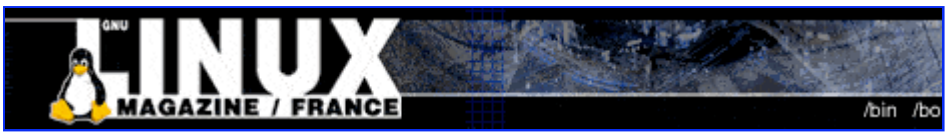

28 août 2008

Installer un serveur Syslog

Catégorie : Sécurité Tags : lmhs

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Lorsque l'on possède une ou plusieurs machines ouvertes sur l'extérieur, il est de bon ton de garder en permanence un œil sur ce qui s'y passe. Cependant, en cas de problème, les journaux pourraient avoir disparu de manière accidentelle ou délibérée.
Pour corriger cela, il suffit de prévoir une machine dédiée.

Les systèmes UNIX sont très bavards pour peu qu'on les y pousse. Ainsi, toutes les informations sur l'activité d'un système peuvent être connues, analysées, affichées et stockées. De cette manière, en cas d'attaque de la part d'un utilisateur malveillant ou d'un incident technique, on peut efficacement trouver la cause et la source du problème.
Malheureusement, l'administrateur malchanceux ayant déjà fait l'objet d'une attaque vous le dira, la première chose que tente de faire un pirate lorsqu'il prend le contrôle d'un système est d'effacer ses traces.
Parmi les éléments logiciels visés à ce moment, nous avons le système de gestion des journaux. Comprendre et exploiter pleinement ce système vous permettra, à défaut de vous prémunir, de conserver une trace sûre de toute activité licite ou non sur vos machines.
En « exportant » les journaux sur une machine dédiée et sécurisée, vous rendrez la tâche bien délicate pour un pirate souhaitant rester discret. De la même manière, en cas de gros problème sur l'une ou l'autre machine serveur, vous conserverez un maximum d'informations sur l'incident, même si celui-ci n'est pas le fait d'une attaque. Recycler un vieux 486 ou Pentium en serveur dans cette optique est donc plus que recommandé.

Le système classique

Nous ne nous attarderons pas ici sur le système traditionnellement utilisé sur les systèmes UNIX

et BSD. Cette solution reposant sur les démons syslogd et klogd, bien que fonctionnelle, sera avantageusement remplacée par la nouvelle génération de gestion des journaux d'activité.
Il est cependant important de garder en tête la manière habituelle dont cela fonctionne étant donné que le nouveau système repose énormément sur les principes déjà en œuvre.
Le premier démon, syslogd, est chargé de la gestion des requêtes qui lui sont envoyées par toutes les applications du système. Ce démon est une version évoluée de l'utilitaire Berkeley du même nom. Le second, klogd est pour sa part chargé de l'aspect « noyau ».
Il gérera tous les messages en provenance du programme le plus important du système : le noyau Linux.
syslogd utilise deux types d'informations pour qualifier un évènement à prendre en charge. Dans un premier temps, celui-ci est désigné par sa priorité ou, en d'autres termes, son importance (ici listé par ordre croissant) :

- ~~none~~ : aucune information sur le niveau d'importance ;
- ~~debug~~ : simples informations de débogage ;
- ~~info~~ : messages d'informations simples ;
- ~~notice~~ : messages significatifs mais informationnels ;
- ~~warning~~ : messages d'avertissement ;
- ~~error~~ : messages d'erreur ;
- ~~crit~~ : une situation critique se présente ;
- ~~alert~~ : messages d'alerte, une intervention humaine est demandée ;
- ~~emerg~~ : le système devient instable/inutilisable.

