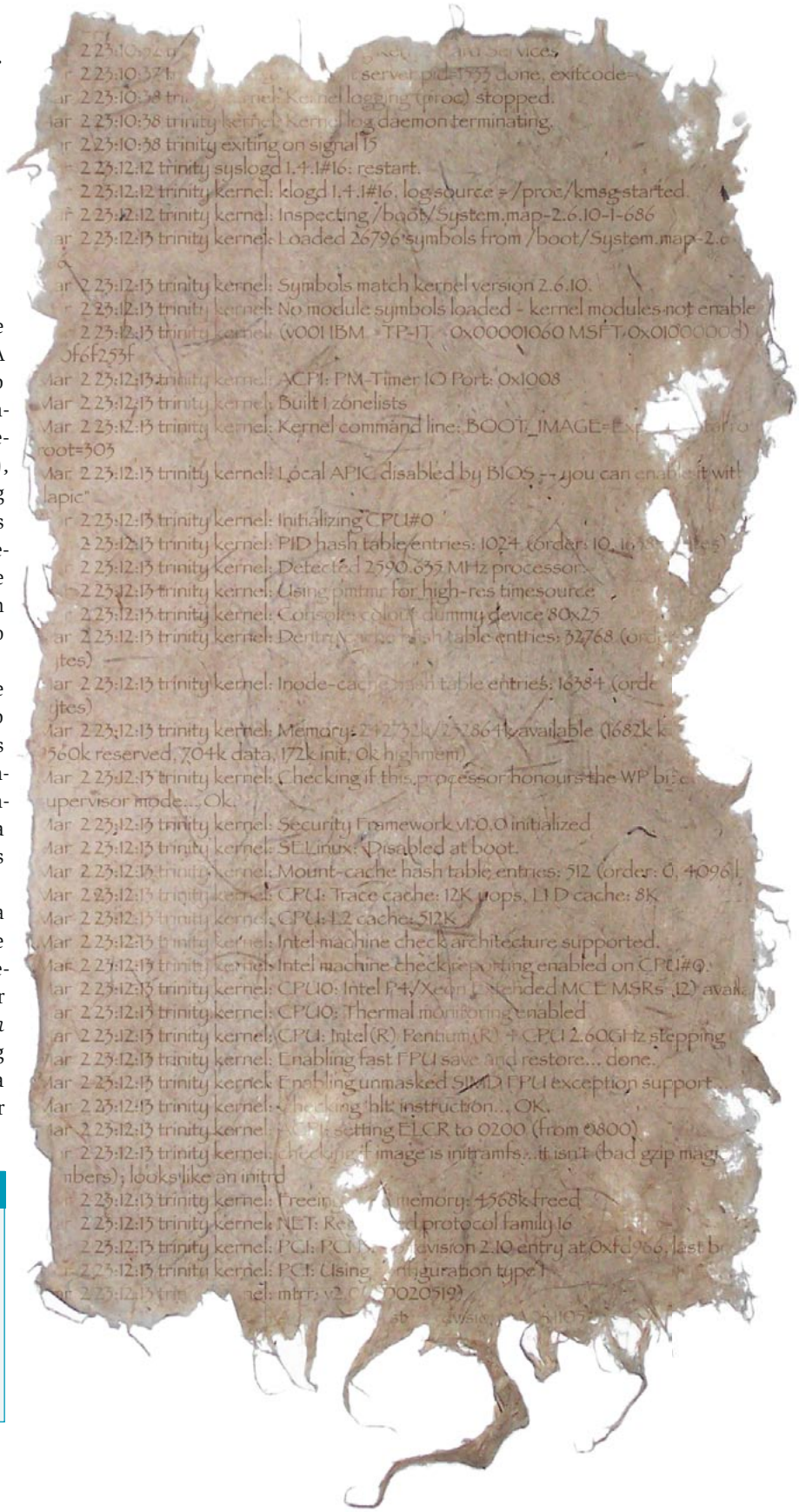
Configurando um servidor de *log* com o *syslog-ng*

Registros históricos

O *syslog-ng* (*syslog new generation*) veio para suprir várias das deficiências do bom e velho *syslog* – e é bem mais flexível que seu antecessor. Aprenda neste artigo como montar um servidor de log com essa poderosa ferramenta.

