Aula 15 - Sistema de Log e Auditoria

Os sistemas de mensagens e logs dos Unix foram muito criticados por um bom tempo pelo fato de não serem padronizados e muitas vezes apresentarem algumas lacunas ou deficiências. Entretanto esse cenário, no universo Linux, tem um horizonte bem melhor desde muitos anos, pois para a sorte dos administradores, existem duas aplicações que de forma satisfatória implementam um subsistema de informações e logs, são eles: O syslogd e klogd.

O klogd é o sistema de log do kernel. O funcionamento dele depende de um buffer que armazena todas as informações notificadas pelo kernel que pode posteriormente encaminhá-las para o sistema de log do syslog. Por sua vez, o syslog é mais encorpado e flexível. Este sistema é regido basicamente por um arquivo de configuração, o /etc/syslogd.conf.

A diferença essencial entre ambos é de que o klogd executa em nível de kernel enquanto que o syslog executa em nível de usuário (userspace).

15.1 - Klogd

O sistema de log do kernel é um sistema independente que funciona como um daemon ou serviço, e tem como principal função obter as mensagens escritas pelo núcleo do sistema e repassá-las a algum subsistema e em alguns casos diretamente ao usuário.

Para tanto, o klogd pode fazer a leitura dessas mensagens de duas maneiras. Uma, feita diretamente de um sistema virtual de arquivos, que é o /proc. Em segunda opção, o klogd pode fazer a obtenção das mensagens através de uma camada de interface de sistema que na prática é uma composição de chamadas de sistema. O principal arquivo, fonte para o sistema do klogd, é o arquivo /proc/kmsg. Este arquivo armazena os eventos gerados pelo kernel na medida em que os mesmos acontecem. O sistema de klog os lê e os envia para um arquivo texto ou então os repassa para o syslog.

Mais detalhadamente, entende-se que o kernel reserva um buffer para escrever seus eventos e atividades, e a partir deste buffer, o klogd pode coletar informações importantes para seu funcionamento.

Um utilitário que trabalha diretamente com o buffer do kernel é o dmesg, que após o inicio do sistema faz a leitura dentro desse buffer de eventos do kernel e os transcreve para /var/log/dmesg. Isso é importante para o processo de boot, pois o kernel armazena o carregamento e seus passos iniciais, permitindo análises do boot a posteriori.

Uma vez recebida as mensagens do sistema operacional, o klogd pode repassá-las ao syslog, quando o mesmo existir. Para que as mensagens sejam repassadas corretamente, o klogd as classifica em níveis de prioridade. Estes níveis são definidos de 0 até 7, onde o nível 0 é o mais prioritário e supostamente deve ser alarmado no sentindo que o usuário possa ter informações de depuração.

15.1.1 - Resolução de símbolos

Como já mostrado no item de compilação do kernel, existe uma tabela que indica ao usuário quais são as referências dos símbolos utilizados por um determinado kernel. Essa tradução de endereços em símbolos é gerada em momento de compilação e reside no arquivo System.map.

Uma boa prática é manter os arquivos de System.map no mesmo diretrório das imagens do kernel. A partir dessa tabela de referência, o sistema de klogd pode traduzir de maneira mais inteligível os erros e ações de um kernel.

Essa tarefa de tradução é bastante simples e eficaz quando realizada em kernel estáticos, isto é, monolíticos. Isto por conta do kernel poder endereçar todos seus símbolos internos, uma vez que estes não serão alteradas em execução.

Por outro lado, quando o kernel é modular, existem vários problemas relacionados ao endereçamento dos símbolos, pois a dificuldade aumenta quando um kernel tem que identificar qual é o endereço dentro de um módulo, uma vez sabendo que este pode ser carregado ou descarregado da memória em tempo de execução.

A solução para sistemas modulares, é fazer com que através de chamadas de sistemas, o kernel receba o endereço base de um módulo quando este é carregado na memória. Isto garante ao menos que uma referência seja identificada para um determinado erro ou mensagem quando os mesmos forem encontrados.

Além disso, alguns desenvolvedores, podem adicionar aos códigos de seus módulos, chamadas de registro de referência que informam ao kernel no momento do carregamento qual deve ser a tabela de mapeamento interna daquele módulo.

