou de traduction, on utilise le programme fonctionnant dans l'espace utilisateur appelé iptables. Mais il faut savoir que ce n'est pas un seul moyen de contrôler les règles de filtrage du trafic réseau dans le noyau du système.

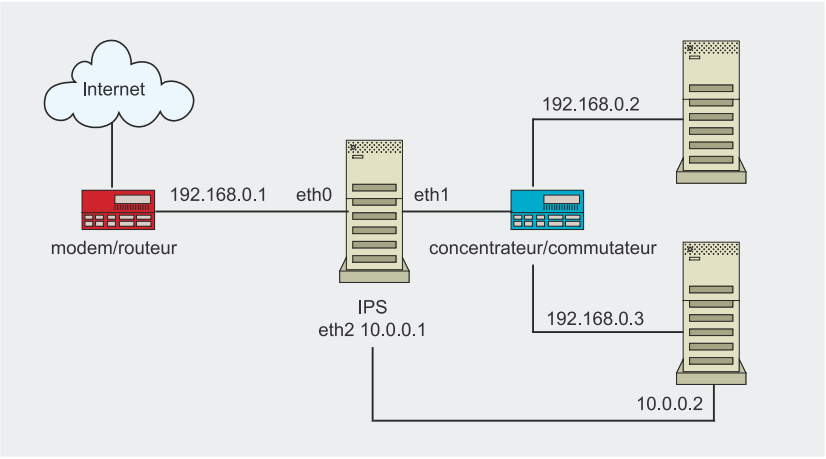

Figure 1. *La place du système IPS dans le réseau*

sur le blocage de l'adresse IP donnée. Deuxièmement, l'intrus peut exploiter cette propriété de l'IDS pour bloquer un groupe d'adresses donné en simulant les attaques provenant de celles-ci.

La solution très efficace de ces problèmes sont les systèmes de prévention d'intrusions – IPS (en anglais *Intrusion Prevention System*) qui intègrent les pare-feux et les systèmes de détection d'intrusion (IDS). Les systèmes IPS sont implantés dans le réseau de la même façon que les pare-feux – sur le chemin des paquets, de façon à ce que toutes les données envoyées sur le réseau passent par celui. L'IPS analyse ces données du point de vue des propriétés caractéristiques aux types des attaques connus et en fonction de leur classification, il les laissent passer ou les bloquent.

Il existe beaucoup de solutions IPS sur le marché. Leurs prix sont différents : de quelques jusqu'à quelques milles dollars. Essayez de construire votre propre système IPS basé sur les programmes disponibles généralement sur Internet.

Outils

Dans cet article le système sera basé sur Linux avec le noyau en version 2.6.12.6. C'est important car les noyaux de la série 2.6 supportent la création des ponts réseau, tandis que les noyaux 2.4 exigent pour cela des correctifs appropriés. La distribution de Linux n'est pas ici essentielle. Ce qui est important, c'est que cela soit une installation assez simple, dépourvue de Xwindow, d'applications multimédias et d'autres outils de ce type.

Le cœur de notre IPS sera le programme *open source Snort IDS* en version 2.4.0. C'est un programme très avancé, utilisé dans quelques systèmes IDS/IPS commerciaux. Quant à vous, servirez-vous de la version 2.4.0 parce qu'elle est intégrée au projet *snort_inline* permettant de télécharger les paquets non via la librairie libpcap, comme cela a lieu dans la configuration standard de *Snort*, mais via le mécanisme netfilter et le programme iptables.

De plus, vous aurez besoin de quelques bibliothèques et outils. Avant tout, ce sont les bibliothèques libnet 1.0.x, LIBIPQ et le programme *bridge-utils*. La bibliothèque LIBIPQ appartient au paquet iptables et vous la trouverez dans les modules supplémentaires de développement ou installerez à partir des sources, en installant iptables par la commande `make install-devel`. Utiliserez aussi le programme Oinkmaster qui permettra la mise à jour automatique de la base de signatures.

L'ordinateur sur lequel vous lancerez l'IPS, est muni de trois cartes réseaux. Seulement une d'elles aura l'adresse IP affectée et elle servira à la gestion de la machine. Deux autres seront configurées uniquement jusqu'à la couche 2 du modèle OSI et les paquets envoyés sur le réseau seront transférés entre elles. Ainsi, votre IPS constituera un pont, transparent pour les autres périphériques et ordinateurs. Le schéma d'un réseau après la connexion de l'IPS de ce type est présenté sur la Figure 1. Dans cet article, on ne construit pas tout le réseau présenté – on se concentrera à la création du périphérique IPS.

