Sécurité

Un système d'information a, selon les cas, une importance variable ; dans certains cas, il est vital à la survie d'une entreprise. Il doit donc être protégé en conséquence contre divers risques, ce que l'on regroupe communément sous l'appellation de sécurité.

SOMMAIRE

MOTS-CLÉS

- Pare-feu
- Netfilter
- IDS/NIDS

Définir une politique de sécurité

Le terme de « sécurité » recouvre une vaste étendue de concepts, d'outils et de procédures, qui ne s'appliquent pas à tous les cas. Il convient de s'interroger sur ce que l'on souhaite accomplir pour choisir lesquels mettre en œuvre. Pour sécuriser un système, il faut se poser quelques questions ; si l'on se lance tête baissée dans la mise en œuvre d'outils, on risque de se focaliser sur certains aspects au détriment des plus importants.

Il est donc crucial de se fixer un but. Pour cela, il s'agit d'apporter des réponses aux questions suivantes :

- Que cherche-t-on à protéger ? La politique de sécurité à mener ne sera pas la même selon que l'on cherche à protéger les ordinateurs ou les données. Et s'il s'agit des données, il faudra également se demander lesquelles.
- Contre quoi cherche-t-on à se protéger ? Est-ce d'un vol de données confidentielles ? De la perte accidentelle de ces données ? De la perte de revenu associée à une interruption de service ?
- Également, de qui cherche-t-on à se protéger ? Les mesures de sécurité à mettre en place différeront largement selon que l'on cherche à se prémunir d'une faute de frappe d'un utilisateur habituel du système ou d'un groupe d'attaquants déterminés.

Il est d'usage d'appeler « risque » la conjonction des trois facteurs : ce qui doit être protégé, ce qu'on souhaite éviter, et les éléments qui essaient de faire en sorte que cela arrive. La réunion des réponses à ces trois questions permet de modéliser ce risque. De cette modélisation découlera une politique de sécurité, qui se manifestera à son tour par des actions concrètes.

Il faudra enfin prendre en compte les contraintes qui peuvent limiter la liberté d'action. Jusqu'où est-on prêt à aller pour sécuriser le système ? Cette question a un impact majeur, et la réponse apportée est trop souvent formulée en seuls termes de coût, alors qu'il faut également se demander jusqu'à quel point la politique de sécurité peut incommoder les utilisateurs du système, ou en dégrader les performances, par exemple.

Une fois que l'on a établi une modélisation du risque dont on cherche à se prémunir, on peut se pencher sur la définition d'une politique de sécurité.

Dans la plupart des cas, on s'apercevra que le système informatique peut être segmenté en sous-ensembles cohérents plus ou moins indépendants. Chacun de ces sous-systèmes pourra avoir ses besoins et ses contraintes propres, il faudra donc en général les considérer séparément lors de la définition des politiques de sécurité correspondantes. Il conviendra alors de toujours garder à l'esprit le principe selon lequel un périmètre court et bien défini est plus facile à défendre qu'une frontière vague et longue. L'organi-

ATTENTION Portée de ce chapitre

La sécurité est un sujet vaste et sensible, que nous ne saurions traiter complètement dans le cadre d'un seul chapitre. Nous nous limiterons ici à délimiter quelques points importants et présenter quelques-uns des outils et méthodes qui peuvent servir dans le domaine, mais la littérature est abondante et des ouvrages entiers ont été écrits sur le sujet. On pourra par exemple se rapporter à l'ouvrage *Sécuriser un réseau Linux* de Bernard Boutherin et Benoît Delaunay (collection Cahiers de l'Admin, éditions Eyrolles) ; à *Sécurité informatique, principes et méthode* de Laurent Bloch et Christophe Wolfhugel (collection blanche, éditions Eyrolles également) ; ou, en anglais, à l'excellent *Linux Server Security* de Michael D. Bauer (éditions O'Reilly).

NOTE Remise en question permanente

Bruce Schneier, un des experts mondialement reconnus en matière de sécurité (pas uniquement informatique, d'ailleurs) lutte contre un des mythes importants de la sécurité par l'expression « La sécurité est un processus, non un produit. » Les actifs à protéger évoluent au fil du temps, de même que les menaces qui pèsent dessus et les moyens dont disposent les attaquants potentiels. Il est donc important, même si une politique de sécurité a été parfaitement conçue et mise en œuvre, de ne pas s'endormir sur ses lauriers. Les composants du risque évoluant, la réponse à apporter à ce risque doit également évoluer à leur suite.