POR FLAVIO FERRAZ



O *syslog* é o *daemon* padrão para registro de eventos em sistemas Unix e similares. A estrutura do seu arquivo de configuração (`syslog.conf`) é bem simples e se resume basicamente a duas opções: *recursos* (*Facilities*, erroneamente traduzido como "facilidades" por alguns), que especificam a procedência da mensagem de log – ou seja, que tipo de mensagem de log queremos gravar (como, por exemplo, ocorrências em sistemas de email e autenticação) – e *prioridades*, que definem qual o nível de importância da mensagem a ser registrada e a "etiquetam" com prefixos como *alert*, *crit* e *emerg*, entre outros. (veja o **t**)

Essa estrutura simples funciona razoavelmente bem numa configuração padrão; porém, quando necessitamos de recursos como classificação mais abrangente dos eventos do sistema, tratamento avançado de rede, verificação de integridade das mensagens ou mesmo criptografia, o *syslog* começa a mostrar ineficiência para atender às exigências dos sistemas Unix atuais.

O *syslog-ng* (*syslog new generation*) [1] surgiu para suprir essas deficiências. Ele permite a adição de filtros entre as mensagens e a verificação de integridade, possui suporte a criptografia e pode ser utilizado em conjunto com programas como o *ssh* para autenticar ou criptografar mensagens de log que serão enviadas a *hosts* remotos. O programa foi originalmente escrito, e ainda é mantido, por Balazs Scheidler.

Quadro 1: Sintaxe do *syslog.conf*

Um trecho do arquivo de configuração do syslog (`syslog.conf`), que segue o formato: `recurso.prioridade arquivo de log`.

```
mail.info  -/var/log/mail.info
mail.warn  -/var/log/mail.warn
mail.err /var/log/mail.err
```

Obtendo e instalando o *syslog-ng*

O código fonte do programa pode ser obtido em [1]. Depois de descompactá-lo (com `tar -zxvf syslog-ng-<versão>.tar.gz`) basta entrar no diretório recém-criado (`cd syslog-ng-<versão>/`) e, como root, compilá-lo com a tradicional seqüência `./configure`, `make` e `make install`.

Para instalar o *syslog-ng* em um sistema Debian, basta usar o pacote `.deb` disponível em [2] ou recorrer ao utilitário *apt-get*, com o comando: `apt-get install syslog-ng`.

Configurando o *syslog-ng*

Por padrão, o arquivo de configuração do *syslog-ng* se chama `syslog-ng.conf` e pode ser encontrado no diretório `/etc`. Nesse arquivo podemos definir regras para a gravação de mensagens de *log* a partir das seguintes diretivas: `options{}`, `source{}`, `destination{}`, `filter{}` e `log{}`. Cada uma delas pode conter configurações adicionais que são usualmente separadas por um ponto-e-vírgula.

A sintaxe utilizada no arquivo `syslog-ng.conf` é muito similar à estrutura usada em linguagens de programação e lembra um pouco a linguagem C. As declarações são terminadas por ponto-e-vírgula; espaços em branco são ignorados e podem ser usados para aumentar a legibilidade do arquivo (veja a **listagem 1**). ➠

Quadro 2: Recomendações da NBR ISO/IEC 17799

O registro dos serviços do sistema e das atividades dos usuários é tão importante que a norma de segurança da informação NBR ISO/IEC 17799 no item 9.7.1 (registro de eventos) recomenda o seguinte:

"Convém que trilhas de auditoria, registrando as exceções e outros eventos de segurança relevantes, sejam produzidas e mantidas por um período de tempo acordado, para auxiliar em investigações futuras e na monitoração do controle de acesso. Convém que os registros (*logs*) de auditoria também incluam:

a) identificação dos usuários;

b) datas e horários de entrada (*log-on*) e saída (*log-off*) no sistema;

c) identidade do terminal ou, quando possível, a sua localização;

d) registro das tentativas de acesso ao sistema aceitas e rejeitadas;

e) registro das tentativas de acesso a outros recursos e dados aceitas e rejeitadas;

Certos registros (*logs*) de auditoria, podem ser guardados como parte da política de retenção de registros ou devido aos requisitos para a coleta de evidências."