15.1.2 - Depuração e sinais para o klogd

A depuração e visualização de mensagens do log do kernel são, na maioria dos casos, repassadas diretamente ao syslog do sistema, entretanto, estas também podem ser repassadas, paralelamente a algum arquivo ou saída padrão pré definida. É o caso, de saídas de log nos terminais do sistema. Os níveis mais alarmantes, nível 1 ou 0, deveriam ser impressos diretamente em um terminal de uma sessão ativa qualquer.

Níveis de prioridade de logs do kernel

0 Situação de emergência (KERN_EMERG).
1 Um erro crucial ocorreu (KERN_ALERT).
2 Um erro crítico ocorreu (KERN_CRIT).
3 Um erro ocorreu (KERN_ERR).
4 Um alerta foi gerado (KERN_WARNING).
5 Tudo está certo, mas algo deve ser verificado (KERN_NOTICE).
6 Verborrágico e informações detalhadas (KERN_INFO).
7 Mensagem de acompanhamento (KERN_DEBUG).

Isto é feito de maneira flexível e pode ser alterada através da variável de kernel (sysctl.kernel.printk).

Por padrão, tudo que não for depuração. Ou seja, todos os níveis, exceto o nível 7 são submetidos para os terminais ativos além de serem enviados para o sistema de log (syslog). Para que o nível de alarme visível em terminal seja alterado, é necessário parametrizar o daemon de klogd com o atributo -c. Por exemplo a execução de "klogd -c 3", faria com que somente mensagem com níveis inferiores a 3 fossem redirecionadas ao terminal. Uma maneira alternativa de configurar o que deve ser redirecionado ao terminal é utilizando o comando dmesg, através do parâmetro n, como "dmesg -n 5" que faz com que os níveis 5 ou menores sejam direcionados ao terminal.

Nem todos os sinais de processos podem ser entendidos pelo processo do daemon do klogd. Apesar de ainda não termos abordado o conceito de processos ou sinais, o que é importante saber nesse momento, é de que o processo de klogd pode receber interrupções que o fazem parar ou reconfigurar seu modo de operação. Os principais sinais aceitos são: SIGTERM, SIGHUP, SIGKILL e SIGINT, que vão informar ao klogd que fechem suas fontes de informação e pare de executar de forma natural.

Outros sinais como SIGUSR1 que serve para que o kernel recarregue todos os símbolos de módulos. Isto é necessário quando módulos são carregados e descarregados da memória. De outra forma, o SIGUSR2 serve para informar ao klogd que este deve recarregar somente os símbolos estáticos e também os de módulos.

Finalmente, SIGSTP e SIGCONT são sinais que fazem com o klogd reinicie suas atividades. O primeiro faz com que o daemon fique executando em um loop de espera enquanto que o segundo vai fazer com que o mesmo retome a execução.

15.2 - Syslogd

O syslog, além de um sistema de logs, pode ser visto como um protocolo para encaminhamento e armazenamento de mensagens. Inicialmente ele foi uma aplicação criada para auxiliar ao programa sendmail quando o mesmo devia registrar seus eventos. Atualmente ele se tornou um padrão "de fato" no universo Unix e o IETF mantém um projeto para normatizá-lo. Nessa linha, existem várias opções de syslog como o syslog-ng, sysklog ou no caso das últimas versões de Debian, que utilizam o rsyslog.

O syslog possui níveis, assim como o klogd, e pelo fato de sua operação ser baseada em um protocolo; o mesmo tem sua comunicação feita através de sockets. Isso possibilita o envio de uma mensagem para um servidor de syslog remoto, através do redirecionamento de tal mensagem para o socket correspondente. Por padrão, a conexão IP, pode ser realizada através de sockets escutando na porta 514 e transportados por UDP.

Basicamente, o que o syslogd efetua, é o recebimento de uma mensagem e seu armazenamento mediante uma configuração pré estabelecida em um arquivo. Por definição o syslog trabalha com o relógio do sistema que lhe dá timestamps para cada um dos eventos recebidos.