NOTE **Politiques extrêmes**

Dans certains cas, le choix des actions à mener pour sécuriser un système peut être extrêmement simple.
Par exemple, si le système à protéger se compose exclusivement d'un ordinateur de récupération qu'on n'utilise que pour additionner des chiffres en fin de journée, on peut tout à fait raisonnablement décider de ne rien faire de spécial pour le protéger, puisque la valeur intrinsèque du système est faible, celle des données est nulle puisqu'elles ne sont pas stockées sur cet ordinateur, et qu'un attaquant potentiel ne gagnerait à s'infiltrer sur ce « système » qu'une calculatrice un peu encombrante. Le coût de la sécurisation d'un tel système dépasserait probablement largement celui du risque.
À l'opposé, si l'on cherche à protéger absolument la confidentialité de données secrètes, au détriment de toute autre considération, une réponse appropriée serait la destruction complète de ces données (avec effacement des fichiers, puis pulvérisation des disques durs, dissolution dans de l'acide, etc.). Si les données doivent de plus être préservées, mais pas nécessairement accessibles, et que le coût n'est pas soumis à des contraintes, on pourra commencer en stockant ces données gravées sur des plaques de platine iridié stockées en divers bunkers répartis sous différentes montagnes dans le monde, chacun étant bien entendu entièrement secret mais gardé par des armées entières…
Pour extrêmes qu'ils puissent paraître, ces deux exemples n'en sont pas moins des réponses adaptées à des risques définis, dans la mesure où ils découlent d'une réflexion qui prend en compte les buts à atteindre et les contraintes présentes. Lorsqu'il s'agit d'une décision raisonnée, aucune politique de sécurité n'est moins respectable qu'une autre.

sation du réseau devra donc être pensée en conséquence, afin que les services les plus sensibles soient concentrés sur un petit nombre de machines, et que ces machines ne soient accessibles qu'à travers un nombre minimal de points de passage, qui seront plus faciles à sécuriser que s'il faut défendre chacune des machines contre l'intégralité du monde extérieur. On voit clairement apparaître ici l'utilité des solutions de filtrage du trafic réseau, notamment par des pare-feu. On pourra pour cela utiliser du matériel dédié, mais une solution peut-être plus simple et plus souple est d'utiliser un pare-feu logiciel, tel que celui intégré dans le noyau Linux.

Pare-feu ou filtre de paquets

Un pare-feu est une passerelle filtrante : il applique des règles de filtrage aux paquets qui le traversent (c'est pourquoi il n'est utile qu'en tant que point de passage obligé).

L'absence de configuration standard explique qu'il n'y ait pas de solution prête à l'emploi. Des outils permettent en revanche de simplifier la configuration du pare-feu netfilter en visualisant graphiquement les règles définies. L'un des meilleurs est sans doute **`fwbuilder`**.

B.A.-BA **Pare-feu**

Un pare-feu (*firewall*) est un ensemble matériel ou logiciel qui trie les paquets qui circulent par son intermédiaire en provenance ou vers le réseau local, et ne laisse passer que ceux qui vérifient certaines conditions.

CAS PARTICULIER Pare-feu local

Un pare-feu peut limiter son action à une seule machine (et non pas un réseau local complet) ; son rôle principal est alors de refuser ou limiter l'accès à certains services, voire de se prémunir contre l'établissement de connexions sortantes par des logiciels indésirables que l'utilisateur pourrait avoir installés (volontairement ou pas).

Les noyaux Linux des séries 2.4 et 2.6 intègrent le pare-feu netfilter, que l'outil **`iptables`** permet de configurer.

Fonctionnement de netfilter

netfilter dispose de trois tables distinctes, donnant les règles régissant trois types d'opérations sur les paquets :

- `filter` pour les règles de filtrage (accepter, refuser, ignorer un paquet) ;
- `nat` pour modifier les adresses IP et les ports source ou destinataires des paquets ;
- `mangle` pour modifier d'autres paramètres des paquets IP (notamment le champ ToS — *Type Of Service* — et les options).

Chaque table contient des listes de règles appelées chaînes ; les chaînes standards servent au pare-feu pour traiter les paquets dans différentes circonstances prédéfinies. L'administrateur peut créer d'autres chaînes, qui ne seront employées que si l'une des chaînes standard les appelle.

La table `filter` compte trois chaînes standard :

- `INPUT` : concerne les paquets destinés au pare-feu ;
- `OUTPUT` : concerne les paquets émis par le pare-feu ;
- `FORWARD` : appliquée aux paquets transitant via le pare-feu (et dont il n'est donc ni la source ni le destinataire).

La table `nat` dispose également de trois chaînes standards :

- `PREROUTING` : modifie les paquets dès qu'ils arrivent ;
- `POSTROUTING` : modifie les paquets alors qu'ils sont prêts à partir ;
- `OUTPUT` : modifie les paquets générés par le pare-feu lui-même.