Listagem 1: Exemplo de arquivo syslog-ng.conf

```
01 options { long_hostnames(off); sync(0); };
02 source src { unix-dgram("/dev/log"); internal(); };
03 source src { unix-dgram("/dev/log"); internal(); udp(); };
04 destination authlog { file("/var/log/auth.log" owner("root") group("adm") perm(0640)); };
05 destination syslog { file("/var/log/syslog" owner("root") group("adm") perm(0640)); };
06 destination console { usertty("root"); };
07 destination console_all { file("/dev/tty8"); };
08 filter f_authpriv { facility(auth, authpriv); };
09 filter f_syslog { not facility(auth, authpriv); };
10 log { source(src); filter(f_authpriv); destination(authlog); };
11 log { source(src); filter(f_syslog); destination(syslog); };
```

Tabela 1: Opções globais

Opção	Descrição
`schain_hostnames( yes | no )`	Além de mostrar o nome do *host* que enviou uma mensagem de *log* via protocolo UDP ou TCP, mostra o nome de todos os *hosts* que a manipularam (valor padrão=`yes`).
`sskeep_hostnames( yes | no )`	Nomes dos *hosts* "confiáveis" para envio de mensagens via protocolo TCP/UDP (valor padrão=`no`).
`ssuse_fqdn( yes | no )`	Grava o nome completo dos *hosts* que enviaram mensagens via protocolo TCP/UDP (valor padrão=`no`).
`ssuse_dns( yes | no )`	Resolve o endereço IP dos *hosts* que enviaram mensagens via protocolo TCP/UDP (valor padrão=`yes`).
`sstime_reopen( number )`	Número de segundos que uma conexão TCP aguarda antes de reconectar (valor padrão=`60`).
`sslog_fifo_syze( number )`	Número de mensagens que vão para a fila na memória quando o *syslog-ng* está ocupado. Quando a fila está cheia, as novas mensagens são bloqueadas (valor padrão=`100`).
`sssync( number )`	Número de mensagens que são gravadas no arquivo de *log* antes dele ser sincronizado (valor padrão=`0`)
`ssowner( string )`	Dono dos arquivos de *log* criados pelo *syslog-ng* (valor padrão=`root`).
`ssgroup( string )`	Grupo ao qual pertencem os arquivos de *log* criados pelo *syslog-ng* (valor padrão=`root`)
`ssperm( number )`	Permissões de acesso aos arquivos de *log* criados pelo *syslog-ng*.
`ssdir_owner( string )`	Dono dos diretórios criados pelo *syslog-ng* (valor padrão=`root`).
`ssdir_group( string )`	Grupo ao qual pertencem os diretórios criados pelo *syslog-ng* (valor padrão=`root`).
`ssdir_perm( number )`	Permissão de acesso aos diretórios criados pelo *syslog-ng* (valor padrão=`0700`).