15.2.1 - Syslogd.conf

Quanto à configuração do daemon do syslog, devemos nos preocupar basicamente com o arquivo de configuração que reside em /etc/, pois nele é possível definir o nível do evento que se

deseja armazenar para cada um dos recursos reconhecidos e além disso determinar qual o destino desse armazenamento.

Lembrando, que no caso do Debian, devemos encontrar o rsyslog. Logo o arquivo em /etc será o /etc/rsyslogd.conf.

Para a configuração do syslog, em relação aos recursos, é possível classificá-los como:

- * - todos os recursos
- auth - execuções ligadas à autorização
- authpriv - execuções privadas de atuorização
- cron - execuções relativas ao agendador
- daemon - servidores do sistema (daemons)
- ftp - relativo ao daemon FTP
- kern - mensagem proveniente ou relativa ao kernel
- local 0-7 - variantes de mensagens locais. Vão do nível 0 até o 7
- lpr - sistema de impressão e spoolling de impressão
- mail - relativo ao sistema de email como sendmail ou outros SMTP
- mark - sistema de timestamp
- news - sistema de Usenet
- syslog - relativo ao próprio sistema de log
- user - relacionados às sessões ou comandos de usuários

Em relação aos níveis de alarme dos eventos, os mesmos podem ser definidos de forma idêntica aos níveis de prioridade do kernel.

- emerg
- alert
- crit
- err
- warning
- notice
- info
- debug

Finalmente, o último parâmetro importante para a configuração do arquivo syslogd.conf é o local onde serão armazenadas as informações de cada um dos recursos selecionados. As possibilidades para ações ou destinos podem escolhidas na lista:

- filename escreve em um arquivo determinado
- @hostname escreve em um servidor através do nome do host
- @ipaddress escreve em um servidor através de um endereço
- |fifoname escreve o log para um pipe
- user1,user2,... escreve para o usuário user1, user2 e assim por diante
-* escreve para todos usuários logados

No caso da ação final ser um arquivo, é possível que o mesmo receba as seleções utilizando a sincronização do sistema de arquivos. Para que a sincronização, esteja desligada, basta adicionar um sinal de subtração (-) antes do nome do arquivo final. Isto dará muito mais informações ao syslog, porém a um custo muito elevado de uso de disco.

Algo necessário notar é que toda e qualquer alteração na configuração do syslog exige o reinicio do serviço. O scripts de inicialização como já de praxe podem ser encontrados em /etc/init.d/rsyslogd e /etc/init.d/klogd.

Um exemplo de arquivo de configuração para o rsyslogd.conf pode ser descrito nas linhas abaixo:

```
$FileOwner root
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$IncludeConfig /etc/rsyslog.d/*.conf
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err
```

Além de seleções e ações é possível manipular operadores seletores para detalhar ainda mais a informação a ser armazenada. Os principais seletores são:

.
Ex: mail.err
Sem seletor especificado, indica que todos níveis do serviço mail que sejam superiores a err serão selecionados

;
Ex: mail.info;mail.err
Indica que os logs do serviço mail para os níveis de info até err serão selecionados

=
Ex: mail.=info
Indica que somente os log do serviço de email para o nível de info serão selecionados

!=
Ex: mail.warning;mail.!=alert
Seleciona as mensagens do serviço de mail que vão desde o nível de warning excetuando-se o nível de alert.

!
Ex: mail.debug;mail.!err
Seleciona para o serviço de mail, todos os níveis que vão desde debug até o nível err (excluindo err).

15.2.2 - Arquivos de log padrão

Dentro do sistema de log existem alguns arquivos diferenciados, como por exemplo o lastlog

e wtmp. Estes servem para armazenar o registro de conexões dos usuários no sistema. São diferentes porque são arquivos binários e podem ser acessados através das ferramentas como o lastlog e last.