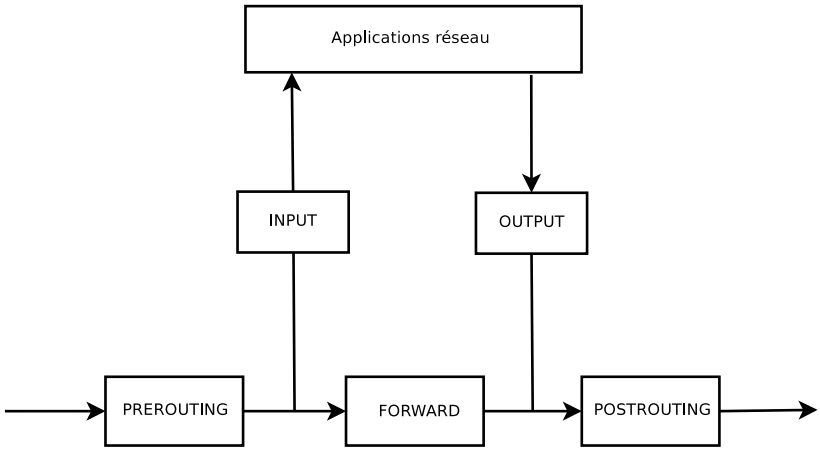

Figure 14–1
Ordre d'emploi des chaînes de netfilter

Chaque chaîne est une liste de règles, prévoyant une action à exécuter quand certaines conditions sont remplies. Le pare-feu parcourt séquentiellement la chaîne s'appliquant au paquet traité, et dès qu'une règle est satisfaite, il « saute » (l'option `-j` vient de *jump*) à l'emplacement indiqué pour continuer le traitement. Certains de ces emplacements sont standardisés et correspondent aux actions les plus courantes. Une fois une de ces actions enclenchée, le parcours de la chaîne est interrompu parce que le sort du paquet est normalement décidé (sauf exception explicitement mentionnée ci-dessous) :

B.A.-BA **ICMP**

ICMP (*Internet Control Message Protocol*, ou protocole des messages de contrôle sur Internet) est très employé pour transmettre des compléments d'information sur les communications : il permet de tester le fonctionnement du réseau avec la commande **`ping`** (qui envoie un message ICMP *echo request* auquel le correspondant est normalement tenu de répondre par un message *echo reply*), signale le refus d'un paquet par un pare-feu, indique la saturation d'un tampon de réception, propose une meilleure route (un meilleur trajet pour les prochains paquets à émettre), etc. Plusieurs RFC définissent ce protocole ; les premières, 777 et 792, furent rapidement complétées et étendues.

- http://www.faqs.org/rfcs/rfc777.html
- http://www.faqs.org/rfcs/rfc792.html

Rappelons qu'un tampon de réception est une petite zone mémoire contenant les données reçues par le réseau avant qu'elles ne soient traitées par le noyau. Si cette zone est pleine, il est alors impossible de recevoir d'autres données et ICMP signale le problème de sorte que le correspondant réduise la vitesse de transfert pour essayer d'atteindre un équilibre.

- `ACCEPT` : autoriser le paquet à poursuivre son parcours ;
- `REJECT` : rejeter le paquet (ICMP signale une erreur, l'option `--reject-with` *type* d'**`iptables`** permet de choisir le type d'erreur renvoyée) ;
- `DROP` : supprimer (ignorer) le paquet ;
- `LOG` : enregistrer (via **`syslogd`**) un message de log contenant une description du paquet traité (cette action retourne après exécution à sa position dans la chaîne appelante — celle qui a invoquée l'action — c'est pourquoi il est nécessaire de la faire suivre par une règle REJECT ou DROP si l'on veut simplement enregistrer la trace d'un paquet qui doit être refusé) ;
- `ULOG` : enregistrer un message de log via **`ulogd`**, plus adapté et plus efficace que **`syslogd`** pour gérer de grandes quantités de messages (cette action renvoie aussi le fil d'exécution à sa position dans la chaîne appelante) ;
- *nom_de_chaîne* : évaluer les règles de la chaîne indiquée ;

- `RETURN` : stopper l'évaluation de la chaîne courante et revenir sur la chaîne appelante (si la chaîne courante est une chaîne standard, dépourvue de chaîne appelante, effectuer l'action par défaut — il s'agit d'une action particulière qui se configure avec l'option `-P` de **`iptables`**) ;
- `SNAT` : effectuer du *Source NAT* (des options précisent les modifications à effectuer) ;
- `DNAT` : effectuer du *Destination NAT* (des options précisent les modifications à effectuer) ;
- `MASQUERADE` : effectuer du **`masquerading`** (SNAT particulier) ;
- `REDIRECT` : rediriger un paquet vers un port particulier du pare-feu lui-même ; action notamment utile pour mettre en place un mandataire (ou proxy) web transparent (il s'agit d'un service pour lequel aucune configuration n'est nécessaire, puisque le client a l'impression de se connecter directement au destinataire alors que ses échanges avec le serveur transitent systématiquement par le mandataire).

D'autres actions, concernant davantage la table `mangle`, ne sont pas mentionnées ici. Vous en trouverez la liste exhaustive dans la page de manuel `iptables(8)`.

Syntaxe d'iptables

La commande **`iptables`** permet de manipuler les tables, les chaînes et les règles. L'option `-t` *`table`* indique la table sur laquelle opérer (par défaut, c'est `filter`).

Les commandes

L'option `-N` *`chaîne`* crée une nouvelle chaîne ; l'option `-X` *`chaîne`* supprime une chaîne vide et inutilisée. L'option `-A` *`chaîne règle`* ajoute une règle à la fin de la chaîne indiquée. L'option `-I` *`chaîne numrègle règle`* insère une règle avant la règle numérotée *numrègle*. L'option `-D` *`chaîne numrègle`* ou `-D` *`chaîne règle`* supprime une règle dans la chaîne (la première syntaxe l'identifie par son numéro et la seconde par son contenu). L'option `-F` *`chaîne`* supprime toutes les règles de la chaîne (si celle-ci n'est pas mentionnée, elle supprime toutes les règles de la table). L'option `-L` *`chaîne`* affiche le contenu de la chaîne. Enfin, l'option `-P` *`chaîne action`* définit l'action par défaut pour la chaîne donnée (seules les chaînes standards peuvent en avoir une).

