ADMINISTRATION ET MAINTENNACE D'UN SYSTÈME GNU/LINUX

Protection des réseaux

SYSTÈME D'EXPLOITATION GNU/LINUX

CENTRE DES CLASSES DE PRÉPARATION DU BTS – ESSAOUIRA-

Mr. Mostafa ESSADDOUKI (essaddouki@gmail.com)

Table des matières

I. Routage et filtrage

1. Configuration d'un serveur Linux en tant que routeur

a. *Activation du routage sur un serveur Linux*

L'activation du routage se fait en modifiant le contenu du fichier **/proc/sys/net/ipv4/ip_forward**. Ce fichier contient un seul caractère, par défaut **0** pour indiquer que le routage est inactif.

Modification du fichier ip_forward pour activer le routage

echo 1 > /proc/sys/net/ipv4/ip_forward

Une fois cette manipulation effectuée, la machine Linux est prête à router les paquets se présentant sur ses interfaces. Ce paramètre est volatile et sera perdu dès la machine éteinte. Toutefois, on peut évidemment annuler le routage en effectuant l'opération inverse.

Modification du fichier ip_forward pour désactiver le routage

echo 0 > /proc/sys/net/ipv4/ip_forward

Autre possibilité, la commande **sysctl** qui permet de modifier dynamiquement des paramètres fonctionnels du noyau. **sysctl** permet de modifier directement tous les fichiers se trouvant sous l'arborescence **/proc/sys**.

Activation du routage avec sysctl

sysctl net.ipv4.ip_forward=1

Ces commandes sont effectives toute la durée de la session et doivent être retapées après chaque redémarrage. On peut bien entendu les placer dans un script de service appelé au démarrage, ou modifier le fichier **/etc/sysctl.conf**.

Activation permanente du routage dans le fichier /etc/sysctl.conf

net.ipv4.ip_forward = 1

b. *Consultation de la table de routage*

À ce stade, le routeur Linux est parfaitement capable de router les paquets. Toutefois, il ne pourra le faire que vers des réseaux connus, c'est-à-dire référencés dans sa table de routage.

La table de routage est maintenue en mémoire mais elle peut être consultée par quelques commandes.

Affichage de la table de routage par la commande route

route -n

Le paramètre -n est facultatif, mais il fait gagner beaucoup de temps à l'affichage car il dispense la commande de tenter de résoudre les adresses renvoyées en noms. Or, si l'adresse en question

n'est pas renseignée dans une zone DNS inverse, cette requête se fait pour rien et il faut attendre plusieurs secondes pour que l'affichage arrive.

Affichage de la table de routage par la commande netstat

netstat -nr

Où l'option -r demande à la commande d'afficher la table de routage et -n de ne pas faire de résolution de noms. La commande **netstat** a de nombreux usages, mais elle est souvent utilisée dans ce simple cadre de consultation de la table de routage.

c. *Gestion des routes statiques*

Les seules entrées présentes automatiquement dans la table de routage sont les réseaux auxquels le routeur est directement connecté, ainsi que la passerelle par défaut. Le routeur peut donc exploiter ces entrées de la table de routage sans autre configuration. Si le routeur doit router des paquets vers d'autres réseaux, il faudra ajouter manuellement les routes dans la table de routage.

Ajout de route statique dans la table de routage

route add -net réseau_cible netmask masque gw routeur

Ajout de passerelle par défaut

route add default gw routeur

route add -net 0.0.0.0 gw routeur

Dans la deuxième syntaxe, 0.0.0.0 représente la route par défaut. Cette représentation de la route par défaut est universelle et applicable sur la quasi-totalité des systèmes exploitant une table de routage IP.

Bien entendu, il est possible de supprimer les routes statiques qui ne sont plus nécessaires ou enregistrées par erreur.

Suppression de route statique de la table de routage

route del -net réseau_cible netmask masque

2. iptables

Les iptables sont utilisées pour gérer le filtrage de paquets IP au sein d'un système Linux. Elles exploitent une commande unique : **iptables**, et se configurent par l'application successive de règles de gestion de paquets. Les iptables peuvent filtrer le trafic en transit dans un routeur Linux, mais aussi le trafic entrant et sortant de tout serveur ou poste de travail à une seule interface.

a. *Les tables*

Les iptables s'appuient sur des tables associées à un mode fonctionnel. Selon le type de règle que l'on souhaite ajouter au fonctionnement des iptables, on précisera la table associée. Les tables principales utilisées sont **filter** pour le filtrage de paquets et **nat** pour la translation d'adresses entre un réseau privé et un réseau public.

La table **filter** est la table par défaut. Aussi, quand on établit une règle iptables dans un but de filtrer les paquets est-elle sous-entendue et donc non précisée.

La table **nat** sert à la translation d'adresses et doit être systématiquement précisée quand elle est invoquée.

b. *Les chaînes*

Une chaîne iptables représente un type de trafic du point de vue de sa circulation dans une machine. Les chaînes permettent de préciser si une règle doit s'appliquer à du trafic qui entre dans une machine, qui en sort ou qui la traverse.

La chaîne **INPUT** désigne le trafic entrant, la chaîne **OUTPUT** désigne le trafic sortant, et la chaîne **FORWARD** désigne le trafic qui traverse la machine, entrant par une interface et sortant par une autre. Attention, même si un paquet qui traverse le routeur est d'un point de vue physique respectivement entrant, traversant et sortant, iptables le considérera comme traversant seulement (chaîne FORWARD). Les chaînes INPUT et OUTPUT sont réservées au trafic à destination ou en provenance explicite de l'hôte soumis aux règles.

Une autre chaîne appelée **POSTROUTING** et utilisée dans la configuration du NAT a pour objet d'appliquer un traitement à un paquet après une opération de routage.

Les chaînes sont toujours indiquées en majuscules dans une syntaxe iptables.

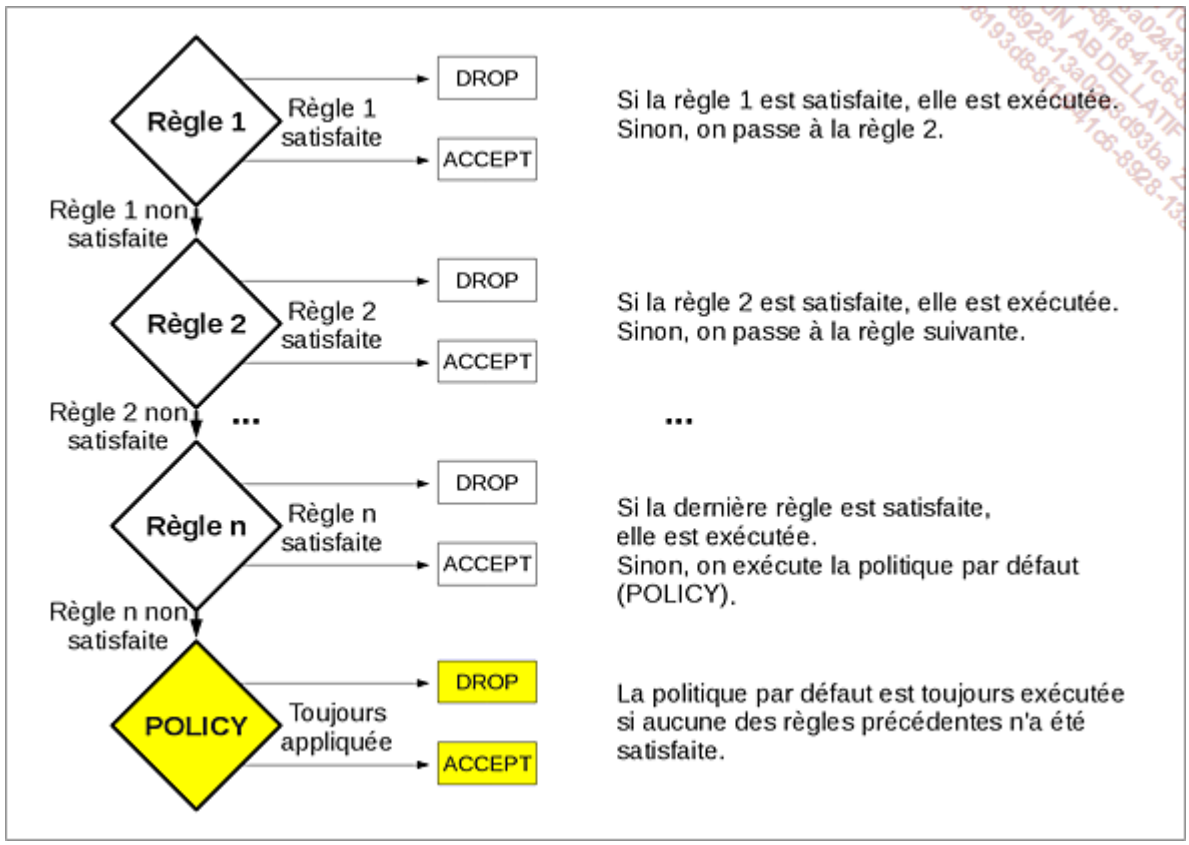
c. *Les actions*

Quand une règle est satisfaite, une action est engendrée par le système sur le paquet testé. Les principales actions sont **ACCEPT** qui laisse passer le paquet et **DROP**, qui le détruit.