En plus de cela, syslogd organise les messages en fonction de leur type. On parle alors de « facilités » (facilities). Voici les facilités disponibles :

- ~~authpriv~~ (anciennement auth) pour tout ce qui traite de la sécurité et de l'authentification des services et des utilisateurs ;
- ~~cron~~ pour ce qui est en rapport avec l'ordonnancement des tâches (~~cron~~ et ~~at~~) ;
- ~~daemon~~ pour les messages provenant des démons en activité ;
- ~~kern~~ pour les messages du kernel (transitant par klogd) ;
- ~~local0~~ à ~~local7~~ pour les informations envoyées par des programmes qui permettent de choisir une facilité spécifique ;
- ~~lpr~~ pour ce qui est relatif au système d'impression ;
- ~~mail~~ pour ce qui concerne la messagerie électronique ;
- ~~news~~ pour les messages en rapport avec les services Usenet ;
- ~~syslog~~ pour les informations relatives au fonctionnement de syslogd lui-même ;
- ~~user~~ pour les informations génériques envoyées par les programmes « utilisateurs » ;
- ~~uucp~~ pour ce qui concerne le système UUCP ;
- ~~ftp~~ pour les messages concernant les services FTP ;
- ~~mark~~ est une facilité interne permettant d'ajouter des horodatages.

En utilisant à la fois le niveau d'importance des messages et la facilité concernée, on peut dispatcher les informations dans des journaux d'activité ou des consoles (~~/dev/tty*~~).
La configuration de syslogd se fait via le fichier ~~/etc/syslog.conf~~.
La syntaxe est relativement lourde et rigide (surtout en comparaison de ce qui va suivre dans l'article) :

```
auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none   -/var/log/syslog
```

```
mail.err                |/dev/xconsole
*.emergr                @hote.domain.fr
```

Les définitions s'entendent ligne par ligne. Nous avons sur la gauche les critères de sélection sous la forme facilité.importance et sur la droite la destination du message.
Pour cette dernière, nous avons le choix entre un fichier directement désigné, un fichier qu'il ne sera pas nécessaire de synchroniser après écriture (signe -), un tube nommé (|) ou encore une redirection vers le syslogd d'un autre système (@).
Il est également possible de jongler avec les critères de sélection en utilisant des négations (!) ou encore en précisant un niveau d'importance unique (=).
Il n'en reste pas moins que tout ceci est d'une rigidité affligeante. Heureusement, quelqu'un a eu la bonne idée de revoir ce système qu'on n'hésitera pas maintenant à qualifier de vieillissant.

La nouvelle génération

Le nouveau système généralement disponible (parfois par défaut) dans la quasi-totalité des distributions se nomme syslog-ng, « ng » signifiant « new generation ». Vous allez le constater, il s'agit d'un changement radical au niveau syntaxique, tout en conservant divers éléments du traditionnel syslogd.
Commençons donc par préciser que les notions de facilités et de niveaux d'importance sont toujours présentes et parfaitement compatibles.
En revanche, les auteurs de syslog-ng ont compris que cela n'était pas suffisant pour répartir convenablement les messages dans les divers fichiers journaux. Ainsi, la notion de filtre a été intégrée dans syslog-ng.
On peut filtrer en fonction de la facilité et de l'importance mais également en fonction, par exemple, du programme générant le message, de l'hôte d'où il provient et, comble du bonheur, d'expressions régulières définies par l'utilisateur. L'architecture générale de la configuration de syslog-ng tient en quatre éléments.

Les sources de messages

Les messages dans un système UNIX, comme GNU/Linux, peuvent provenir de plusieurs origines. Nous avons tout d'abord le système de gestion des journaux lui-même. Cette provenance est définie sous la désignation internal.
Nous avons ensuite les messages système envoyés par les applications qui utilisent la fonction syslog(). Ces messages sont récupérés par syslog-ng via /dev/log. Enfin, nous avons les messages en provenance du noyau Linux qui seront récupérés dans /proc/kmsg.
Voici un exemple de définition d'une source dans la configuration de syslog-ng (/etc/syslog-ng/syslog-ng.conf) :

```
source s_all {
        internal();
        unix-stream("/dev/log");
        file("/proc/kmsg" log_prefix("kernel: "));
};
```