Les règles

Chaque règle s'exprime sous la forme *conditions* `-j` *action options_de_l'action*. En écrivant bout à bout plusieurs conditions dans la même règle, on en produit la conjonction (elles sont liées par des *et* logiques), donc une condition plus restrictive.

La condition `-p` *protocole* sélectionne selon le champ protocole du paquet IP, dont les valeurs les plus courantes sont `tcp`, `udp`, et `icmp`. Préfixer le protocole par un point d'exclamation inverse la condition (qui correspond alors à tous les paquets n'ayant pas le protocole indiqué). Cette manipulation est possible pour toutes les autres conditions énoncées ci-dessous.

La condition `-s` *adresse* ou `-s` *réseau/masque* vérifie l'adresse source du paquet ; `-d` *adresse* ou `-d` *réseau/masque* en est le pendant pour l'adresse de destination.

La condition `-i` *interface* sélectionne les paquets provenant de l'interface réseau indiquée ; `-o` *interface* sélectionne les paquets en fonction de leur interface réseau d'émission.

D'autres conditions plus spécifiques existent, qui dépendent des conditions génériques déjà définies. La condition `-p tcp` peut par exemple être accompagnée de conditions sur les ports TCP avec `--source-port` *port* et `--destination-port` *port*.

L'option `--state` *état* indique le statut du paquet dans une connexion (le module **`ipt_conntrack`**, qui implémente le suivi des connexions, lui est nécessaire). L'état `NEW` désigne un paquet qui débute une nouvelle connexion. L'état `ESTABLISHED` concerne les paquets d'une connexion existante et l'état `RELATED` les paquets d'une nouvelle connexion liée à une connexion existante (c'est le cas des connexions `ftp-data` d'une session `ftp`).

La section précédente détaille la liste des actions possibles, mais pas les options qui leur sont associées. L'action `LOG` dispose ainsi de plusieurs options visant à :

- indiquer la priorité du message à **`syslog`** (`--log-priority`, de valeur par défaut `warning`) ;
- préciser un préfixe textuel pour différencier les messages (`--log-prefix`) ;
- indiquer les données à intégrer dans le message (`--log-tcp-sequence` pour le numéro de séquence TCP, `--log-tcp-options` pour les options TCP et `--log-ip-options` pour les options IP).

L'action `DNAT` dispose de l'option `--to-destination` *adresse*:*port* pour indiquer la nouvelle adresse IP et/ou le nouveau port de destination. De la même manière, l'action `SNAT` dispose de l'option `--to-source` *adresse*:*port* pour indiquer la nouvelle adresse et/ou le nouveau port source.

L'action `REDIRECT` dispose de l'option `--to-ports` *`port(s)`* pour indiquer le port ou l'intervalle de ports vers lesquels rediriger les paquets.

Créer les règles

Il faut invoquer **`iptables`** une fois par règle à créer ; c'est pourquoi on consigne habituellement tous les appels à cette commande dans un fichier de script pour mettre en place la même configuration à chaque redémarrage de la machine. On peut écrire ce script à la main mais il est souvent intéressant de le préparer à l'aide d'un outil de plus haut niveau, tel que **`fwbuilder`**.

Son principe est simple. Dans une première étape, il faut décrire tous les éléments susceptibles d'intervenir dans les différentes règles :

- le pare-feu et ses interfaces réseau ;
- les réseaux (et plages d'IP associées) ;
- les serveurs ;
- les ports correspondant aux services hébergés sur les différents serveurs.

On crée ensuite les règles par simple glisser/déposer des différents objets, quelques menus contextuels permettant de modifier la condition (l'inverser, par exemple). Il ne reste qu'à saisir l'action souhaitée et à la paramétrer.