Dans une syntaxe iptables, l'action (**target** dans le manuel en ligne) est annoncée par le paramètre **-j**.

Les actions sont toujours indiquées en majuscules dans une syntaxe iptables.

d. *Le traitement des règles*



Les règles sont appliquées une par une à tout paquet filtré. Si une règle est satisfaite, une action est engagée sur le paquet et le traitement s'arrête. Si une règle n'est pas satisfaite, la règle suivante est testée. Dans le cas où aucune des règles n'est satisfaite, le paquet subit un traitement par défaut paramétré dans une règle spécifique appelée politique (policy).

Il est possible d'afficher les règles appliquées dans l'ordre pour chacune des chaînes.

Affichage des règles effectives

iptables -L

Exemple d'affichage des règles

Cet exemple affiche les règles en vigueur sur un système Linux non configuré. On y voit la politique appliquée pour chacune des chaînes, et on constate l'absence de règles de filtrage.

```
alpha:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

La commande **iptables -L** affiche une interprétation des règles en vigueur. Si on souhaite connaître les syntaxes qui ont permis d'établir ces règles, il est préférable d'utiliser l'option **-S**.

Exemple d'affichage des règles selon les syntaxes

L'option **-S** est particulièrement utile quand on est confronté à un système configuré par un tiers et qu'on ne sait pas quelles sont les commandes qui ont conduit à une configuration.

```
alpha:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

II. Administration d'un pare-feu avec les iptables

1. Politiques

a. *Principe des politiques de pare-feu*

Un pare-feu peut fonctionner selon deux modes distinct : « tout ce qui n'est pas autorisé est interdit », ou « tout ce qui n'est pas interdit est autorisé ». Pour définir le comportement par défaut, les iptables permettent de définir pour chaque chaîne une action par défaut.

Définition de la politique par défaut des iptables

iptables -P chaine action

Où *chaine* représente le type de trafic (INPUT, OUTPUT et FORWARD), et *action* le comportement souhaité (DROP ou ACCEPT).

b. *Configuration d'une politique de base*

Si l'hôte à configurer est appelé à devenir un pare-feu, il est probable que tout trafic soit interdit par défaut. Cette configuration courante consiste à définir sur les trois chaînes INPUT, OUTPUT et FORWARD une politique de non-traitement des paquets.

Configuration d'une politique restrictive

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -P FORWARD DROP

2. Filtrage de paquets

a. *Politique et règles*

Après avoir configuré une politique qui décrit le comportement de base du pare-feu, il faut créer des règles spécifiques aux éléments de trafics que l'on souhaite laisser passer ou interdire. La philosophie du pare-feu est : on définit le comportement général avec les politiques, et on gère au cas par cas les comportements spécifiques avec des règles.

b. *Création de règles*

Pour chaque élément de trafic devant être autorisé ou interdit, il faudra créer une règle spécifique.

Syntaxe création d'une règle de gestion de trafic

iptables -A chaine -s ip_source -d ip_dest -p protocole --dport port -j action

iptables : création de règle	
-A *chaine*	On ajoute une règle dans la chaîne *chaine* (INPUT, OUTPUT ou FORWARD).
-s *ip_source*	Facultatif : l'adresse IP source d'où proviennent les paquets soumis à la règle. Si l'adresse est une adresse de réseau, préciser le masque.
-d *ip_dest*	Facultatif : l'adresse IP de destination vers laquelle vont les paquets soumis à la règle. Si l'adresse est une adresse de réseau, préciser le masque.
-p *protocole*	Indique le protocole utilisé dans le paquet soumis à la règle. Valeurs courantes : udp, tcp, icmp.
--dport *port*	Facultatif : indique le port de destination du paquet soumis à la règle.
-j *action*	Indique comment traiter le paquet soumis à la règle (ACCEPT ou DROP).