Além do wtmp e lastlog outros arquivos comumente encontrados são:

- syslog
- messages
- auth.log
- secure
- boot.log
- dmesg
- boot.msg
- debug
- faillog
- sudo.log
- setuid.log

Uma ferramenta interessante, que pode servir para conexão direta ao syslog é a ferramenta logger. Com ela é possível enviar mensagens ao socket padrão do syslog. Um exemplo de seu uso seria:

#> logger -t TAG “OLA ESTOU LOGANDO ESSE TEXTO”

ou

#> logger -p local3.info -t PROGRAMA “EXECUTANDO OPERACAO NORMAL”

Na primeira opção, o logger enviará uma mensagem para o daemon de syslog com nível notice para a facilidade ou serviço user e adicionando um rótulo igual à TAG nesse evento. Já no segundo, a mensagem será enviada como nível de info até o serviço local3 e com rótulo igual a PROGRAMA. Lembrando sempre que o armazenamento no syslog deve ocorrer associado a uma estampa de tempo dessas ocorrências.

Além da ferramenta logger, seria possível utilizar as chamadas de sistema openlog( ) e closelog( ) que fazem a mesma ação de encaminhamento ao syslog. Mais informações podem ser vistas nos manuais dessas chamadas.

15.2.3 - A ferramenta para rodízio de logs

A maioria destes arquivos de registro tende a crescer continuamente. Para poder conservá-los por um tempo ainda razoável, existe um sistema de apoio que gera, conforme uma configuração pré determinada, os arquivos compactados dos registros antigos e também determina o descarte ou criação dos locais dos novos logs.

A ferramenta mais popular e presente em quase todas as distribuições é o logrotate. Sua configuração está descrita em /etc/logrotate.conf e também nos arquivos do diretório /etc/logrotate.d.

Vale a pena entender quais são os principais parâmetros e seus significados, porque somente assim é possível ao administrador de um sistema definir ou corrigir a política de armazenamento e

retenção dos registros de um determinado sistema.

De forma bastante rápida, os parâmetros do serviço de rodízios podem ser:

compress - compacta todas versões geradas
daily, weekly, monthly - executa o rodízio na frequência estipulada
delaycompress - compacta as versões geradas exceto a corrente e a primeira gerada
endscript - fim de ações que se iniciam em postorotate ou prerotate
errors emailaddr - envia erros para o endereço de email “emailaddr”
missingok - não retorna erro caso o arquivo de log não exista
notifempty - não faz o rodízio caso o arquivo de log esteja vazio
olddir dir - especifica um diretório para encaminhar as versões mais velhas
postrotate - executa um script para ser invocado após o rodízio
prerotate - executa um script antes do rodízio
rotate n - determina o número de versões de log a serem armazenadas
sharedscripts - executa um script para todo o grupo de logs
size=logsize - executa o rodízio baseado no tamanho do arquivo de log “logsize”
create mod user group - cria um novo arquivo logo após o rotate com os respectivos atributos
nocreate - não cria, mantém o mesmo arquivo existente

Portanto, para cada arquivo de log, deve haver uma configuração no sistema de logrotate que indique qual será a ação de retenção e como será essa retenção. Para cada entrada existente no diretório /etc/logrotate.d devemos encontrar arquivos com os nomes dos respectivos arquivos de log. Por exemplo, os arquivos /etc/logortate.d/apache2 e /etc/logrotate.d/ppp poderiam ser assim descritos:

```
/var/log/apache2/*.log {
    weekly
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    postrotate

        if [ -f "`. /etc/apache2/envvars ; echo $
  {APACHE_PID_FILE:-/var/run/apache2.pid}`" ]; then
            /etc/init.d/apache2 reload > /dev/null
        fi
    endscript
                        }

/var/log/ppp-connect-errors {
        weekly
        rotate 4
        missingok
        notifempty
        compress
        nocreate
```

}

Se um arquivo de log não for encontrado no diretório logrotate.d, supostamente ele deverá seguir as regras de rodízio “gerais” existentes em /etc/logrotate.conf. Como no exemplo:

```
weekly
rotate 4
create
compress
include /etc/logrotate.d

/var/log/wtmp {
  missingok
  monthly
  create 0664 root utmp
  rotate 1
}
```

De toda a sorte, é possível realizar testes em cima de um configuração de logrotate. Isto é feito através do comando manual de logrotate: “logrotate -f /etc/logrotate.conf”. Caso contrário, é necessário aguardar a execução do crontab (agendador de tarefas), que instala um entrada para a execução diária do logrotate.