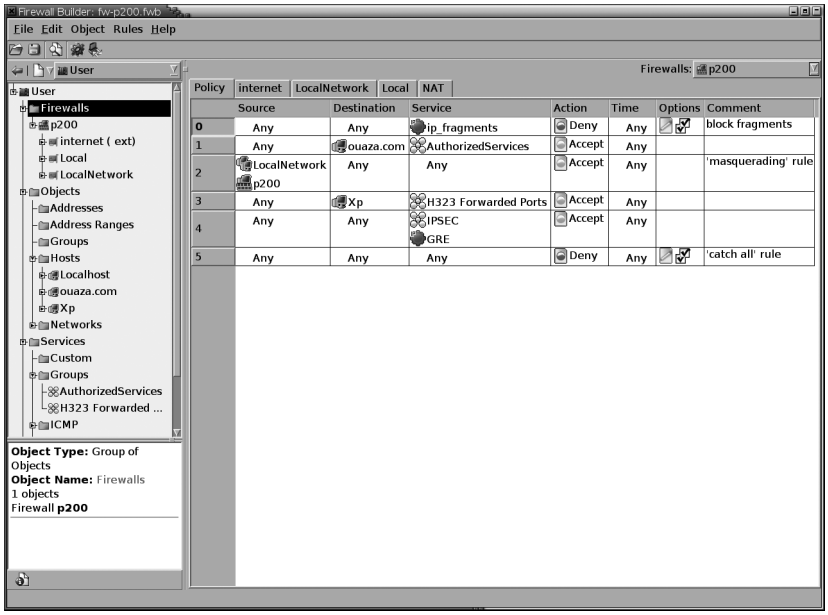

Figure 14–2
Fwbuilder en action

fwbuilder peut alors générer un script de configuration du pare-feu selon les règles saisies. Son architecture modulaire lui permet de générer des scripts pour les pare-feu de différents systèmes (**iptables** pour Linux 2.4/2.6, **ipchains** pour Linux 2.2, **ipf** pour FreeBSD et **pf** pour OpenBSD).

Le paquet *fwbuilder* contient l'interface graphique tandis que *fwbuilder-linux* contient les modules pour les pare-feu Linux (et notamment celui dédié à **iptables**, que l'on souhaite employer pour configurer notre pare-feu netfilter) :

```
# aptitude install fwbuilder fwbuilder-linux
```

Installer les règles à chaque démarrage

Si le pare-feu doit protéger une connexion réseau intermittente par PPP, le plus simple est de changer le nom du script de configuration du pare-feu et de l'installer sous `/etc/ppp/ip-up.d/0iptables` (attention, le nom de fichier ne doit pas contenir de point, sinon il ne sera pas pris en compte). Ainsi, il sera rechargé à chaque démarrage d'une connexion PPP.

Dans les autres cas, le plus simple est d'inscrire le script de configuration du pare-feu dans une directive `up` du fichier `/etc/network/interfaces`. Dans l'exemple ci-dessous, ce script s'appelle `/usr/local/etc/arrakis.fw`.

EXEMPLE Fichier interfaces avec appel du script de pare-feu

```
auto eth0
iface eth0 inet static
    address 192.168.0.1
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    up /usr/local/etc/arrakis.fw
```

Supervision : prévention, détection, dissuasion

La supervision fait partie intégrante d'une politique de sécurité. Elle est nécessaire à plusieurs titres : l'objectif de la sécurité n'est pas uniquement de garantir la confidentialité des données, mais aussi de pouvoir assurer le bon fonctionnement des services. Il est donc impératif de veiller que tout fonctionne comme prévu et de détecter au plus tôt les comportement inhabituels et les changements dans la qualité du service fourni. Surveiller l'activité peut permettre de détecter des tentatives d'intrusion

et donc de s'en protéger avant que cela ne porte à conséquences. Ce chapitre va donc passer en revue des outils permettant de surveiller différents aspects d'un système Debian. Il complète la section dédiée à la supervision du chapitre 12 « Administration avancée ».

Surveillance des logs avec logcheck

Le programme **logcheck** scrute par défaut les fichiers de logs toutes les heures et envoie par courrier électronique à `root` les messages les plus inhabituels pour aider à détecter tout nouveau problème.

La liste des fichiers scrutés se trouve dans le fichier `/etc/logcheck/logcheck.logfiles` ; les choix par défaut conviendront si le fichier `/etc/syslog.conf` n'a pas été complètement remodelé.

logcheck peut fonctionner en 3 modes plus ou moins détaillés : *paranoid* (paranoïaque), *server* (serveur), et *workstation* (station de travail). Le premier étant le plus verbeux, on le réservera aux serveurs spécialisés (comme les pare-feu). Le deuxième mode, choisi par défaut, est recommandé pour les serveurs. Le dernier, prévu pour les stations de travail, élimine encore plus de messages.

Dans tous les cas, il faudra probablement paramétrer **logcheck** pour exclure des messages supplémentaires (selon les services installés) sous peine d'être envahi chaque heure par une flopée de messages inintéressants. Leur mécanisme de sélection étant relativement complexe, il faut lire à tête reposée le document `/usr/share/doc/logcheck-database/README.logcheck-database.gz` pour bien le comprendre.

Plusieurs types de règles sont appliquées :

- celles qui qualifient un message comme résultant d'une tentative d'attaque (elles sont stockées dans un fichier du répertoire `/etc/logcheck/cracking.d/`) ;
- celles qui annulent cette qualification (`/etc/logcheck/cracking.ignore.d/`) ;
- celles qui qualifient un message comme une alerte de sécurité (`/etc/logcheck/violations.d/`) ;
- celles qui annulent cette qualification (`/etc/logcheck/violations.ignore.d/`) ;
- et enfin celles qui s'appliquent à tous les messages restants (les *System Events*, ou événements système).

ATTENTION Ignorer un message

Tout message marqué comme une tentative d'attaque ou une alerte de sécurité (suite par exemple à une règle du fichier `/etc/logcheck/violations.d/monfichier`) ne pourra être ignoré que par une règle des fichiers `/etc/logcheck/violations.ignore.d/monfichier` ou `/etc/logcheck/violations.ignore.d/monfichier-`*`extension`*.

Un événement système sera systématiquement signalé, sauf si une règle de l'un des répertoires `/etc/logcheck/ignore.d.{paranoid,server,workstation}/` dicte de l'ignorer. Évidemment, seuls les répertoires correspondant à des niveaux de verbosité supérieurs ou égaux au niveau sélectionné sont pris en compte.

ASTUCE
Vos logs en fond d'écran

Certains administrateurs aiment voir les messages de logs défiler en temps réel. Ils pourront les intégrer dans le fond d'écran de leur bureau graphique avec la commande **`root-tail`** (du paquet Debian éponyme). Le programme **`xconsole`** (du paquet *xbase-clients*) les fera défiler dans une petite fenêtre ; les messages sont directement issus de **`syslogd`** par l'intermédiaire du tube nommé `/dev/xconsole`.

Surveillance de l'activité

En temps réel

`top` est un utilitaire interactif qui affiche la liste des processus en cours d'exécution. Par défaut, son critère de tri est l'utilisation actuelle du processeur (touche *P*) mais on peut opter pour la mémoire occupée (touche *M*), le temps processeur consommé (touche *T*) ou le numéro de processus ou PID (touche *N*). La touche *k* (comme *kill*) permet d'indiquer un numéro de processus à tuer. *r* (comme *renice*) permet de changer la priorité d'un processus.

Si le processeur semble être surchargé, il est ainsi possible d'observer quels processus se battent pour son contrôle ou consomment toute la mémoire disponible. Il est intéressant en particulier de vérifier si les processus qui consomment des ressources correspondent effectivement aux services réels que la machine héberge. Un processus au nom inconnu tournant sous l'utilisateur www-data doit immédiatement attirer l'attention : la probabilité est forte que cela corresponde à un logiciel installé et exécuté sur la machine en exploitant une faille de sécurité d'une application web...

`top` est un outil de base très souple, et sa page de manuel explique comment en personnaliser l'affichage pour l'adapter aux besoins et aux habitudes de chacun.

`gnome-system-monitor` et **`qps`**, outils graphiques similaires à **`top`**, en proposent les principales fonctionnalités.

ASTUCE
Représentations visuelles de l'activité

Pour des représentations plus visuelles (et plus amusantes) de l'activité de l'ordinateur, on pourra envisager d'installer les paquets `lavaps`, `bubblemon` et `bubblefishymon`. Le premier contient **`lavaps`**, qui représente les processus courants comme les bulles de cire d'une lampe-fusée ; *bubblemon* est une applet pour les panneaux de contrôle des environnements de bureau, qui représente la mémoire utilisée et le taux d'occupation du processeur sous forme d'un aquarium où flottent des bulles ; enfin, **`bubblefishymon`** reprend le même principe, mais y ajoute le trafic réseau sous forme de poissons (et même un canard !).

Historique

La charge du processeur, le trafic réseau ou l'espace disque disponible sont des informations qui varient en permanence. Il est souvent intéressant de garder une trace de leur évolution pour mieux cerner l'usage qui est fait de l'ordinateur.

Il existe de nombreux outils dédié à cette tâche. La plupart peuvent récupérer des données via SNMP (*Simple Network Management Protocol*, ou protocole simple de gestion du réseau) afin d'avoir une centralisation de ces informations. Cela permet en outre de récupérer des informations sur des éléments du réseau qui ne sont pas nécessairement des ordinateurs (comme des routeurs).

ALTERNATIVE **mrtg**

mrtg (du paquet Debian éponyme) est un outil plus ancien et plus rustique capable d'agréger des données historiques et d'en faire des graphiques. Il dispose d'un certain nombre de scripts de récupération des données les plus couramment surveillées : charge, trafic réseau, impacts (*hits*) web, etc.

Les paquets `mrtg-contrib` et `mrtgutils` contiennent des scripts d'exemples, prêts à l'emploi.

Ce livre traite en détail de Munin (voir page 293) dans le cadre du chapitre « Administration avancée ». Debian dispose également de `cacti`. Il est un peu plus complexe à mettre en œuvre : l'usage de SNMP est inévitable et malgré une interface web, les concepts de configuration restent difficiles à appréhender. La lecture de la documentation HTML (`/usr/share/doc/cacti/html/index.html`) sera indispensable si l'on souhaite le mettre en œuvre.

Détection des changements

Une fois le système installé et configuré, l'état de la majorité des fichiers et répertoires (hors données) n'a pas de raison d'évoluer (sauf mises à jour de sécurité). Il est donc intéressant de s'assurer que c'est bien le cas : tout changement inattendu est alors suspect. Les outils présentés dans cette section permettent de surveiller tous les fichiers et de prévenir les administrateurs en cas d'altération inattendue, ou alors simplement de diagnostiquer l'étendue des altérations.

Audit des paquets : l'outil debsums et ses limites

POUR ALLER PLUS LOIN

Se protéger des modifications en amont

debsums peut être utilisé pour détecter les changements effectués sur les fichiers provenant d'un paquet Debian. Mais si le paquet Debian lui-même est compromis, il ne sera d'aucune utilité. Cela pourrait être le cas si le miroir Debian employé est lui-même compromis. Pour se protéger de ces attaques, il faut s'appuyer sur le mécanisme de vérification de signatures numériques intégré à APT (voir page 102) et prendre soin de n'installer que des paquets dont l'origine a pu être certifiée.

debsums est un outil intéressant du point de vue de la sécurité puisqu'il permet de trouver facilement quels fichiers installés ont été modifiés (suite par exemple à des interventions malignes). Mais il convient de nuancer fortement cette affirmation : d'abord, tous les paquets Debian ne fournissent pas les empreintes nécessaires au fonctionnement de ce programme (quand elles existent, elles se trouvent dans un fichier `/var/lib/dpkg/info/`*paquet*`.md5sums`). Rappelons qu'une empreinte est une valeur, généralement numérique (même si elle est codée en hexadécimal), constituant une sorte de signature caractéristique du contenu d'un fichier. Elle est calculée au moyen d'algorithmes (comme le célèbre MD5 ou le moins connu SHA1) qui garantissent dans la pratique que (presque) toute modification du fichier, aussi minime soit-elle, entraînera un changement de l'empreinte ; c'est l'« effet d'avalanche ». C'est pourquoi une empreinte numérique permet de vérifier que le contenu d'un fichier n'a pas été altéré. Ces algorithmes ne sont pas réversibles, c'est-à-dire que pour la plupart d'entre eux il est impossible de retrouver un contenu inconnu à partir de la seule empreinte. De récentes découvertes scientifiques tendent à infirmer l'inviolabilité de ces principes, mais cela ne remet pas encore en cause leur usage puisque la création de contenus différents générant la même empreinte semble être très contraignante.

D'autre part, les fichiers `md5sums` sont stockés sur le disque dur : un intrus consciencieux modifiera ces fichiers pour leur faire refléter les nouvelles sommes de contrôle des fichiers sur lesquels il sera intervenu.

On peut contourner le premier inconvénient en demandant à **debsums** d'utiliser directement un paquet `.deb` pour effectuer le contrôle au lieu de se reposer sur le fichier `md5sums`. Mais il faut au préalable télécharger les fichiers `.deb` correspondants :

```
# apt-get --reinstall -d install `debsums -l`
[ ... ]
# debsums -p /var/cache/apt/archives -g
```

L'autre souci se contourne de la même manière : il suffit d'effectuer la vérification par rapport à un fichier `.deb` intègre. Mais cela impose de disposer de tous les fichiers `.deb` des paquets installés et d'être assuré de leur intégrité. Pour cela, le plus simple est de les reprendre depuis un miroir Debian. Cette opération étant plutôt lente et fastidieuse, ce n'est donc pas une technique à suivre systématiquement dans un but de prévention.

```
# apt-get --reinstall -d install `grep-status -e 'Status: install ok
  ➥ installed' -n -s Package`
[ ... ]
# debsums -p /var/cache/apt/archives --generate=all
```

Attention, cet exemple a employé la commande **grep-status** du paquet `grep-dctrl`, qui n'est pas installé en standard.

Surveillance des fichiers : AIDE

AIDE (*Advanced Intrusion Detection Environment*) est un outil qui permet de vérifier l'intégrité des fichiers et de détecter toute altération par rapport à une image du système préalablement enregistrée et validée. Cette dernière prend la forme d'une base de données (`/var/lib/aide/aide.db`) contenant les caractéristiques de tous les fichiers du système (permissions, horodatages, empreintes numériques, etc.). Cette base de données est initialisée une première fois par **aideinit** ; elle est ensuite employée pour vérifier quotidiennement (script `/etc/cron.daily/aide`) que rien n'a changé. Si des changements sont détectés, le logiciel les enregistre dans des fichiers de journalisation (`/var/log/aide/*.log`) et envoie un courrier à l'administrateur avec ses découvertes.

Le comportement du paquet `aide` se paramètre grâce à de nombreuses options dans `/etc/default/aide`. La configuration du logiciel proprement dit se trouve dans `/etc/aide/aide.conf` et `/etc/aide/aide.conf.d/` (en réalité ces fichiers servent de base à **update-aide.conf** pour créer `/var/lib/aide/aide.conf.autogenerated`). La configuration indique quelles propriétés de chaque fichier il faut vérifier. Ainsi le contenu des fichiers de logs peut varier tant que les permissions associées ne varient pas, mais le contenu et les permissions d'un exécutable doivent être fixes. La syntaxe n'est pas très compliquée mais elle n'est pas forcé-

EN PRATIQUE
Protection de la base de données

Puisque AIDE utilise une base de données pour comparer l'état des fichiers, il faut être conscient que la validité des résultats fournis dépend de la validité de la base de données. Sur un système compromis, un attaquant obtenant les droits root pourra remplacer la base de données et passer inaperçu. C'est pourquoi, pour plus de sécurité, il peut être intéressant de stocker la base de données de référence sur un média accessible en lecture seulement.

ALTERNATIVE **Tripwire**

Tripwire est très similaire à AIDE, la syntaxe de son fichier de configuration est quasiment identique. Le paquet `tripwire` propose en outre un mécanisme de signature du fichier de configuration afin qu'un attaquant ne puisse pas le changer pour le faire pointer vers une version différente de la base de données.

ment intuitive pour autant. La lecture de la page de manuel `aide.conf(5)` est donc bénéfique.

Une nouvelle version de la base de données est générée chaque jour dans `/var/lib/aide/aide.db.new` et peut être utilisée pour remplacer la base officielle si tous les changements constatés étaient légitimes.

B.A.-BA **Dénis de service**

Une attaque de type « déni de service » a pour seul objectif de rendre un service réseau inexploitable. Que cela soit en surchargeant le serveur de requêtes ou en exploitant un bogue de celui-ci, le résultat est toujours le même : le service en question n'est plus fonctionnel, les utilisateurs habituels sont mécontents et l'hébergeur du service réseau visé s'est fait une mauvaise publicité (en plus d'avoir éventuellement perdu des ventes, s'il s'agit par exemple d'un site de commerce en ligne).

DÉCOUVERTE **Les paquets checksecurity et chkrootkit/rkhunter**

Le premier paquet contient plusieurs petits scripts qui effectuent des vérifications de base sur le système (mot de passe vide, détection de nouveaux fichiers setuid, etc.) et alertent l'administrateur si nécessaire. Malgré son nom explicite, il ne faut pas se fier seulement à ce paquet pour vérifier la sécurité d'un système Linux.

Les paquets `chkrootkit` et `rkhunter` permettent de rechercher des potentiels *rootkits* installés sur le système. Rappelons qu'il s'agit de logiciels destinés à dissimuler la compromission d'un système et permettant de conserver un contrôle discret sur la machine. Les tests ne sont pas fiables à 100% mais ils permettent tout de même d'attirer l'attention de l'administrateur sur des problèmes potentiels.

Détection d'intrusion (IDS/NIDS)

`snort` (du paquet Debian éponyme) est un outil de détection d'intrusions (NIDS — *Network Intrusion Detection System*) : il écoute en permanence le réseau pour repérer les tentatives d'infiltration et/ou les actes malveillants (notamment les dénis de service). Tous ces événements sont enregistrés puis signalés quotidiennement à l'administrateur par un message électronique résumant les dernières 24 heures.

Son installation demande plusieurs informations. Il faut ainsi y préciser la plage d'adresses couverte par le réseau local : il s'agit en réalité d'indiquer toutes les cibles potentielles d'attaques. On précisera également l'interface réseau à surveiller. Il s'agit en général d'`eth0` pour une connexion Ethernet, mais on pourra aussi trouver `ppp0` pour une connexion ADSL ou RTC (Réseau Téléphonique Commuté, ou modem classique) voire `wlan0` pour certaines cartes Wi-Fi.

Le fichier de configuration de **`snort`** (`/etc/snort/snort.conf`) est très long et ses abondants commentaires y détaillent le rôle de chaque directive. Il est fortement recommandé de le parcourir et de l'adapter à la situation locale pour en tirer le meilleur parti. En effet, il est possible d'y indiquer les machines hébergeant chaque service pour limiter le nombre d'incidents rapportés par **`snort`** (un déni de service sur une machine bureautique n'est pas aussi dramatique que sur un serveur DNS). On peut encore y renseigner les correspondances entre adresses IP et MAC (il s'agit d'un numéro unique identifiant chaque carte réseau) pour détecter les attaques par *ARP-spoofing* (travestissement d'ARP), qui permettent à une machine compromise de se substituer à une autre (un serveur sensible par exemple).

ALTERNATIVE **Le NIDS hybride : prelude**

Prelude est une solution hybride parce qu'il ne se contente pas de faire NIDS, il remplit également certaines fonctionnalités de supervision centralisée. Ceci est possible grâce à son architecture modulaire : un serveur (le *manager* du paquet *prelude-manager*) centralise les alertes détectées par des capteurs (*sensors*) de plusieurs types. Le paquet *prelude-nids* propose un capteur qui, comme **`snort`**, surveille le trafic réseau. Le paquet *prelude-lml* (*Log Monitor Lackey*, ou laquais de surveillance de journaux système) surveille quant à lui les fichiers de *logs*, à l'instar de **`logcheck`** (voir page 328), déjà étudié.

ATTENTION **Rayon d'action**

`snort` est limité par le trafic qu'il voit transiter sur son interface réseau : il ne pourra évidemment rien détecter s'il n'observe rien. Branché sur un commutateur (*switch*), il ne surveillera que les attaques ciblant la machine l'hébergeant, ce qui n'a qu'un intérêt assez limité. Pensez donc à relier la machine employant **`snort`** au port « miroir », qui permet habituellement de chaîner les commutateurs et sur lequel tout le trafic est dupliqué.

Pour un petit réseau doté d'un concentrateur (*hub*), le problème ne se pose pas : toutes les machines reçoivent tout le trafic.