Autorisation des ping sortants et entrants

Chaque type de flux doit faire l'objet d'une règle iptables.

```
alpha:~# iptables -A OUTPUT -p icmp -j ACCEPT
alpha:~# iptables -A INPUT -p icmp -j ACCEPT
```

Autorisation du trafic http traversant en provenance d'un réseau

```
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp -dport 80 -j ACCEPT
alpha:~#
```

Une configuration erronée sur un pare-feu peut avoir des conséquences dramatiques. Il est recommandé pour vérifier sa bonne configuration d'utiliser un scanner de ports depuis une machine distante. La commande nmap -F suivie de l'adresse IP de la machine protégée permet de vérifier très rapidement (Fastmode) que les ports sont bien bloqués ou ouverts.

c. Gestion des règles

Les règles sont appliquées dans leur ordre de création et le système leur applique automatiquement un numéro d'ordre.

Affichage des numéros de règles effectives

iptables -L chaine --line-numbers -n

Suppression d'une règle

iptables -D chaine numéro

Où *numéro* représente le numéro de la ligne obtenu avec la commande précédente.

Insertion d'une règle

iptables -I chaine numéro conditions -j action

Où *conditions* représente les critères de sélection du paquet soumis à la règle (adresses IP, ports et protocoles).

Exemple de gestion de règles

La gestion dynamique des règles est tellement pénible que l'usage établi veut plutôt que l'on exploite un fichier de script comprenant toutes les règles, et qu'on le recharge complètement après modification.

```
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp
dpt:23
2    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp
dpt:53
3    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp
dpt:80
alpha:~# iptables -D FORWARD 1
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp
dpt:53
2    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp
dpt:80
alpha:~# iptables -I FORWARD 1 -s 192.168.1.0/24 -p tcp --dport 22 -j
ACCEPT
alpha:~# iptables -L FORWARD --line-numbers -n
Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp
dpt:22
2    ACCEPT     udp  --  192.168.1.0/24       0.0.0.0/0           udp
dpt:53
3    ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           tcp
dpt:80
alpha:~#
```

d. Gestion des flux retour

Dans la plupart des applications réseau, un hôte envoie un paquet à destination d'un autre qui lui répond. On a donc une communication à double sens. Or, dans la configuration d'un pare-feu, on visualise bien les flux aller : par exemple, depuis un navigateur vers un serveur web sur le port 80, mais moins bien les réponses des serveurs qui se font sur un port aléatoire à l'initiative du client supérieur à 1024.

Dans les premiers âges des pare-feu, la solution consistait à autoriser tout trafic entrant dont le port était supérieur à 1024. Les pare-feu avaient alors davantage vocation à empêcher les gens de sortir plutôt que d'éviter les intrusions dans le réseau.

Depuis quelques années, les pare-feu dits « stateful » (à état) sont capables d'autoriser dynamiquement les flux retours du moment qu'ils sont la réponse à un flux en sortie explicitement autorisé.

Autorisation implicite des flux retour

iptables -A chaine -m state --state ESTABLISHED,RELATED -j ACCEPT

L'option -m state permet de réaliser un filtre en fonction de l'état du paquet traité. Les états acceptés : ESTABLISHED et RELATED représentent respectivement des paquets en réponse à un flux aller autorisé, et des paquets issus d'une nouvelle connexion, mais à l'initiative d'une connexion établie et autorisée (par exemple le trafic de données ftp relatif à un trafic de commandes ftp).

Exemple de configuration complète d'un pare-feu

On configure ici un pare-feu qui ne laisse rien passer, à l'exception des réponses aux trafics établis, ainsi que les protocoles nécessaires à la navigation Internet (http, https et dns).

```
alpha:~# iptables -P INPUT DROP
alpha:~# iptables -P OUTPUT DROP
alpha:~# iptables -P FORWARD DROP
alpha:~# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j
ACCEPT
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp --dport 80 -j ACCEPT
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p tcp --dport 443 -j
ACCEPT
alpha:~# iptables -A FORWARD -s 192.168.1.0/24 -p udp --dport 53 -j ACCEPT
```

Dans cet exemple, on configure un pare-feu qui ne laisse rien passer, à l'exception des réponses aux trafics établis, ainsi que les protocoles nécessaires à la navigation Internet (http, https et dns).

3. Gestion du NAT

a. Rappel sur le principe du NAT

Le NAT consiste à réécrire l'en-tête IP d'un paquet qui passe d'un réseau public vers un réseau privé et inversement.