Construction d'un pont

Le pont réseau (en anglais *bridge*) est un équipement qui fonctionne dans la couche de liaison des données du modèle OSI et sert à relier différents segments du réseau informatique. Il existe deux principaux avantages d'utilisation du pont comme IPS ou pare-feu :

- Configuration simple – elle est due au fait que le pont ne possède pas d'adresses IP et peut être placé à l'intérieur du réseau sans changement de l'adressage ou du routage sur d'autres périphériques. La connection de l'IPS de ce type est similaire à la connexion d'un commutateur ordinaire.
- Sécurité – consiste à ce que le périphérique soit transparent, alors pratiquement indétectable par différents types de scanneurs. Il n'a pas d'adresse IP, alors il est impossible de s'y connecter et de l'attaquer. Il est vrai qu'on peut exploiter la faille dans les programmes IPS qui, par exemple, plante au moment du traitement du paquet spéciale-

ment conçu à cet effet, mais heureusement, les problèmes de ce type n'arrivent pas trop souvent.

Commencez la transformation de votre ordinateur en pont par configurer deux interfaces réseau de l'IPS de façon à ce qu'elles échangent les paquets entre elles. Pour ce faire, vous devez compiler le noyau avec les options présentées dans le Listing 1.

Après le redémarrage du système, ajoutez une nouvelle interface virtuelle br0 et lui affectez les interfaces réelles eth0 et eth1, en tapant les commandes suivantes :

```
# ifconfig eth0 0.0.0.0 up
# ifconfig eth1 0.0.0.0 up
# brctl addbr br0
# brctl addif br0 eth0
# brctl addif br0 eth1
# ifconfig br0 0.0.0.0 up
```

Configurez également l'interface eth2, servant à gérer le périphérique :

```
# ifconfig eth2 10.0.0.1 \
  netmask 255.255.255.0 up
```

Dès ce moment, tous les paquets vus par l'interface eth0 seront envoyés vers les segments du réseau de l'autre côté de l'IPS à travers l'interface eth1 et vice-versa. La carte eth2 a une adresse IP affectée et elle permet de se loguer à distance sur le périphérique.

Installation de Snort

L'installation de Snort est une installation standard. Pourtant, pendant la configuration du paquet, il faut ajouter l'option `--enable-inline`, qui permettra au programme de travailler en mode *inline*. Ainsi, Snort pourra être situé sur le chemin des paquets. La configuration, la compilation et l'installation du programme sont effectuées au moyen des commandes suivantes :

```
$ ./configure --enable-inline
$ make
# make install
```

Ensuite, il faut créer le répertoire */etc/snort* et y metter tous les fichiers de configuration nécessaires :

```
# cp classification.config \
  gen-msg.map \
  generators \
  reference.config \
  sid sid-msg.map \
  snort.conf \
  threshold.conf \
  unicode.map \
  /etc/snort
```

À la fin, il faut modifier le fichier de configuration principal *snort.conf*. Avant tout, vous ne disposez pas encore de signatures d'attaques, commentez donc toutes les lignes ajoutant les fichiers de signatures qui se trouvent à la fin du fichier et qui ont la forme `include $RULE_PATH/*.rules` (le début de la ligne est précédé du caractère #). Changez aussi la valeur de la variable définissant le répertoire comprenant ces fichiers de `var RULE_PATH ../rules` en `var RULE_PATH /etc/snort/rules`.

Vérification des signatures

Les règles d'attaques que l'on peut télécharger à partir du site de Snort sont divisées en trois groupes : les signatures payables (*subscription rules*), les signatures nécessitant l'enregistrement (*registration rules*) et les signatures universellement disponibles (*unregistered rules*). Vu que les règles universellement disponibles ne sont mises à jour qu'au moment de l'édition de la version successive de Snort et l'accès aux règles payables doit être régulièrement acquitté – il est préférable d'utiliser les règles disponibles après l'enregistrement.

Mais avant de télécharger et d'installer les règles, testez tout ce que vous avez réussi à construire jusqu'alors. Vous créerez quelques exemples de signature qui vous permettront de connaître les possibilités de votre IPS. Pour cela, vous exploiterez trois nouvelles types de règles– disponibles uniquement en version *inline* – définissant les

Types de systèmes IPS

Le périphérique présenté est un système réseau de détection des intrusions (NIPS, en anglais *Network Intrusion Prevention System*). Actuellement, c'est le type de systèmes IPS le plus populaire. D'autres systèmes de ce type sont :

- Commutateurs de septième couche (de l'anglais *Layer Seven Switches*) – ce sont les périphériques très similaires à l'IPS présenté. En général, ils servent à répartir les charges sur plusieurs dispositifs, mais sont aussi capables d'arrêter les paquets sélectionnés à partir d'une base de règles.
- IPS d'application (HIPS, en anglais *Host Intrusion Prevention System*) – ce sont des solutions logicielles, installées localement sur chaque station protégée qui sont intégrées au système d'exploitation et surveillent le fonctionnement des autres applications. Ils permettent de protéger le système contre les dangers les plus connus, comme les débordement de tampon, les virus, les chevaux de Troie ou les logiciels espions.

Listing 1. *La configuration du noyau du système*

```
Device Drivers
  Networking support
    Networking options
      <*> 802.1d Ethernet Bridging
      Network packet filtering (replaces ipchains)
        <*> Bridged IP/ARP packets filtering
          IP: Netfilter Configuration
            <*> Userspace queueing via NETLINK
            <*> IP tables support (required for filtering/masq/NAT)
          Bridge: Netfilter Configuration
            <*> Ethernet Bridge tables (ebtables) support
```

actions entreprises par Snort au moment du lancement de la signature. Ce sont :

- *drop* – Snort enregistrera le fait de l'apparition du paquet qui correspond à la signature et enverra à l'iptables le signal de refus.
- *sdrop* – le paquet sera refusé, mais l'information sur ce fait ne sera pas enregistrée.
- *reject* – le paquet sera refusé et enregistré et la connexion interrompue (RST en cas du protocole TCP) ou le paquet *ICMP Port Unreachable* sera envoyé (en cas du protocole UDP).

Pour que les règles de type *reject* soient capables de réinitialiser les connexions, il faut ajouter au fichier de configuration l'option `config layer2resets`, grâce à laquelle l'IPS enverra les paquets de réinitialisation à partir des interfaces ne possédant pas d'adresse IP. Par défaut, l'adresse MAC source dans ces paquets est l'adresse de la carte réseau d'entrée, mais vous pouvez la modifier au moyen de l'option `config layer2resets: 00:01:02:03:04:05`.

La première signature se présente ainsi : `drop tcp any any -> any 22 (classtype:attempted-user; msg: "Port 22 Connection Initiated";)`. C'est une règle très simple qui reconnaît, bloque et enregistre tous les paquets TCP passant par l'IPS qui sont destinés au port 22. En résultat, l'IPS empêchera l'établissement des connexions aux serveurs SSH. Le Listing 2 présente la notation dans le journal d'évènements qui sera créée par Snort après avoir intercepté les paquets correspondant à la signature. Comme vous pouvez voir, c'est le paquet SYN qui commence le processus d'établissement de la connexion du protocole TCP.

La seconde règle : `alert icmp any any <> any any (classtype: attempted-user; msg:"ICMP Echo Request"; icode:0; itype:8;)` reconnaîtra et enregistrera tous les paquets ICMP de type Echo Request. Les fichiers journaux de Snort contiendront la notation

Listing 2. *La réaction de Snort à la première signature*

```
[**] [1:0:0] Port 22 Connection Initiated [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/19-20:19:07.436667 192.168.0.2:1049 -> 193.219.28.2:22
TCP TTL:128 TOS:0x0 ID:702 IpLen:20 DgmLen:48 DF
******S* Seq: 0x29821EB9  Ack: 0x0  Win: 0xFAF0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

Listing 3. *La réaction de Snort à la deuxième signature*

```
[**] [1:0:0] ICMP Echo Request [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/19-20:12:57.194560 192.168.0.2 -> 212.76.32.1
ICMP TTL:128 TOS:0x0 ID:420 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512  Seq:256  ECHO
```

Listing 4. *La réaction de Snort à la troisième signature*

```
[**] [1:0:0] DNS Request [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/19-20:21:12.989775 192.168.0.2:1041 -> 212.76.39.45:53
UDP TTL:128 TOS:0x0 ID:818 IpLen:20 DgmLen:59
Len: 31
```

similaire à celle présentée dans le Listing 3.

La dernière signature de test est la plus intéressante : `alert udp any any <> any 53 (classtype:attempted-user; msg:"DNS Request"; content: "yahoo"; replace:"lycos";)`. Cette règle détectera et enregistrera tous les paquets UDP à destination du port 53, c'est-à-dire du serveur DNS qui contiennent la chaîne de caractères *yahoo*. Les paquets seront laissés passés par l'IPS, mais le mot *yahoo* sera changé en *lycos*. C'est le champ `replace` dans la signature qui en est responsable ; il définit en quoi sera changé le contenu du champ `content`.

En résultat, quand une requête concernera l'adresse *www.yahoo.com*, le serveur DNS répondra par l'adresse IP du serveur *www.lycos.com*, et les journaux contiendront l'information présentée dans le Listing 4. Cette propriété de Snort *inline* est fort utile dans la protection des systèmes honeypot, quand vous voulez que l'intrus y pénètre, mais ne soit pas capables d'effectuer une attaque efficace contre un ordinateur du réseau. Il suffit de modifier dans le système IPS la signature reconnaissant le shellcode de façon affichée dans le Listing 5 et toutes les attaques qui lui ressemblent ne réussiront pas.

Toutes les règles ci-dessus doivent être placées dans le répertoire */etc/snort/rules* dans le fichier *test.rules*, et à la fin du fichier */etc/snort/snort.conf*, ajoutez la ligne `include $RULE_PATH/test.rules`. Configurez l'iptables de façon à ce que les paquets passent à travers Snort et lancez ce dernier :

```
# iptables -P FORWARD DROP
# iptables -A FORWARD -j QUEUE
# snort -Q \
  -c /etc/snort/snort.conf \
  -l /var/log/snort -v
```

La dernière commande lance Snort en mode *inline* (option `-Q`). La configuration est téléchargée à partir du fichier */etc/snort/snort.conf* (`-c`), et les journaux sont sauvegardés dans le répertoire */var/log/snort* (`-l`). Pendant les tests, servez-vous aussi de l'option `-v`, grâce à laquelle l'IPS fonctionne en mode information et affiche beaucoup de messages qui permettent de vous faire une idée précise des erreurs commises.

Listing 5. *Une simple modification de la signature corrompra le shellcode et empêchera l'attaque efficace*

Avant la modification :

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any ←
  (msg:"SHELLCODE Linux shellcode"; ←
  content:"|90 90 90 E8 C0 FF FF FF|/bin/sh"; ←
  reference:arachnids,343; classtype:shellcode-detect; sid:652; rev:9;)
```

Après la modification :

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any ←
  (msg:"SHELLCODE Linux shellcode"; ←
  content:"|90 90 90 E8 C0 FF FF FF|/bin/sh"; ←
  replace:"|90 90 90 E8 C0 FF FF FF|/ben/sh"; ←
  reference:arachnids,343; classtype:shellcode-detect; sid:652; rev:9;)
```

L'option `-v` sera remplacée par `-D`, ainsi, Snort fonctionnera en tâche de fond comme démon.

Installation des règles officielles

Il est temps de munir votre système de prévention d'attaques de signatures officielles des attaques. Vu que vous allez vous servir des signatures disponibles pour les utilisateurs enregistrés, vous devez créer un compte utilisateur sur le site de Snort. Une fois ce compte créé, vous téléchargerez les signatures récentes, les déplacerez vers le répertoire */etc/snort* et les décompacterez.

L'action par défaut entreprise par Snort pour toutes les règles consiste à enregistrer l'attaque détectée (directive `alert`). Étant donné que vous avez l'intention de bloquer ces attaques, vous devez modifier toutes les règles, en changeant l'action `alert` en `drop`. Pour ce faire, tapez la commande :

```
$ for f in `ls *.rules` ;\
  do  sed s/^alert/drop/g \
  $f > ${f}.new ; \
  mv ${f}.new $f ; \
  done
```

Il faut aussi corriger la partie finale du fichier *snort.conf* de façon à ce que les signatures soient chargées lors du démarrage du programme (on a commenté toutes les lignes activant les fichiers de signature). À la fin, vous lancez Snort :