Aula 16 - Agendador de tarefas

16.1 - Comandos repetitivos

É fácil entender a necessidade de termos um sistema de execução de tarefas agendadas. Em muitos casos, o administrador deseja que tarefas rotineiras sejam executadas, sem ter que contar com a intervenção humana direta. Por isso, os agendadores são muito utilizados para processos em lotes, execuções de backup, atualizações de softwares, entre outras tarefas. Em geral é muito eficiente para todas aquelas que demandam a facilidade da automatização.

Os principais aplicativos para agendamento de tarefas no Linux são o crontab e at. Ambos tem um modo de trabalho parecido, pois através de uma agenda, são orientados a processarem determinadas rotinas.

16.2 - Cron

O crontab é um software de agendamento, que quando instalado, pode ser executado para cada um dos usuários do sistema ou então em um nível de sistema.

Para a execução do crontab pelo sistema, nos referimos ao diretórios referentes às frequências de execuções como por exemplo, /etc/cron.daily, /etc/cron.monthly e /etc/hourly entre outros. Dentro de cada um desses diretórios é possível encontrar arquivos de scripts ou aplicações invocadas na frequência de seu diretório, isto é, diariamente, toda hora, todo mês, etc.

Um exemplo de arquivos dentro /etc/cron.daily poderia ser:

```
-rwxr-xr-x  1 root root  314 2009-02-10 11:45 aptitude
-rwxr-xr-x  1 root root  502 2008-11-04 19:43 bsdmainutils
-rwxr-xr-x  1 root root  89 2009-01-26 16:55 logrotate
-rwxr-xr-x  1 root root  954 2009-03-19 07:17 man-db
-rwxr-xr-x  1 root root  646 2008-11-04 19:37 mlocate
-rw-r--r--  1 root root  102 2008-11-12 13:47 .placeholder
-rwxr-xr-x  1 root root 2149 2008-11-17 07:52 popularity-contest
-rwxr-xr-x  1 root root 3349 2008-11-12 13:47 standard
-rwxr-xr-x  1 root root 1309 2009-01-23 16:33 sysklogd
-rwxr-xr-x  1 root root  469 2008-11-17 08:16 sysstat
```

Pontualmente, usando como exemplo o arquivo de logrotate, deve-se entender que esse arquivo vai invocar a execução do sistema de rodízio de logs todos os dias.

Outra forma de execução do crontab é através da instalação pessoal para cada um dos usuários. Para tal, basta realizar a edição do arquivo pessoal de crontab para um determinado

usuário. Isto é feito através de:

#> crontab -e

Edita a tabela de agendamento para o usuário corrente.

#> crontab -u root -l

Lista a tabela de agendamento para o usuário root.

A sintaxe da tabela de agendamento do crontab é baseada na data do agendamento e na sua tarefa em si. Exemplos de linha de crontab seriam:

```
1 2 22 1 *      ntpdate ntp.ansp.br
1****           updatedb
15 3 * * 0      /sbin/sunday.sh
```

Onde os 5 primeiros campos determinam a data da agenda e por fim o comando a ser acionado.

Para cada uma das colunas de data, é preciso conhecer as especificações:

minuto hora dia do mês mês dia da semana

Portanto, na primeira linha do crontab, entende-se que o minuto 1 da hora 2 do dia 22 do mês 1 deve executar o ntpdate, que é um aplicativo de sincronização de data/horário.

Já na terceria linha, nota-se que às 03:15 h de todos os domingos, o sistema deverá invocar a aplicação /sbin/sunday.sh. Para que as horas e dias sejam utilizados corretamente, é aconselhável verificar o manual da seção 5 do crontab. Exemplo disso são os dias da semana, onde 0 e 7 identificam o domingo.

16.3 - At

Bem mais simples que o aplicativo cron, é o aplicativo at, que executa exatamente o que seu nome intuitivamente nos informa. Este determina a execução de um script em lote pontualamente para uma data.

Exemplos de execução do at podem ser vistas a seguir:

#> at -m 15:45 < /bin/script.sh

Determina a execução de /bin/script.sh às 15:45 h e envia um email após a conclusão para o usuárioque realizou o agendamento.

#>at -l

Lista as tarefas agendadas para o at.

#>atrm 1

Remove o índice 1 da lista de execuções de tarefa do at.