Tabela 2: *Source Drivers* para o *syslog-ng*

Source	Descrição
`internal()`	Recebe mensagens oriundas do *daemon syslog-ng*.
`file("nome_do_arquivo" [opções])`	Recebe mensagens do arquivo especial `/proc/kmsg`.
`pipe("nome_do_arquivo" )`	Recebe mensagens de um *named pipe*.
`unix_stream("nome_do_arquivo" [opções])`	Recebe mensagens de *sockets* Unix no modo *stream* (com conexão).
`unix_dgram("nome_do_arquivo" [opções])`	Recebe mensagens de *sockets* Unix no modo *datagram* (sem conexão).
`tcp([ip(adrress)] [port(#)] [max-connections(#)])`	Recebe mensagens de um determinado *host* remoto via protocolo TCP em uma porta específica (valor padrão=`514`), podendo receber qualquer mensagem TCP que entre pela interface de rede local (valor padrão=`all`).
`udp([ip(address)] [port(#)])`	Recebe mensagens de um determinado host remoto via protocolo UDP em uma porta específica (valor padrão=`514`), podendo receber qualquer mensagem UDP que entre pela interface de rede local (valor padrão=`all`).

Opções Globais

As opções globais são configuradas através da diretiva `options{}` do arquivo `syslog-ng.conf`. São genéricas e podem, inclusive, ser utilizadas dentro de outras declarações. A **tabela 1** lista algumas das opções mais comuns.

Sources

A diretiva *source* identifica a origem das mensagens de *log*. Para isso, ela possui parâmetros conhecidos como *drivers*. Veja o exemplo a seguir:

```
source s_local { unix-stream("/dev/log"); internal(); };
```

Na linha acima nós criamos um *source* chamado `s_local` que, por sua vez, irá obter mensagens de log do *socket* `/dev/log` (eventos locais do sistema) e dos processos locais do *syslog-ng*. O driver que utilizamos, `unix-stream`, é específico para capturar mensagens de registro vindas dos *sockets* de um sistema Unix.

O *syslog-ng* reconhece um número bastante alto de *source drivers*. Além de capturar *logs* de *sockets* do Unix, aceita mensagens enviadas via conexão TCP, protocolo UDP, *named pipes* ou arquivos especiais (definidos no `/proc`). Veja na **tabela 2** quais são os *source drivers* suportados.

Os drivers `tcp()` e `udp()` lêem mensagens oriundas de *hosts* remotos através dos protocolos TCP e UDP, respectivamente. A porta padrão utilizada pelo *syslog-ng* é a 514, que por sua vez fica em escuta em todas as interfaces.

O exemplo a seguir mostra a declaração `source` para os drivers `tcp()` e `udp()` com IP e porta definidos:

```
source s_tcp { tcp( ip(192.168.33.33) port (4455) ); };
source s_udp { udp(); };
```

No exemplo acima o *source* `s_tcp` irá receber mensagens de *log* provenientes do endereço IP 192.168.33.33 na porta 4455 e o `source s_udp` irá aceitar mensagens UDP vindas de qualquer endereço IP na porta 514 em todas as interfaces locais.

Tabela 3: Destinations

Driver	Descrição
`file("filename[$MACROS]")`	Grava mensagens para arquivos de *log* em formato texto padrão ASCII. Se o arquivo não existir, o *syslog-ng* irá criá-lo. As macros podem ser passadas como opções para os *drivers* ou no lugar do nome de um arquivo. Nesse caso, é possível simular nomes de arquivos dinâmicos.
`tcp("address" [port(#);])`	Transmite mensagens via protocolo TCP para uma porta específica (a porta padrão é a 514) em um dado *host* ou endereço IP.
`udp("address" [port(#);])`	Transmite mensagens via protocolo UDP para uma porta específica (a porta padrão é a 514) em um dado *host* ou endereço IP.
`pipe("pipename")`	Envia mensagens para *named pipes* como `/dev/xconsole`.
`unix_stream("filename" [options])`	Envia mensagens estabelecendo conexões do tipo *stream* para um *socket* do Unix como `/dev/log`.
`unix_dgram("filename" [options])`	Envia mensagens sem estabelecer conexão no modo *datagram* para um *socket* do Unix como `/dev/log`.
`usertty(username)`	Envia mensagens para usuários específicos no console.
`program("/caminho/para/o/programa")`	Envia mensagens para a entrada padrão do programa especificado.