Nous définissons ici une source appelée s_all qui regroupe des informations en provenance des trois points cités plus haut. Comprenez bien qu'une source est définie dans la configuration.
Nous aurions, par exemple, tout aussi bien pu définir une source s_interne pour internal(); et une

autre s_reste pour le reste (application et kernel). Il s'agit ici de fabriquer les briques qui nous permettront d'organiser notre système de gestion des journaux.
La syntaxe du fichier de configuration, bien que très différente du traditionnel syslog.conf sera rapidement assimilée. La portée d'une définition est spécifiée par des accolades. Chaque ligne ou portée se termine par un point-virgule. Enfin, chaque directive peut prendre en argument un ou plusieurs paramètres entre parenthèses.

Les destinations

Nous savons d'où tirer les informations sur le fonctionnement du système. A présent, nous devons décider où les envoyer. Dans la plupart des cas, il s'agira de fichiers placés dans /var/log mais, tout comme avec syslogd, on pourra choisir de les envoyer dans un tube ou sur un périphérique.

```
destination df_mail {
        file(„/var/log/mail.log“);
};
```

Nous définissons ici une première destination possible. A ce stade, nous n'avons pas encore décidé quels messages transiteront vers ces destinations (même si leur nom est plutôt explicite).
Cette première sentence définit un fichier comme destination. La directive (ou driver) file prend ainsi en paramètre le nom du fichier à utiliser.

```
destination dp_tty7 {
        pipe(„/dev/tty7“);
};
```

La seconde destination est un tube. Tout naturellement, la directive utilisée sera pipe et prendra en argument, au minimum, la cible de votre choix. Ici, nous envoyons les messages sur la septième console, mais nous aurions également pu utiliser /dev/xconsole.
Notez que si vous souhaitez renvoyer les messages sur un terminal afin que l'utilisateur responsable en prenne connaissance, on préfèrera la directive usertty avec, en argument, le nom d'utilisateur. A la charge du système de retrouver le ou les terminaux que cet utilisateur a sous les yeux.

```
destination df_facility_dot_err {
        file(„/var/log/$FACILITY.err“);
};
```

La troisième destination définie ici présente une caractéristique intéressante de syslog-ng. Il s'agit du fait de pouvoir utiliser des variables pour nommer les fichiers de destination.
Ici, nous utilisons directement le nom de la facilité pour nommer le fichier dans /var/log. Inutile donc de définir une destination par facilité, cette simple configuration suffit à régler le problème (je vous avais bien dit que le nouveau système était bien plus souple, non ?).
Ces exemples sont relativement simples et courants. Ils sont tirés du fichier de configuration par défaut tel qu'installé sur une Debian Sarge. Il est possible d'utiliser d'autres paramètres pour ces directives afin, par exemple, de définir un propriétaire aux fichiers journaux. Tous ces paramètres sont détaillés dans la documentation de syslog-ng. N'hésitez pas à la consulter pour affiner vos réglages.

Les filtres

Nous avons désigné les sources et les destinations mais pour pouvoir trier et organiser les flux de messages, il nous reste des briques élémentaires importantes à construire. Tout comme avec les deux précédents éléments, on définira tout un jeu de filtres réutilisables. Voici quelques exemples :

```
filter f_mail {
        facility(mail);
};
```

Nous définissons un ~~filter~~ nommé ~~f_mail~~ destiné à ne laisser passer que les messages concernant la facilité ~~mail~~. La directive ~~facility~~ prend, tout simplement, en argument le nom d'une facilité.

```
filter f_at_least_warn {
        level(warn..emerg);
};
```

Voici un filtre légèrement plus évolué. Il s'agit de conserver les messages ayant un niveau d'importance au minimum égal à ~~warm~~.
Contrairement à syslogd, le comportement par défaut de ce genre de filtre est de sélectionner uniquement les messages du niveau spécifié. Si l'on souhaite traiter « un niveau et plus », on utilisera la syntaxe présentée ici utilisant les deux points (..).

```
filter f_messages {
        level(info,notice,warn)
          and not facility(auth,authpriv);
};
```