Les adresses IP publiques étant non routables sur l'Internet, un paquet qui proviendrait d'une adresse privée ne pourrait pas trouver de route retour, parce qu'aucun routeur n'accepterait de le renvoyer chez lui. De toute façon, les réseaux privés étant démultipliés à l'infini (il existe des millions de réseaux 192.168.1.0), il ne serait pas possible de maintenir dans les tables de routage des routeurs d'Internet une route cohérente vers le réseau d'origine.

La solution consiste donc pour sortir d'un réseau privé à remplacer l'adresse IP de l'expéditeur privé par l'adresse IP publique (unique sur Internet) du routeur réalisant le NAT. La traçabilité des translations (remplacement des adresses IP privées) se fait par rapport au port expéditeur utilisé : pour chaque translation réalisée, le routeur garde en mémoire le port expéditeur

employé. Le paquet retour arrivant sur l'adresse publique du routeur et sur le port employé par l'expéditeur, l'adresse originelle du client est facilement retrouvée par le routeur NAT.

b. Diagnostic de la configuration NAT d'un routeur

Le NAT est géré dans une table spécifique appelée **NAT**. Toute configuration touchant au NAT se fera avec la commande **iptables** en précisant qu'on travaille sur la table NAT. Les chaînes traitées dans la table NAT sont **PREROUTING**, **POSTROUTING** et **OUTPUT**, représentant le trafic à modifier avant le routage, après, ou directement en sortie de la machine.

Affichages de la configuration NAT

iptables -t nat -L

iptables -t nat -S

c. Connexion d'un réseau privé à un réseau public

Dans cette configuration qui est aussi la plus courante, l'adresse IP d'expéditeur des hôtes du réseau privé est remplacée par l'adresse publique du routeur NAT.

Configuration du NAT

iptables -t nat -A POSTROUTING -o carte_ext -j action_nat

Nat avec iptables : options et paramètres	
-t nat	La règle concerne la table de NAT.
-A POSTROUTING	On ajoute une règle à la chaîne POSTROUTING, pour un traitement après routage.
-o *carte_ext*	Désigne la carte réseau par laquelle les paquets sortent du pare-feu.
-j *action_nat*	Désigne le mode d'action du NAT, supporte deux options : SNAT si l'adresse publique est fixe, et MASQUERADE si l'adresse publique est dynamique.

Exemple de configuration du NAT

```
alpha:~# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Dans cet exemple, eth1 est l'interface connectée au réseau public.

4. Scripts de configuration des règles de filtrage

a. Red Hat et les iptables

Les systèmes Red Hat et leurs dérivés proposent un service iptables qui permet d'appliquer une configuration de filtrage ou de NAT automatiquement. Le démarrage du service applique la configuration, et son arrêt annule tout filtrage. Ce fonctionnement est extrêmement pratique et permet de gérer un pare-feu Red Hat de façon très confortable.

b. *Création de services personnalisés de pare-feu avec les iptables*

On constate assez vite que la création de règles de filtrage et de NAT avec les iptables a quelque chose de fastidieux. Par conséquent, après avoir déterminé les règles dont on a besoin, on aura tout intérêt à les placer dans un script.

Exemple de script de configuration de pare-feu

Ce type de script dispense d'avoir à gérer les règles une par une en cas de modification de la configuration. Il est beaucoup plus facile d'insérer une ligne dans le script que de décaler la numérotation des règles en mémoire. Toutefois, il faut annuler toute règle avant chaque application du script.

```
#!/bin/bash
# nom du fichier : /etc/parefeu_on
# Politique de base
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# NAT avec eth0 en interne et eth1 en sortie - adresse IP publique fixe
iptables -t nat -A POSTROUTING  -o eth1 -j SNAT --to-source 81.2.3.4
# gestion des paquets retours
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED
-j ACCEPT
# trafic autorisé en sortie
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p udp --dport 53 -j ACCEPT
```

Bien entendu, il ne faudra pas oublier de le rendre exécutable.

Il sera également utile de créer un script d'annulation de toute règle de filtrage. Il peut en effet être utile d'autoriser plus ou moins provisoirement tout trafic, pour une mise à jour du pare-feu ou un usage applicatif ponctuel.

Exemple de script d'annulation de filtrage

```
#!/bin/bash
# nom du fichier : parefeu_off
# Effacement des règles
iptables -F
# Politique permissive
iptables -P INPUT ACCEPT
```

```
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

Enfin, on peut créer un script de gestion de service normalisé.