Destinations

O *syslog-ng* pode ser configurado para enviar mensagens de *log* para vários destinos: arquivos ASCII, *named pipes*, *hosts* remotos via protocolos TCP e UDP, TTYs (terminais), *sockets* do Unix e entrada padrão de programas. A **tabela 3** mostra os tipos de destino permitidos no *syslog-ng* e que, assim como na declaração source, também são conhecidos como *drivers*.

Quando especificamos um nome de arquivo como destino para gravar mensagens, podemos usar macros para criar esse nome ou parte dele, de acordo com a conveniência desejada. (**tabela 4**) Por exemplo, para gravar mensagens de *log* em um arquivo que corresponda ao dia da semana, podemos usar a macro `WEEKDAY`:

Tabela 4: Macros suportadas em destinos *file()*

Macro	Significado
`PROGRAM`	O nome do programa que envia a mensagem.
`HOST`	O nome do *host* que originou a mensagem.
`FACILITY`	O recurso para o qual a mensagem foi gerada.
`PRIORITY` ou `LEVEL` (sinônimos)	Para designar o nível de prioridade.
`YEAR`	O ano atual.
`MONTH`	O mês atual.
`DAY`	O dia atual.
`WEEKDAY`	O nome do dia da semana (Monday, Tuesday, etc)
`HOUR`	A hora atual.
`MIN`	O minuto atual.
`SEC`	O segundo atual.

```
destination d_logdia { file("/var/log/$WEEKDAY"); };
```

Quanto às opções de destino, ainda é possível determinar o dono, o grupo e as permissões de acesso para o arquivo de *log*. Observe:

```
destination d_userlog { file("/var/log/userlog" owner(user) ➥
group(wheel) perm(0640)); };
```

No exemplo, o arquivo `userlog` pertencerá ao usuário `user`, grupo `wheel` e será criado com a permissão `640`.

Filters

As opções de filtragem do *syslog-ng* são extremamente poderosas e, ao contrário do *syslog*, que faz o roteamento das mensagens baseado apenas em prioridade/nível e recurso, o *syslog-ng* pode fazer a filtragem pelo nome do programa que emite a mensagem de *log* e do *host* que as envia. É possível inclusive montar filtros com expressões regulares e utilizar os operadores lógicos `and`, `or` e `not`. A declaração `filter{}` consiste do nome do filtro e um ou mais critérios conectados por operadores bem definidos. (**tabela 5**) Veja a seguir alguns exemplos de uso da declaração `filter{}`: ➡

```
filter f_authpriv {facility(auth, ➥
authpriv);};
filter f_debug {not facility(auth, ➥
authpriv, news, mail);};
filter f_messages {level(info .. warn) ➥
and not facility(auth, authpriv, cron, ➥
daemon, mail, news); };
filter f_suportematch { host("suporte") ➥
and match("sshd"); };
```

No primeiro exemplo, o filtro `f_authpriv` pega todas as mensagens gravadas pelos recursos `auth` e `authpriv`. O segundo filtro, `f_debug`, captura todas as mensagens de *log* para todos os recursos, com exceção de `auth, authpriv, news` e `mail`. O terceiro filtro, `f_messages`, captura todas as mensagens de nível `info` e `warn` para todos os recursos com exceção de `auth, authpriv, cron, daemon, mail` e `news`. Já o quarto filtro, `f_suportematch`, captura os *logs* provenientes do *host suporte* que contenham a *string* `sshd`.

Log Statements

A diretiva `log` serve para combinar os elementos que foram definidos nas outras declarações (`source, filter` e `destination`), a fim de elaborar as regras para gravação das mensagens de *log*. Veja um exemplo:

```
source s_local { unix-stream("/dev/➥
log"); internal(); };
filter f_suportematch { host("suporte") ➥
and match("sshd"); };
destination d_userlog { file("/var/log/➥
userlog" owner(user) group(wheel) ➥
perm(0640)); };
log { source(s_local); filter(f_➥
suportematch); destination(d_userlog); };
```

A diretiva `log {}` concentra as declarações de origem, filtro e destino para definir as opções com as quais as mensagens serão gravadas. Na última linha do exemplo, usamos o *source* `s_local` que especifica a origem do *log* como o *Unix socket* `/dev/log`; o filtro `f_suportematch` captura as mensagens de *log* provenientes do *host suporte* que contenham a *string sshd*; por fim, o destino `d_userlog` irá gravar o *log* no arquivo `/var/log/userlog`. O arquivo pertence ao usuário *user* e ao grupo *wheel* e terá a permissão de acesso *640*.

Configurando o servidor

Bem, agora que já conhecemos a estrutura do *syslog-ng* podemos começar a configurar o servidor de *log*. O arquivo de configuração do servidor é o `/etc/syslog-ng/syslog-ng.conf`. Para fazer com que o *syslog-ng* receba *logs* de outros *hosts*, devemos usar o driver `udp()` no estado *source*:

```
source src { unix-dgram("/dev/log"); ➥
internal(); udp(); };
```

Em nosso exemplo, iremos capturar os *logs* provenientes do *host linux*. As mensagens serão gravadas no arquivo `linux.log` que está dentro do diretório `/var/log`. Criaremos uma opção de destino chamada `linuxlog`, para gravar as mensagens oriundas do *host linux*. Perceba que com a opção `destination` podemos especificar um arquivo de *log* separado para cada máquina cliente:

```
destination linuxlog { file("/var/log/➥
linux.log" owner("root") group("adm") ➥
perm(0640)); };
```

Criaremos também uma opção de destino chamada `linuxsshd` para receber as mensagens de *log* da máquina *linux* que contenham a palavra `sshd` e enviá-las ao arquivo `/var/log/linuxsshd.log`:

```
destination linuxsshd { file("/var/log/➥
linuxsshd.log" owner("root") group("adm") ➥
perm(0640)); };
```

Configurando as opções de filtragem

Utilizando a declaração *filter*, criaremos um filtro chamado `f_linux`, para capturar os *logs* provenientes do host *linux*:

```
filter f_linux { host("linux"); };
```

Criaremos um filtro chamado `f_linuxmatch`, para capturar os *logs* vindos do *host linux* que contenham a *string* `sshd`:

```
filter f_linuxmatch { host("linux") and ➥
match("sshd"); };
```

Configurando as opções de *log*

Como vimos, é na configuração das opções de *log* que agregamos as declarações *source*, *filter* e *destination* para formar uma única opção para gravação das mensagens. Vamos criar uma entrada para capturar os *logs* da máquina *linux*:

```
log { source(src); filter(f_linux); ➥
destination(linuxlog); };
```

Agora criamos uma entrada para capturar os *logs* desta máquina que contenham a palavra `sshd`.

```
log { source(src); filter(f_linuxmatch); ➥
destination(linuxsshd); };
```

Tabela 5: Funções da diretiva *filter()*

Função(critério)	Descrição
`facility(facility-name)`	Recurso pelo qual a mensagem será filtrada.
`priority(priority-name)`	Prioridade das mensagens.
`priority(priority-name1, priority-name2)`	As prioridades, separadas por vírgulas, podem ser expressas como alta ou baixa.
`level(priority-name)`	As mesmas prioridades.
`program(program-name)`	Programa que emite a mensagem.
`host(hostname)`	*Host* do qual as mensagens foram recebidas.
`match(regular-expression)`	Expressões regulares que serão avaliadas no corpo da mensagem.
`filter(filter-name)`	Outros filtros a serem avaliados.