```
# snort -Q -D \
  -c /etc/snort/snort.conf \
  -l /var/log/snort
```

Mais il ne faut pas oublier que l'installation d'un nouveau système IPS dans l'environnement réseau et l'activation du blocage pour toutes les règles n'est pas recommandée. Tous les périphériques IDS/IPS doivent être ajustés à un réseau concret pour éliminer de fausses alarmes qui apparaissent toujours au début de leur fonctionnement. Si vous forcez le système à bloquer tout ce qui lui paraît suspect sans lui apprendre la spécificité de votre réseau, il se peut qu'une partie des services ne fonctionnent pas ou bien manifestent des perturbations car l'IPS ne laissera passer que certains paquets. Il est donc préférable de vérifier comment les procédures réagissent au trafic typique de votre réseau en enregistrant les attaques détectées et exclure ces règles qui suscitent de fausses alarmes. C'est à ce moment qu'on peut permettre au périphérique de bloquer les attaques.

Mises à jour automatiques

Chaque système IDS/IPS, même s'il utilise les règles récentes, devient très vite inactuel. De nouveaux dangers apparaissent aussi rapidement que les systèmes typiques – pour rester efficaces – doivent être régulièrement mis à jour. Les mises à jour manuelles sont assez fastidieuses, essayez donc de les automatiser à l'aide de l'outil Oinkmaster version 1.2. Pour cela, outre le programme en tant que tel, vous aurez besoin de ce qu'on appelle OinkCode qui vous permettra d'accéder aux règles destinées aux utilisateurs enregistrés de Snort. Vous pouvez générer le code après vous être logués sur votre compte sur le site de Snort.

Oinkmaster est un script en Perl, donc son installation est assez simple :

```
$ tar zxvf oinkmaster-1.2.tar.gz
$ cd oinkmaster-1.2
# cp oinkmaster.pl /usr/local/bin/
# cp oinkmaster.conf /etc/
```

La configuration qui consiste à éditer le fichier *oinkmaster.conf* ne doit pas poser de problème. Tout d'abord, il faut décider quelles signatures l'on veut télécharger. Vu que vous tenez aux règles les plus récentes, vous modifierez donc la ligne `# url = http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/snortrules-snapshot-CURRENT.tar.gz` de façon à ôter le caractère `#` du début de la ligne, et au lieu de `<oinkcode>`, saisissez le code généré par vous à l'aide du script du site de Snort.

Si vous laissez la configuration de l'Oinkmaster comme ça, de nouvelles signatures auront une forme standard, c'est-à-dire, elles informeront sur les attaques détec-

Configuration de l'iptables

Pour diriger les données envoyées via le réseau vers un programme dans l'espace utilisateur, vous avez utilisé la commande `iptables -A FORWARD -j QUEUE` qui embrasse le flux de données entier. En résultat, tous les paquets passant par l'IPS seront analysés. Mais vous pouvez observer seulement les connexions choisies. Par exemple, si vous voulez que Snort n'effectue la recherche que dans les paquets envoyés vers les serveurs Web, vous pouvez utiliser la commande `iptables -A FORWARD -p tcp --dport 80 -j QUEUE`.

À propos de l'auteur

Michał Piotrowski, titulaire de la maîtrise d'informatique, a plusieurs d'années d'expérience dans l'administration des réseaux et des systèmes. Pendant plus de trois années, il travaillait comme inspecteur de sécurité dans une institution supportant l'autorité de certification supérieure dans l'infrastructure polonaise de PKI. Actuellement, il est spécialiste en sécurité téléinformatique dans l'une des plus grandes institutions financières en Pologne. Dans ses moments libres, il s'occupe de la programmation et de la cryptographie.

Sur Internet

- *http://www.snort.org* – le site du projet Snort,
- *http://bridge.sourceforge.net* – le site du jeu d'outils bridge-utils,
- *http://www.netfilter.org* – le site du projet netfilter et du programme iptables,
- *http://www.packetfactory.net/libnet/* – le site de la bibliothèque libnet,
- *http://oinkmaster.sourceforge.net/* – le site du programme Oinkmaster.

tées. Quant à vous, vous voulez que les attaques soient bloquées, vous devez donc ajouter au fichier *oinkmaster.conf* une notation grâce à laquelle toutes les règles téléchargées seront modifiées par le changement de l'action standard `alert` en `drop` : `modifysid * "^alert" | "drop"`. De la même manière, vous pouvez déterminer dans le programme quelles règles seront désactivées par défaut (directive `disablesid <n° de la signature>`). C'est très utile quand votre IPS est bien ajusté et vous ne voulez pas que la mise à jour des signatures compromette tout, par exemple active les règles que vous avez décidé à désactiver.

Le programme est démarré par la commande :

```
# oinkmaster.pl
  -o /etc/snort/rules/
```

où le paramètre `-o` définit le répertoire avec de nouvelles règles. Le paramètre `-b` est aussi très utile car il indique le répertoire dans lequel les fichiers de signatures précédents seront déplacés. Pour que tout fonctionne correctement, après chaque mise à jour des règles, Snort doit être rechargé. Alors, la dernière chose à faire est de créer un simple script qui automatisera tout le processus et l'ajouter à */etc/crontab* ou à un fichier d'un autre gestionnaire de tâches. ●