Il est possible de combiner plusieurs conditions pour le filtrage des messages.
Ainsi, en utilisant des opérateurs logiques, on peut affiner une sélection à l'extrême. Le filtre présenté ici sélectionne les messages de niveau ~~info~~, ~~notice~~ et ~~warn~~ qui ne concernent pas les facilités ~~auth~~ (obsolète mais encore utilisé dans certains programmes) et ~~authpriv~~.

```
filter f_xconsole {
    facility(daemon,mail)
        or level(debug,info,notice,warn)
        or (facility(news)
           and level(crit,err,notice));
};
```

Ce filtre ~~f_xconsole~~ présente la possibilité de pousser très loin les imbrications de conditions grâce aux opérateurs logiques. Nous sommes maintenant très très loin de syslogd.
Il existe d'autres directives permettant de filtrer en fonction d'autres éléments.
Citons par exemple ~~match~~ permettant d'utiliser des expressions régulières ou encore ~~netmask~~ se basant sur l'adresse source du message et vérifiant la correspondance avec un sous-réseau donné.
Encore une fois, c'est dans la documentation officielle de syslog-ng que vous découvrirez tout cela.

Les journaux

Avec nos trois briques de base, les sources, les destinations et les filtres, nous pouvons gérer efficacement les messages du système :

```
log {
        source(s_all);
        filter(f_mail);
        destination(df_mail);
```

```
}
```

log nous permet de connecter des sources avec des destinations via les filtres. Oui, je parle bien au pluriel puisqu'il est parfaitement possible dans la portée d'un log de spécifier plusieurs sources, plusieurs filtres et plusieurs destinations.
On utilisera autant de portées log que nous souhaitons faire de connexion. L'ordre des portées a une importance puisque les messages seront traités dans leur ordre d'apparition.
On pourra utiliser la directive flags afin de spécifier la manière dont le traitement se déroulera. Ainsi, si l'on souhaite qu'une portée interrompe le traitement, on ajoutera flags(final); dans cette dernière.
Si un message correspond, toutes les occurrences suivantes de log seront ignorées.

Exporter les journaux

Comme dit plus haut, disposer d'un système de gestion des journaux efficace et performant est une chose importante. Cependant, en cas d'incident ce système peut, lui-même, avoir subi des dommages.
Le fait d'effacer ses traces est d'ailleurs la première chose que fera un intrus lorsqu'il prendra le contrôle de tout ou partie de votre système. La solution pour résoudre ce problème est l'objet même de cet article. Il faut dédier une machine sûre qui accumulera les messages système d'un ou plusieurs hôtes à surveiller.
Ne confondez pas pour autant cela avec de la supervision telle que le proposerait d'autres solutions comme Nagios. Exporter les journaux vers une machine distante reste une solution d'analyse post-incident et non de la prévention.
Syslog-ng permet très simplement, de par son architecture modulaire, de travailler en réseau. Ainsi, pouvoir envoyer les messages à un hôte distant revient simplement à définir une nouvelle destination et à l'utiliser :

```
destination dudp_molly {
  udp(„192.168.0.68“ port(514)):
};
```

udp est une directive permettant d'envoyer les messages en UDP à l'hôte spécifié en paramètre. Il est également possible d'utiliser un protocole avec connexion, auquel cas on utilisera la directive tcp.
Côté hôte recevant l'information, il suffira de définir une nouvelle source ou de l'inclure dans celle déjà définie :

```
source s_morgane {
  udp (ip(192.168.0.68) port(514));
};
```

La directive est la même mais les arguments diffèrent. Aucun argument n'est nécessaire, mais on précisera, comme ici, l'interface locale devant être « écoutée » ainsi que le port par mesure de sûreté.
Selon cette architecture, deux solutions s'offrent à vous. Soit vous filtrez puis envoyez à la machine dédiée, soit vous envoyez les données brutes en masse et filtrez ensuite. C'est en particulier la bande passante à votre disposition qui influera sur la décision. Lorsqu'on travaille avec un réseau local, on préfèrera, tout naturellement envoyer les données en masse.
Avec une connexion à faible débit, moins d'informations transiteront via le réseau et mieux se

porteront vos machines.