Tabela 6: Tabela verdade para pacotes UDP

	Porta aberta	Porta fechada	Porta em DROP	Porta em REJECT
Ação	um pacote UDP é recebido	um pacote UDP é recebido	um pacote UDP é recebido	um pacote UDP é recebido
Resultado	nada é retornado	um pacote *icmp3* é retornado	nada é retornado	um pacote *icmp3* é retornado

Reiniciando o *syslog-ng*

Depois de definir as opções necessárias no arquivo de configuração do servidor (`sylog-ng.conf`) devemos reiniciar o *daemon* do *syslog-ng*. Em um sistema Debian, por exemplo, basta executar como *root* os dois comandos mostrados a seguir:

```
# /etc/init.d/syslog-ng stop
# /etc/init.d/syslog-ng start
```

A porta padrão usada pelo servidor é a 514 e o protocolo da camada de transporte é o UDP. Use o comando `netstat` para saber se a porta 514 está aberta e se o processo do *syslog-ng* está "ouvindo" a rede por ela:

```
# netstat -anpd | grep 514
```

Configurando o cliente

O arquivo de configuração do cliente é o mesmo do servidor, ou seja, `/etc/sylog-ng/syslog-ng.conf`. No cliente *syslog-ng*, definimos como os arquivos de *log* serão enviados para o servidor.

Primeiro indicamos o *socket* `/dev/log` como o lugar para onde serão enviados os *logs* locais.

```
source src { unix-dgram("/dev/log"); ➥
internal(); };
```

Devemos especificar o endereço IP do *host* para onde a mensagem de *log* será enviada (servidor de *log*). Usamos para isso uma *declaração de destino*. Por exemplo, para enviar as mensagens de *log* ao servidor 192.168.0.161, a seguinte linha deverá ser acrescentada:

```
destination d_udp { udp("192.168.0.161" ➥
port(514)); };
```

O *syslog-ng* trabalha com os mesmos níveis e recursos de seu predecessor *syslog*. Vamos criar um filtro chamado `f_envio` que irá capturar mensagens de *log* nos níveis: `info`, `warn`, `err` e `crit` e recursos: `auth`, `authpriv`, `daemon`, `mail`, `news`, `cron` e `user`. As mensagens de *log* que vamos enviar para o servidor devem corresponder aos critérios estabelecidos nesse filtro. Para isso, a linha a seguir deverá ser acrescentada ao arquivo de configuração do *syslog-ng*:

```
filter f_envio { level(info, warn, err, ➥
crit) and facility(auth, authpriv, ➥
daemon, mail, news, cron, user); }; ➥
```

Listagem 2: Sugestão de *firewall*

```
01 #!/bin/bash
02 PF="/sbin/iptables"
03 CL="192.168.2.2"
04 SV="192.168.2.135"
05 $PF -P INPUT DROP
06 $PF -P FORWARD DROP
07 $PF -P OUTPUT ACCEPT
08 $PF -A INPUT -s $CL -d $SV -p icmp  --icmp-type 8 -j ACCEPT
09 $PF -A INPUT -s $CL -d $SV -p icmp --icmp-type 0 -j ACCEPT
10 $PF -A INPUT -s $CL -d $SV -p udp --sport 1024:65535 --dport 514 -j ACCEPT
11 $PF -A INPUT -s 0/0 -d $SV -p udp --dport 514 -j REJECT
```

Nas opções de *log*, novamente iremos agregar as declarações `source`, `filter` e `destination` para formar uma única opção de *log* personalizada. Assim, para enviar os tipos de *log* definidos no filtro `f_envio` ao destino `d_udp` (que corresponde ao servidor de *log* cujo endereço IP é 192.168.0.161), a linha a seguir deverá ser acrescentada ao arquivo de configuração do *syslog-ng*:

```
log { source(src); filter(f_envio); ➥
destination(d_udp); };
```

Finalmente, para enviar seu *log* personalizado com as opções de filtragem para o servidor, reinicie o *daemon* do *syslog-ng*. Em um sistema Debian podemos executar os seguintes comandos:

```
# /etc/init.d/syslog-ng stop
# /etc/init.d/syslog-ng start
```

Novamente, verifique se a porta 514 está aberta e se o processo do *syslog-ng* está "ouvindo" nela.

```
# netstat -anpd | grep 514
```

Firewall para o servidor de *log*

É extremamente importante configurarmos um firewall no servidor de *logs*. Nessa máquina, a única porta que deverá ficar "aberta" é a porta 514/UDP (porta padrão do *syslog-ng*) e as únicas máquinas que deverão ter acesso ao servidor são as máquinas clientes.

