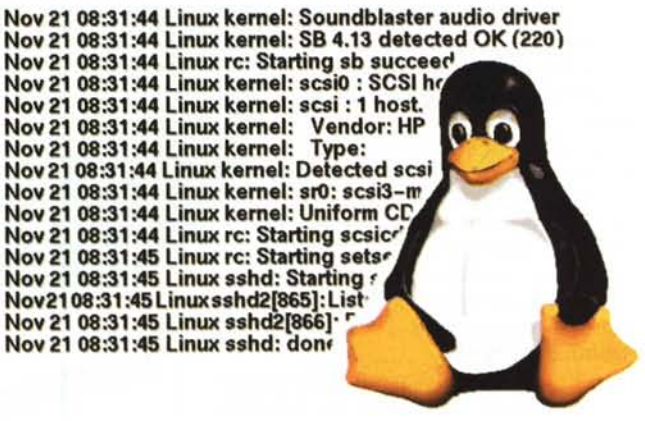
Come funziona Linux: Tenere traccia dell'attività del sistema

Decima parte

di Giuseppe Zanetti



"Capita spesso che qualcosa non funzioni oppure funzioni male e che si debba andare a cercarne le cause. Per fortuna, da sistema operativo che si rispetti, anche Linux tiene traccia, nei "file di log", di tutti gli eventi importanti che accadono...".

Un aspetto che abbiamo tralasciato nelle scorse puntate è quello dell'analisi dei file di log (registrazione, diario). Si tratta di quei file in cui viene tenuta traccia dell'attività del kernel e di altri programmi, come i server che gestiscono i servizi per Internet. Viene inoltre tenuta la lista dei collegamenti degli utenti, sia da console/terminale che via rete (telnet, ftp, ...).

Spesso si fa ricorso ai file di log solamente come strumenti di "debugging" per risolvere eventuali problemi della macchina. In realtà questo non è il solo scopo per cui essi sono utili. Essi servono anche ad esempio per avere un "accounting" sull'utilizzo delle risorse (ad esempio per vedere chi è venuto a visitare il nostro sito WWW oppure per dividere i costi del traffico Internet fra vari clienti) e per motivi di sicurezza.

L'analisi frequente e regolare di questi file potrebbe infatti sembrare a prima vista una operazione inutile, specialmente fintanto che il computer sembra funzionare correttamente. In realtà, al pari del backup, si tratta di una buona abitudine che, una volta acquisita, aiuta a proteggersi da guai maggiori. Ad esempio analizzando il file /var/log/secure è possibile accorgersi se mentre siamo collegati ad Internet qualcuno sta collegandosi alla nostra macchina per vedere i nostri file o, peggio, per usarla come deposito di file illeciti o come "ponte" per eseguire altri collegamenti non troppo leciti (magari verso le altre macchine della nostra rete). Di default infatti alcune distribuzioni di Linux lasciano attivi molti servizi di rete, alcuni dei quali se non opportunamente configurati possono essere dei veri e propri buchi di sicurezza.

I log non servono certamente a bloccare un pirata, ma, se usati correttamente, almeno aiutano a capire che sta accadendo o è accaduto qualcosa e che si deve correre ai ripari.

La prima volta che si analizza un file di log si perderà certamente parecchio tempo, in quanto ci saranno diversi giorni passati da verificare. Tuttavia le volte successive il tutto sarà molto più veloce, sia perchè ormai si sarà presa dimestichezza col formato generato dai vari programmi, sia perchè i messaggi da analizzare saranno in quantità minore. E' bene compiere tale operazione di frequente, almeno per i file riguardanti la sicurezza, in quanto solo in questo modo è possibile bloccare un eventuale attacco prima che l'autore riesca a compiere danni importanti.

Alcuni programmi vengono spesso addirittura utilizzati al solo scopo di generare dei file di log: l'esempio più classico è il proxy Squid, il quale può essere configurato per intercettare e tenere traccia di tutto il traffico generato dai PC di una rete aziendale durante la "navigazione" in Internet (in modo assolutamente trasparente per l'utente... perciò in ufficio è molto più sicuro leggere la classica "rosea" su carta invece che la versione on-line).

In Linux il "logging" e la segnalazione degli errori può essere gestita sia direttamente dal programma (come avviene per Squid), oppure mediante l'apposita funzione standard syslog, presente in una delle librerie standard del linguaggio C. Essa viene chiamata passandole come parametri il messaggio di cui si vuole tenere traccia, e opportuni valori detti "facility" e "priority".

La facility è una sorta di organizzazione per categorie dei messaggi di log, che specifica il sottosistema che ha generato il messaggio. Ad esempio tutti i programmi che gestiscono la posta e che utilizzano syslog per il logging dovrebbero usare la facility "mail".

Le facility, definite da Linux in /usr/include/syslog.h, sono le seguenti:

auth	auth-priv
cron	daemon
kern	lpr
mail	news
syslog	user
uucp	local0-local7

Le priorità invece definisce la "severità" del messaggio. Quelle possibili in Linux sono (in ordine crescente):

debug	info
notice	warning (warn)
error (err)	crit
alert	emerg (panic)

I messaggi generati non vengono scritti direttamente sui file di log, bensì passati ad un apposito demone, syslogd, che si occupa di gestirli. In base ai valori di facility e priority è possibile scegliere, usando opportune regole configurabili dall'amministratore di sistema, dove dirigere i diversi messaggi: ad esempio quelli riguardanti la gestione della posta elettronica vengono scritti nel file /var/log/messages, mentre quelli critici vengono sia salvati su file che mostrati direttamente sulla console di sistema.

Una tipica linea di file di log generata usando syslogd ha la seguente struttura:

```
data macchina servizio[pid del processo]: messaggio
```

ad esempio il seguente messaggio in /var/log/access indica una connessione riuscita alla nostra macchina mediante ftp:

```
Mar  6 16:22:12 freddy in.ftpd[3097]: connect from 1.2.3.4
```

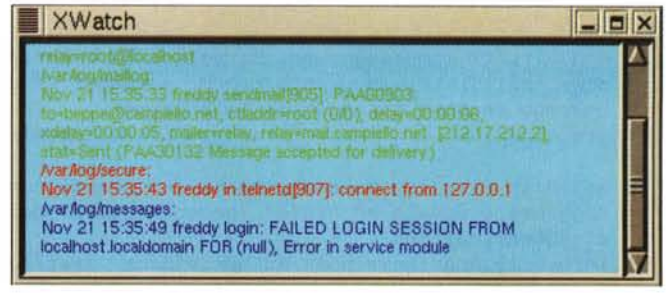
Il tipo ed la quantità di messaggi di log generati variano da programma a programma e possono in molti casi essere variati utilizzando degli appositi selettori nella linea di comando con cui si esegue il programma. Nel caso dei demoni che gestiscono servizi Internet, di solito questo aspetto viene tenuto in seria considerazione, in quanto essi sono accessibili anche da persone all'esterno della macchina.

Il programma telnetd, che gestisce i collegamenti remoti usando telnet, tiene traccia, oltre che dei tentativi riusciti di accesso, anche di situazioni atipiche come tentativi ripetuti di connessione da uno stesso indirizzo oppure tentativi di utilizzo dell'account "root".

Ftpd, il server che gestisce lo scambio di file via FTP, manda invece a syslogd informazioni sull'utente connesso e contemporaneamente tiene per proprio conto un log dei file trasferiti in /var/log/xferlog. Per informazioni su come viene gestito il logging da altri servizi o programmi, si consulti la relativa documentazione in linea (es: man ftpd).

In quasi tutte le distribuzioni di Linux, la locazione standard per i file di log è la directory /var/log. I file di log più usati sono i seguenti:

/var/log/messages	messaggi generici generati da diversi programmi, kernel, ...
/var/log/secure	messaggi relativi all'autenticazione (connessioni effettuate e fallite)
/var/log/maillog	messaggi relativi al sistema di gestione della e-mail (SMTP, POP3)
/var/log/boot.log	messaggi generati alla partenza dal kernel gli script in /etc/rc.d/
/var/log/debug	messaggi di debugging (es: PPP)
/var/log/httpd/access_log	log di accesso al web server
/var/log/httpd/error_log	errori generati dal web server
/var/log/squid/access_log	utilizzo del proxy da parte degli utenti
/var/log/lastlog	ultimo accesso da parte degli utenti al sistema (formato proprietario del programma lastlog)
/var/log/wtmp	utenti attualmente collegati al sistema (formato proprietario usato da who, w, ...)
/var/log/utmp	storia dei collegamenti al sistema (formato proprietario usato da last, ...)
/var/log/dmesg	messaggi del kernel
/var/log/xferlog	log del server ftpd



Xwatch permette di tenere sotto controllo contemporaneamente più file di log nella stessa finestra evidenziando ognuno mediante diversi colori.

Si presti attenzione al fatto che sotto /var/log sono presenti anche file generati da altri programmi senza ricorrere al meccanismo di syslog (ad esempio /var/log/wtmp).

Un discorso a parte meritano i messaggi generati dal kernel. Non potendo essere spediti direttamente a syslog, essi di solito vengono gestiti dal kernel stesso, che per default li manda nella console e li inserisce in un buffer circolare di 16384 byte. In dati ivi inseriti possono essere ottenuti attraverso il file speciale /proc/kmsg. Mediante l'opportuno programma klogd, è possibile inserire tali messaggi nel meccanismo di syslog alla pari di quelli generati dai processi utente. Il log dei messaggi generati dal kernel è anche disponibile direttamente nel file /var/log/dmesg.

Di seguito è riportata la configurazione di /etc/syslogd.conf di un sistema Red Hat 6.2. Il formato utilizzato è

```
facility.priority                    destinazione
```

dove tutti questi valori possono essere sostituiti dal simbolo * ad indicare "tutti" (perciò mail.* indica i messaggi generati dal sottosistema di email, mentre *.emerg indica i messaggi critici generati da qualunque sottosistema). Come da prassi, il carattere # introduce un commento.

TRIPWIRE

Uno strumento da affiancare all'analisi dei file di log è Tripwire (http://www.tripwire.com). Esso permette di verificare l'integrità dei propri file rispetto alle firme degli stessi salvate (su dischetto o su altro supporto asportabile) al momento dell'installazione della macchina. Di ogni file vengono salvate non solo le informazioni relative a dimensioni, permessi, proprietario e data di modifica (tutti questi valori facilmente falsificabili), ma anche uno o più "checksum" ottenuti facendo uso di tecniche diverse (CRC32, MD5, ...). In questo modo è possibile in ogni momento verificare quali file sono eventualmente stati modificati rispetto alla configurazione originale. Ovviamente il controllo per essere efficace deve essere effettuato abbastanza spesso. Nel file di configurazione è possibile specificare in modo fine quali file controllare, limitando eventualmente la visione a solo alcuni parametri, ad esempio proprietario e permessi, cosa molto utile per file che variano spesso nel tempo.

L'unico "neo" di tripwire è che si tratta di un software non disponibile sotto licenza GPL. Si stanno tuttavia realizzando dei programmi analoghi disponibili liberamente.

```
# Manda tutti i messaggi del kernel sulla console (di serie
# non è abilitato, in quanto troppi messaggi in console
# possono rendere illeggibile lo schermo)
#kern.*                                 /dev/console

# salva in /var/log/messages tutti i messaggi con priorità
# maggiore o uguale ad "info" (eccetto quelli della posta e
# di autenticazione)
*.info;mail.none;news.none;authpriv.none  /var/log/messages

# salva tutti i messaggi relativi all'autenticazione
authpriv.*                              /var/log/secure

# salva i messaggi relativi alla posta elettronica
mail.*                                  /var/log/maillog

# i messaggio con priorità di tipo "emergenza" vengono
# salvati in tutti i file e mostrati sulla console
     *.emerg                                    *

# errori generati dai sistemi di gestione di mail e news
uucp,news.crit                          /var/log/spooler

# messaggi generati al boot
local7.*                                /var/log/boot.log

# vari tipi di messaggi generati dal server di news innd
news.=crit                              /var/log/news/news.crit
news.=err                               /var/log/news/news.err
news.notice                             /var/log/news/news.notice

# messaggi di debugging
*.=debug                                 /var/log/debug
```

Ci si ricordi per rendere effettive le modifiche di far ripartire il demone syslogd, ad esempio con " /etc/rc.d/rc3.d/S30syslog restart" se si usa Red Hat.

```
# /etc/rc.d/rc3.d/S30syslog restart
Shutting down kernel logger:       [  OK  ]
Shutting down system logger:       [  OK  ]
Starting system logger:            [  OK  ]
Starting kernel logger:            [  OK  ]
```

Log su una macchina remota

La prima cosa che un pirata informatico -- in gergo "cracker", da non confondere con l'hacker: si veda il riquadro "hacker o cracker ?" -- si premura di fare appena penetrato in un sistema altrui è quella di cancellare le proprie tracce dai file di log.

Per compiere tale operazione non occorre moltissimo tempo, in quanto esistono diversi software (rootkit) che lo fanno automaticamente e poi modificano alcuni programmi importanti di sistema (es: login, telnetd, tcpd) in modo che essi non tengano più il log delle connessioni dall'indirizzo del pirata. Inoltre un cracker abbastanza furbo, non solo cancella i log che lo riguardano, ma anche ne inserisce di falsi nel sistema, in modo da confondere le tracce e sviare la ricerca del colpevole.

Queste evenienze di fatto vanificherebbero l'utilità di avere i file di log.

Per limitare queste possibilità, è possibile, invece di tenere i log sulla macchina locale, mandarli ad un demone syslogd residente su un'altra macchina in rete (quest'ultimo deve essere stato lanciato con l'opzione -r). Si tenga presente che in determinate situazioni vengono generati parecchi messaggi di log, che possono generare parecchio traffico e rallentare la rete.

Per mandare tutti i messaggi alla macchina con indirizzo IP 192.168.100.2, sarà sufficiente inserire nel file /etc/syslogd.conf della macchina da sorvegliare una linea del tipo:

```
  # Log all kernel messages to a
remote host
  *.*                @192.168.100.2
```

In questo modo si nega ad un eventuale aggressore che sia riuscito ad acquisire i permessi di root la possibilità di cancellare i vecchi log, ma non quella di generare dei messaggi fasulli. Perdipiù, dato che la comunicazione in rete avviene in chiaro ed in forma non autenticata (protocollo UDP), non è neppure necessario essere sulla macchina oggetto dell'attacco per generare dei log finti.

Mandando in rete i messaggi di log ci si espone inoltre al pericolo che essi vengano intercettati e che le informazioni così reperite possano essere utilizzate per cercare buchi di sicurezza utili ad attaccare la macchina che ha generato i log.

Esistono diversi modi per aggirare tale problema: le nuove versioni di syslog consentono ad esempio di mandare i messaggi ad un programma esterno mediante una pipe. In questo modo è possibile creare un canale di comunicazione criptato utilizzando strumenti tipo ssh (Secure SHell).

In alternativa esistono delle estensioni di syslogd che fanno sì che esso firmi mediante algoritmi crittografici i messaggi generati, in modo da accorgersi se essi sono stati manomessi.

Purtroppo non vi sono ancora dei metodi standard implementati dal syslog fornito di serie con le maggiori distribuzioni di Linux. Esiste tuttavia una implementazione sicura che tiene conto di queste problematiche, che può essere reperita su http://www.core-sdi.com/english/slogging/ssyslog-dl.html.

Utilizzo di syslog nei propri programmi

Nel seguente esempio di codice sorgente (in C, ma syslog è supportato da quai tutti i linguaggi) viene innanzitutto aperta una connessione col sistema di logging utilizzando la funzione openlog, in cui devono essere specificati il nome con cui si vuole che appaiano i messaggi nel file di log e la facility da utilizzare.

La priorità del messaggio viene invece scelta di volta in volta al momento di inviarlo, in modo da non dover chiudere e riaprire ogni volta il collegamento col sistema di log.

```
#include <syslog.h>
...
openlog("pippo", LOG_PID, LOG_LOCAL6);
...
syslog(LOG_INFO,"messaggio poco importante\n");
...
syslog(LOG_CRIT,"condizione critica \n");
...
closelog();
```

Il messaggi generati sono i seguenti:

```
Nov 18 10:51:46 freddy pippo[650]: messaggio poco importante
Nov 18 10:51:46 freddy pippo[650]: condizione critica
```

Per dirigere oppurtunamente i messaggi, si può aggiungere alla configurazione di syslog le seguenti linee:

```
local6.*                /var/log/pippo.log
local6.crit             /dev/console
```

In questo modo i file con facility local6 e priorità info vengono salvati su un file, mentre quelli indicanti condizioni critiche vengono mostrati sulla console di sistema (file speciale /dev/console).

Esempio: log di apache mediante syslog

Il meccanismo di logging centralizzato offerto da syslog è estremamente comodo e funzionale, tuttavia abbiamo visto che non tutti i programmi lo utilizzano.

Per alcuni programmi è possibile ottenere tale funzionalità mandando i messaggi di logging nello standard output oppure nello standard error ed usando uno script esterno che li intercetti e li invii a syslogd. In alcune distribuzioni di Linux sono presenti i programmi "syslog" o "syslog_tst" che compiono operazioni simili a questa e che possono essere adattati a questo scopo. In mancanza di essi si può scriversi da soli un programmino in C o in un altro linguaggio come abbiamo visto in precedenza.

Nel caso di Apache il trucco consiste nell'aggiungere al file di configurazione /etc/httpd/conf/httpd.cont una linea simile alla seguente:

```
CustomLog    "|    /usr/local/apache/bin/logger 192.168.1.2" common
```

In questo modo si apre una pipe verso il programma logger, il quale è un semplice script in perl che genera messaggi syslog (ad esempio con facility "local0" e priority "info") a partire dallo standard input:

hacker o cracker?

La televisione, i quotidiani e, ahinoi, anche alcune riviste di informatica, si ostinano ad utilizzare in modo assolutamente improprio il termine "hacker" come sinonimo di pirata informatico (per il quale una definizione più corretta è "cracker"). In realtà tale termine non è nato con un significato dispregiativo: "hacker" non è colui che penetra con malizia nei sistemi altrui allo scopo di creare danni o rubare dati, bensì, rifacendosi alla definizione originaria di "to hack", chi conosce gli strumenti di cui dispone talmente bene da essere in grado di "trovare delle scorciatoie", ovvero delle soluzioni originali e impensabili per le persone comuni.

Per capire a fondo cosa sia veramente un hacker si può leggere "The Jargon File" (http://www.jargon.org/), una specie di grande vocabolario (un megabyte !) che raccoglie i termini gergali utilizzati dalla categoria, oltre che informazioni interessanti sull'etica e sulla cultura hacker. Ne esiste anche una versione stampata dal MIT Press dal titolo "The New Hacker's Dictionary" (ISBN 0-262-68092-0).

Citando una delle definizioni dello Jargon, l'hacker è "una persona che si diverte a capire il funzionamento profondo di un sistema programmabile, non limitandosi, come la maggior parte degli utenti, ad imparare solamente il minimo necessario".

Fra le idee portate avanti dagli hacker vi è quella della libera diffusione delle informazioni. Non a caso il coordinatore dello Jargon è Eric S. Raymond, l'autore di uno dei testi fondamentali del movimento Open Source, "La Cattedrale ed il Bazaar".

Prima di lasciarvi alla lettura della definizione di hacker dello Jargon, ricordate una cosa molto importante: entrare nell'elite degli hacker non è facile e sicuramente non è sufficiente autoproclamarsi tale... sono gli altri che devono farlo.

L'hacker di Linux per eccellenza ? Linus Torvalds, ovviamente.

Dallo Jargon (http://www.tuxedo.org/~esr/jargon/html/entry/hacker.html)

"[originally, someone who makes furniture with an axe] 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. 2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming. 3. A person capable of appreciating hack value . 4. A person who is good at programming quickly. 5. An expert at a particular program, or one who frequently does work using it or on it; as in `a Unix hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.) 6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example. 7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations. 8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence `password hacker', `network hacker'. The correct term for this sense is cracker .

The term `hacker' also tends to connote membership in the global community defined by the net (see the network and Internet address). For discussion of some of the basics of this culture, see the How To Become A Hacker FAQ. It also implies that the person described is seen to subscribe to some version of the hacker ethic (see hacker ethic).

It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you'll quickly be labeled bogus).

```
#!/usr/local/bin/perl

use Sys::Syslog;

$SERVER_NAME = shift || 'www';
$FACILITY = 'local0';
$PRIORITY = 'info';

Sys::Syslog::setlogsock('unix');
openlog ($SERVER_NAME,'ndelay',$FACILITY);
while (<>) {
      chomp;
      syslog($PRIORITY,$_);
}
closelog;
```

Per mandare i messaggi in un file appropriato, si aggiungano le seguenti linee a /etc/syslogd.conf:

```
local0.info                /var/log/web/access_log
```

Volendo mandare il log su un'altra macchina, si utilizzi invece una cosa del tipo:

```
local0.info                @logserver.pippo.com
```

Ovviamente nella macchina logserver.pippo.com, syslogd deve essere configurato con le linee precedenti, in modo da mandare l'output su un file.

I file wtmp e utmp

Un discorso a parte meritano i file /var/log/wtmp e /var/log/utmp: essi contengono il registro delle connessioni alla macchina e vengono generati automaticamente dal sistema di autenticazione di Linux.

Il loro formato è descritto nel file di "include" . Al contrario dei file di log appena visti, in questo caso si tratta di un formato binario.

Non è un problema, in quanto di solito essi non vengono gestiti direttamente dall'utente ma bensì mediante appositi programmi, come who e w, i quali mostrano la lista degli utenti attualmente collegati al sistema

```
$ who
root       tty1      Nov 21 08:33
beppe      tty2      Nov 21 08:32
webmaster pts/0     Nov 21 08:24
```

oppure last, che visualizza la storia di tutte le connessioni.

```
$ last|more
root       tty1   Tue Nov 21 08:33   still logged in
beppe      tty2   Tue Nov 21 08:32   still logged in
root       tty1   Tue Nov 21 08:23 - 08:32  (00:09)
webmaster pts/0  Tue Nov 21 08:24   still logged in
root       pts/5  Tue Nov 20 10:30 - 10:38  (00:08)
root       pts/5  Tue Nov 20 10:19 - 10:26  (00:06)
root       pts/5  Tue Nov 20 10:12 - 10:13  (00:01)
```

Il fatto che siano in formato binario, non significa che modificarli sia difficile. Un programmino pronto allo scopo è il seguente: ftp://sunsite.unc.edu/pub/Linux/system/admin/log/clnwtmp-1.5.tar.gz.

I file .bash_history

Si tratta di file particolari, presenti nella home directory di ogni utente che utilizza la shell bash, i quali contengono la storia dei comandi impartiti. Essi sono particolarmente utili per ricostruire eventuali operazioni compiute (per malizia o per errore). Ovviamente se l'utente non ha cancellato o falsificato le linee relative al fatto.

Tenere d'occhio i file che crescono

A lungo andare i file di log tendono a crescere fino a riempire il disco della macchina. La soluzione ovvia è quella di "accorciare" ed archiviare a mano i file, facendo ovviamente attenzione a non modificarne i permessi.

Il trucco consiste nell'evitare di cancellare i file con rm (anche perchè non tutti i programmi sono in grado di ricrearseli in caso di bisogno), ma di usare invece l'operatore di ridirezione per sovrascrivere il contenuto del file

```
>nome
```

Volendo fare le cose per bene sarebbe bene non modificare i file mentre syslogd o un altro programma li sta scrivendo. Per far ciò bisognerebbe interrompere e far ripartire i vari programmi con /etc/rc.d/rc3.d/S30syslogd restart.

Esiste un software che da usare in alternativa al metodo manuale che tiene conto di tutti questi aspetti: si tratta di logrotate. Esso viene eseguito quotidianamente mediante il meccanismo di crontab ed agisce in base alle configurazioni presenti nel file /etc/logrotate.conf.

Al pari di molti altri programmi attuali, è possibile aggiungere un "pezzo di configurazione" anche creando un file nella directory /etc/logrotate.d; in questo modo ogni pacchetto software installato può aggiungere in modo pulito le proprie regole di rotazione dei log.

Come esempio vediamo delle possibili regole, tratte dal manuale in linea di logrotate, che indicano come gestire la rotazione del file /var/log/messages e del log del server WWW Apache:

```
errors beppe@profuso.com
compress

/var/log/messages {
 rotate 5
 weekly
 postrotate
      /sbin/killall -HUP syslogd
 endscript
}

"/var/log/httpd/access.log" {
 rotate 5
 mail beppe@profuso.com
 errors beppe@profuso.com
 size=100k
```

```
 postrotate
     /sbin/killall -HUP httpd
 endscript
}

/var/log/news/* {
 monthly
 rotate 2
 missingok
 errors beppe@profuso.com
 postrotate
     kill -HUP `cat /var/run/inn.pid`
 endscript
 nocompress
}
```

L'inizio del file contiene impostazioni globali che fungono da valori di defaut per tutti i file da ruotare e che possono essere sovrascritte da valori presenti nelle definizioni relative ai singoli file. Le prime due righe contengono rispettivamente l'indirizzo di e-mail a cui segnalare eventuali errori e l'indicazione di eseguire la compressione sugli archivi di file di log creati.

Le sezioni successive, che potrebbero essere poste su file a parte sotto /etc/logrotate.d, si applicano invece solamente durante la rotazione dei rispettivi file di log (/var/log/messages, /var/log/httpd/access.log, ...).

Vediamole una per una.

Il file /var/log/messages viene ruotato su base settimanale (weekly) e ne vengono tenute al massimo 5 copie (/var/log/messages.1, /var/log/messages.5).

Le versioni più vecchie vengono perse. Successivamente alla rotazione (ma prima di comprimere la vecchia copia del log), viene mandato un segnale di tipo SIGHUP a syslogd, il quale viene in questo modo forzato a rileggere il proprio file di configurazione e a ricreare eventuali file di log che nel frattempo fossero stati cancellati.

Nel caso del file di log di Apache, esso viene ruotato ogni qualvolta superi la dimensione prestabilita di 100kb. Anche in questo caso vengono tenute 5 copie del file, tuttavia le versioni più vecchie non vengono perse, bensì spedite per e-mail all'indirizzo specificato (in forma non compressa).

L'ultima parte del file di configurazione vale invece per tutti i file contenuti nella directory /var/log/news.

A prescindere dal numero di file presenti nella directory, viene considerata come un'unica rotazione, perciò viene ad esempio generata un'unica e-mail di errore.

Rispetto ai casi precedenti, la rotazione viene eseguita su base mensile, senza generare errori nel caso di file inesistenti (missingok). Le vecchie copie dei file di log non vengono compresse.

Analisi "manuale" dei file di log

Il metodo più utilizzato per tenere sotto controllo i file di log è quello di aprirli (in "sola lettura", dato che vengono aggiornati di contnuo dal sistema) con un editor oppure di scorrerli utilizzando il comando more.

Qualora li si voglia avere sempre sott'occhio conviene usare invece il comando "tail -f", che mostra l'evolvere del/i file in tempo reale. Esso accetta sulla linea di comando anche più file alla volta:

```
# tail -f /var/log/messages /var/log/secure
```

Tuttavia in questo modo l'output dei diversi comandi apparirà mischiato e sarà difficile da tenere sott'occhio. Per migliorare le cose si può utilizzare il programma XWatch (ftp://ftp.icce.rug.nl/pub/unix), il quale mostra in una finestra grafica le linee provenienti dai diversi file di log usando colori diversi.

Strumenti di analisi automatica dei file di log

Lo strumento più interessante e semplice da utilizzare di questa categoria è Swatch (http://www.stanford.edu/~atkins/swatch), il quale permette di analizzare in tempo reale i file di log, segnalando eventuali problemi di sicurezza o malfunzionamenti dei servizi mediante diversi metodi, fra cui la posta elettronica, gli SMS oppure un programma definito dall'utente.

Una alternativa è Xlogmaster (ftp://ftp.gnu.org/pub/gnu/xlogmaster/), il quale serve al duplice scopo di mostrare in modo ordinato i file di log e di attivare degli allarmi nel caso si verifichino determinati eventi configurabili dall'utente. Esso supporta il concetto di plugin, ovvero possono essere aggiunte nuove funzioni senza modificare il programma principale, mediante script esterni.

Nonostante la potenza, si tratta di un programma molto semplice da usare, grazie al fatto di essere interamente gestibile mediante menù.

Si tratta certamente di software molto utili, in quanto permettono di essere avvisati in tempo reale mentre l'evento sta accadendo e non "il giorno dopo" quando i log potrebbero essere stati già compromessi.

Altri programmi utili possono essere reperiti sul sito FTP ftp://sunsite.unc.edu nella directory /pub/Linux/system/admin/log/.

Conclusioni

I file di log non servono certamente ad evitare danni o ad un pirata di entrare nella nostra macchina, ma sono solamente degli strumenti di notifica, che servono tuttavia per rendersi almeno conto che qualcosa è successo (ed è già un ottimo punto di partenza per iniziare ad intervenire).

Quello che abbiamo visto è solo un assaggio. Nelle prossime puntate parleremo ancora molto sull'argomento sicurezza, in quanto porremo la nostra attenzione su quello che forse è il "buco di sicurezza" più importante, ovvero il collegamento in rete.

Nel frattempo per ulteriori informazioni sull'argomento potete trovare degli ottimi spunti nei siti http://www.linuxsecurity.com e http://www.securityfocus.com.

MC