Enregistrer dans MySQL

L'archivage et le stockage des journaux système, même parfaite-ment triés et organisés, posent un problème de gestion d'espace disque. Il ne s'agit pas là du seul problème que l'on rencontre avec des lignes de messages enregistrées dans de simples fichiers texte.
Tout l'intérêt d'utiliser un hôte distant et dédié pour archiver les journaux est de pouvoir garder une trace le plus longtemps possible dans le but d'analyser une attaque, un problème ou un incident.
En poussant plus loin dans ce sens, on arrive aux limites de l'archivage par fichier. Bien que des outils comme grep, sed, awk ou encore perl soit très puissants, une utilisation en production n'est pas la meilleure solution. Lorsqu'on souhaite interroger une masse de données dans le but de faire des recherches, l'outil le plus adéquat reste un système de gestion de bases de données.
L'interrogation et la recherche deviennent des requêtes et on repose alors entièrement sur les performances du SGBD dont c'est la spécialité.
Pour cet exemple, c'est tout naturellement vers MySQL qu'on se dirigera mais d'autres SGBD pourraient parfaitement faire l'affaire. Je pense, par exemple, à PostgreSQL dont les performances, avec de larges bases de données, n'ont pas grand-chose à envier aux leaders commerciaux et propriétaires du marché.
L'utilisation de MySQL conjointement avec un système de gestion des journaux est d'une simplicité déconcertante dans le cas de syslog-ng. Inutile de rechercher un éventuel patch ou support spécifique, les fonctionnalités propres à syslog-ng sont largement suffisantes.
Nous avons vu précédemment dans l'article qu'il était possible de créer une destination pouvant être un tube, un fichier ou un hôte distant. Afin d'enregistrer les messages dans une base MySQL, nous allons utiliser comme destination un programme :

```
destination d_mysql {
  program(„mysql -uroot -pXXXXXXXX syslog“
```

Nous créons la destination ~~d_mysql~~ utilisant la directive ~~program~~ avec en argument, le programme et les options qui lui sont nécessaires.
La commande ~~mysql~~ recevra sur son entrée standard (~~stdin~~) le message qui aura été dirigé vers la destination d_mysql. On remarquera que la seconde ligne n'est pas terminée par ~~);~~, car nous allons ajouter d'autres arguments à la directive ~~program~~. Nous ne pouvons envoyer directe-ment le message à ~~mysql~~. Nous devons le formater de manière à créer une requête SQL d'insertion. Pour cela, nous allons utiliser un paramètre spécifique des directives ~~pipe~~ et ~~file~~ : ~~template.template~~ permet de reformater un message en utilisant les différents éléments qui le composent sous la forme de variables. Voici notre ~~template~~ :

```
template(„INSERT INTO logs
  (host, facility, priority, level,
  tag, date,time, program, msg)
  VALUES
  (‘$HOST’,’$FACILITY’,’$PRIORITY’,
  ‘$LEVEL’,’$TAG’,’$YEAR-$MONTH-$DAY’,
  ‘$HOUR:$MIN:$SEC’,’$PROGRAM’,’$MSG’);\n“)
```

Nous composons littéralement une requête SQL que nous passerons ensuite à ~~mysql~~. Vous pourrez trouver la liste des variables dans la documentation de syslog-ng. Notez cependant que nous les

utilisons presque toutes ici. Il ne nous reste qu'à ajouter un dernier paramètre pour la directive ~~program~~ :

```
template-escape(yes));
};
```