Antes de construirmos nosso script de firewall, devemos saber o que acontece quando enviamos um pacote UDP para uma porta numa das situações mostradas na **tabela 6.**

Veja na **listagem 2** uma sugestão de firewall para o servidor de *log*. Nesse script a variável `CL` corresponde ao endereço IP do cliente e a variável `SV` corresponde ao IP do servidor. É interessante deixar entrar no servidor pacotes *icmp* tipo 8 (*echo request*) e tipo 0 (*echo reply*) provenientes da máquina cliente, para testarmos a conexão entre cliente e servidor.

Observe que todos os pacotes UDP oriundos da máquina cliente com destino à porta 514/UDP do servidor são aceitos; porém os pacotes UDP enviados por qualquer outro *host* com destino à porta 514/UDP do servidor serão bloqueados pelo firewall com um `REJECT`. De acordo com a tabela verdade, quando enviamos um pacote UDP para uma porta em REJECT, um erro *icmp* tipo 3 é retornado, no caso do *iptables* um `icmp-port-unreachable`. Mas quando enviamos um pacote UDP para uma porta que está fechada, também é devolvido um erro *icmp3*. Essa situação poderá gerar um falso positivo quando alguém usar um programa de varredura, como o *nmap*, para verificar a situação da porta, pois o programa poderá acusar que a porta está fechada, quando na verdade, está filtrada com REJECT no firewall.

O *port scanner nmap* [3] poderá ser utilizado para testar seu novo firewall:

```
# nmap -sU -P0 -n -p 514 ip_do_servidor
```

A ferramenta *hping* [4] pode ser usada para ver as respostas enviadas pelo servidor:

```
# hping --udp ip_do_servidor -c 3 -p 514
```

Sem dúvida, o *syslog-ng* é uma alternativa flexível para a construção de um servidor de *logs*. A estrutura do seu arquivo de configuração é mais organizada que a do *syslog* e com mais opções. Os filtros são poderosos, permitindo até que se use a lógica booleana (*and*, *or* e *not*) para filtragem das mensagens. Mas, entre todas as vantagens, o fato de poder organizar de uma maneira simples um arquivo de *log* separado para cada cliente é o que o torna mais viável que o *syslog* padrão.

Leitura Adicional

A segurança de servidores Linux, e a manutenção correta de seus arquivos de *log* é parte dela, é um assunto extenso demais para ser coberto em apenas um artigo nesta revista. Para os interessados no assunto, recomendamos a leitura do livro *Building Secure Servers with Linux* **[5]**, escrito por Michael D. Bauer e publicado pela editora O'Reilly. O décimo capítulo trata justamente sobre o *syslog-ng*, e pode ser baixado gratuitamente, no formato PDF, no site da editora em **[6]**. Infelizmente, o livro ainda não tem uma versão em português. ■

Informações

[1] Página oficial do syslog-ng: http://www.balabit.com/downloads/syslog-ng/1.5

[2] BalaBit IT Security: http://www.balabit.hu/downloads/security/

[3] nmap: http://www.insecure.org/

[4] hping: http://wiki.hping.org/

[5] *Building Secure Servers with Linux*, Michael D. Bauer, Editora O'Reilly

[6] *System log management and monitoring*: http://www.oreilly.com/catalog/bssrvrlnx/chapter/ch10.pdf

[7] Manual de referência do *Syslog-ng*: http://www.balabit.com/products/syslog_ng/reference/book1.html

[8] *Intrusion Detection with Snort*, Jack Koziol, Sams Publishing: http://www.samspublishing.com/title/157870281X/

Sobre o autor

Flávio Ferreira da Cunha é um entusiasta de sistemas Unix. Formado em Redes de computadores pela FASP, possui também a cobiçada certificação LPIC2. Atualmente trabalha na empresa 4LINUX como consultor e instrutor.