Exemple de script de service de pare-feu

Ce script est naturellement à placer dans le répertoire /etc/init.d.

```
#!/bin/bash
# nom du fichier : parefeu
case $1 in
start)
  /etc/parefeu_on ;;
stop)
  /etc/parefeu_off  ;;
status)
  iptables -L  ;;
*)
  echo "Syntaxe : /etc/init.d/parefeu start|stop|status ;;
esac
```

III. Détection des intrusions et des vulnérabilités

1. Les systèmes IDS

a. Les limitations des pare-feu

Les pare-feu dans leur fonctionnement historique filtrent les paquets sur les valeurs contenues dans les en-têtes de couche réseau ou transport, et donc sur les adresses IP ou les ports utilisés. Pour contourner la protection apportée par les pare-feu, de nombreuses applications utilisent des ports courants (tcp 80 notamment) pour faire passer leur propre trafic applicatif. Les pare-feu, souvent configurés pour laisser passer les flux sur ces ports courants, n'y voient que du feu.

Pour assurer un meilleur contrôle, il faut utiliser un équipement plus élaboré, capable de regarder et d'analyser le trafic applicatif, directement et sans se faire tromper par l'annonce d'un port erroné. Ces équipements sont appelés « sondes » en français parce que sondant l'intérieur des paquets, ou encore IDS (*Intrusion Detection System*).

b. Techniques d'analyse

Pour identifier les trafics malicieux, les IDS disposent de trois techniques : la détection d'anomalies, l'analyse de protocoles et l'analyse de signatures.

La détection d'anomalies a pour objet de détecter un comportement anormal, comme par exemple un volume ICMP démesuré, qui indiquerait que l'on est la cible ou l'émetteur d'une attaque par déni de service.

L'analyse de protocole ne cherche pas à repérer une action réellement malicieuse, mais plutôt un trafic applicatif qui ne respecterait pas à la lettre les règles de fonctionnement des protocoles employés. C'est un peu l'histoire du braqueur de banque qui se fait arrêter bêtement parce que ses pneus sont lisses.

Enfin, l'analyse de signatures permet d'identifier des attaques ou comportements malsains déjà référencés. C'est la technique la plus efficace et qui n'est pas sujette à erreur, puisqu'on ne gère que des attaques ou intrusions ayant déjà eu lieu chez un tiers, et donc dûment identifiées.

c. Sources d'information

Les techniques d'analyse, qu'il s'agisse d'analyse de signatures, de protocoles ou de détections d'anomalies s'appuient sur des informations qui évoluent avec le temps. Il est évident que l'analyse de signature ne peut s'appliquer que si l'IDS connaît la signature de l'attaque en cours. De plus, la nature des menaces peut évoluer. Par exemple, un hôte qui aurait envoyé de gros

volumes de trafics SMTP dans les années 80 indiquerait qu'un serveur de messagerie fonctionne bien. La même situation aujourd'hui pourrait montrer que l'hôte en question est infecté par un cheval de Troie et qu'il envoie de gros volumes de SPAM.

Les IDS doivent impérativement récupérer à intervalle régulier les mises à jour de leurs techniques d'analyse ainsi que les bases de signatures. Les éditeurs d'IDS doivent systématiquement maintenir leurs bases d'informations à jour, et les administrateurs des IDS doivent tout aussi régulièrement télécharger ces bases.

De nombreux organismes, associations et entreprises permettent de se tenir au courant des évolutions en matière de techniques d'intrusion et de nuisance. Il est recommandé de connaître l'existence des principaux, et dans le cadre d'une administration réseau avec prise en compte de la sécurité, d'assurer une veille technologique sur ces domaines.

2. OpenVAS

OpenVAS (*Open Vulnerability Assessment Scanner*) est une variante libre du scanner de vulnérabilités Nessus.

a. Le serveur OpenVAS

Le serveur est le cœur de la suite applicative OpenVAS, il scanne et analyse les hôtes du réseau à la recherche de vulnérabilités connues (NVT : *Network Vulnerability Tests*).

b. Les clients OpenVAS

Les clients OpenVAS sont des éléments logiciels en ligne de commande ou avec une interface graphique qui assurent l'analyse des hôtes du réseau à la recherche de vulnérabilités pour renvoyer les résultats au serveur.

c. Récupération des vulnérabilités

OpenVas propose une source publique de vulnérabilités connues sous le nom OpenVas NVT Feed. Il permet aux serveurs de se tenir au courant des dernières vulnérabilités connues, et contient plus de 15000 NVT.