Cette option permet d'activer l'échappement des simples et doubles quotes pour les variables dans la sortie. Ceci est particulièrement intéressant pour les requêtes SQL et permet d'éviter d'éventuelles injections SQL de la part d'un attaquant (un peu comme les Magic Quotes de PHP).
Notre destination est maintenant prête, il ne nous reste plus qu'à créer la base et la table MySQL :

```
CREATE DATABASE syslog;
USE syslog;
CREATE TABLE logs (
  host varchar(32) default NULL,
  facility varchar(10) default NULL,
  priority varchar(10) default NULL,
  level varchar(10) default NULL,
  tag varchar(10) default NULL,
  date date default NULL,
  time time default NULL,
  program varchar(15) default NULL,
  msg text,
  seq int(10) unsigned NOT NULL auto_increment,
  PRIMARY KEY (seq),
) TYPE=MyISAM;
```

On ajoute ensuite un nouveau filtre dans la configuration syslog-ng pour les premiers essais :

```
filter f_auth2 {
  facility(auth, authpriv)
  and
  program(„su|sshd“);
};
```

Enfin, nous assemblons le tout :

```
log {
  source(s_all);
  filter(f_auth2);
  destination(d_mysql);
};
```

Dès lors, tous les messages concer-nant les facilités ~~auth~~ et ~~authpriv~~ en rapport avec les programmes su et sshd seront formatés et envoyés à la commande ~~mysql~~. Notez qu'il n'y aura pas un enregistrement par évènement mais un enregistrement par ligne de message. Une tentative infructueuse d'utilisation de su provoque, par exemple, trois entrées dans la base. Si vous souhaitez affiner ce type de détail, il vous suffira d'ajouter un filtre permettant de sélectionner avec plus de précision les messages qui vous intéressent (grâce à la directive ~~match~~).
Il ne s'agit là, bien sûr, que d'une base de travail. L'utilisation des variables dans la définition de la destination couplée aux capacités de filtrage vous permettent de stocker la totalité des informations utiles dans des tables MySQL. Imaginez ainsi un reformatage débutant ainsi :

```
template(„INSERT INTO $FACILITY (host...
```

Une fois toutes les informations stockées ainsi dans la base, les requêtes d'interrogation ou encore l'écriture de scripts PHP analysant et « graphant » les informations sont un jeu d'enfant. Dans le

même ordre d'idée et toujours en relation avec la notion de base de données, vous pouvez imaginer interfacer de la même manière syslog-ng et RRDtools.

Conclusion

Nous venons de voir que le système de gestion des journaux a grandement évolué avec l'apparition de syslog-ng. Cet article ne présente cependant qu'une partie de ce qu'il est possible de faire.
La syntaxe du fichier de configuration et les fonctionnalités de syslog-ng vous permettent de faire strictement ce qu'il vous plaît. Pourquoi donc s'en priver...

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Posté par Denis Bodor (Lefinnois) | Signature : Denis Bodor | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- Des plugins pour vos blogs !
- Anonymisation
- Canaux cachés (ou furtifs)
- FON : le grand partage du Wifi !
- MakeHuman : interview de l'équipe de développement
- La stéganographie moderne
- La protection du secret : approche juridique
- Mieux connaître OOo Draw : les objets 3D et leur espace
- Calc et les variables
- Noël 94 : le cas Mitnick-Shimomura ou comment le cyber-criminel a souhaité joyeux Noël au samurai

Actuellement en kiosque : 

- # Il y a actuellement

- 749 articles/billets en ligne.

Recherche

- # Catégories

- - Administration réseau
  - Administration système
  - Agenda-Interview
  - Audio-vidéo
  - Bureautique
  - Comprendre
  - Distribution
  - Embarqué
  - Environnement de bureau
  - Graphisme
  - Jeux
  - Matériel
  - News
  - Programmation
  - Réfléchir
  - Sécurité
  - Utilitaires
  - Web

- # Archives

- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 108
- GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 38
- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°49
- À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49
- Linux Pratique Essentiel N°3 - Août/Septembre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°3

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse
- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux