MindWorks UK

TTP
TECHNOLOGY
TRANSFER
PARTNERS
Novell

Syslog(s) in SUSE Linux

Joe R. Doupnik

MindworksUK and Univ of Oxford

jrd@netlab1.oucs.ox.ac.uk

Mind Works IT Training & Event Specialists

JRD Networking Research Institute

What is syslog?

Apps write log messages to kernel logging facility

Syslog daemon listens for messages from kernel, writes them to log files or to a network mate

Three variants are used on Linux:

syslog	**original edition, on SLES 9 and prior**
syslog-ng	**“next generation”, on SLES/D 10/11**
rsyslog	**“remote”, on OpenSUSE 11 and later**

Snippets of documentation are used extensively in the slides which follow

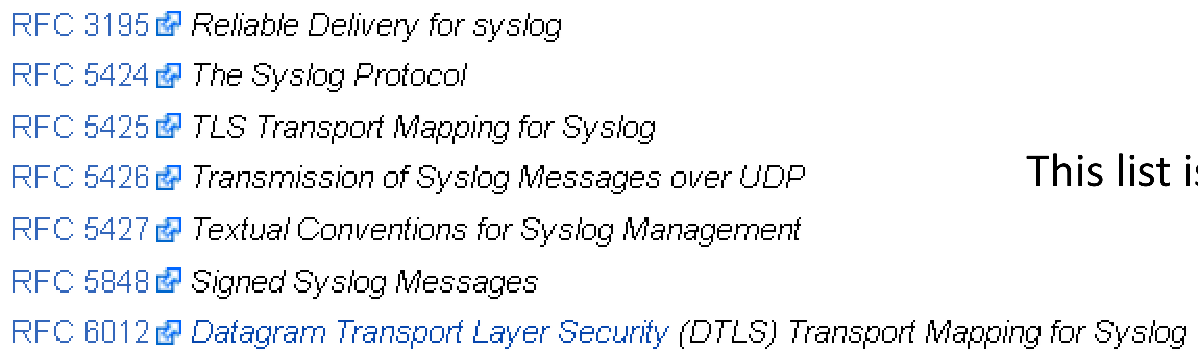
MindWorks UK

Whence syslog

Origin is from Eric Altman to support Sendmail, 80’s

Like many open source projects of that era, there was no published standard

BSD Unix adopted it, and it has become ubiquitous



RFC 3195 *Reliable Delivery for syslog*

RFC 5424 *The Syslog Protocol*

RFC 5425 *TLS Transport Mapping for Syslog*

RFC 5426 *Transmission of Syslog Messages over UDP*

RFC 5427 *Textual Conventions for Syslog Management*

RFC 5848 *Signed Syslog Messages*

RFC 6012 *Datagram Transport Layer Security (DTLS) Transport Mapping for Syslog*

This list is from wikipedia

RFC 5424 is by the author of rsyslog (done in 2009)

Standard syslog service is UDP port 514

Why bother?

Apps can write to their own log files, as many do. They can also interfere with each other.

We may want a central secure log repository on the net (think Sentinel log product suite)

Sending to a syslog daemon is faster than to files because traffic is through a kernel memory pipe

Syslog buffers messages and puts them into centrally placed log files chosen by the manager

A PHP application example

```
// JRD  add login failure to syslog (Apache error.log and messages)
    $when = date('D M d H:i:s o');
    openlog("[$when]", LOG_NDELAY | LOG_PERROR, LOG_AUTH);
    $who = $credentials['auth'];
    syslog(LOG_WARNING, "[auth] [client {$_SERVER['REMOTE_ADDR']}}] user
$who not found");
    closelog();
```

facility

priority

/var/log/apache2/error.log:

```
[Thu Feb 09 15:11:50 2012]: [auth] [client 129.67.101.23] user foobar not found
```

/var/log/warn:

```
Feb  9 15:11:50 netlab1 [Thu Feb 09 15:11:50 2012]: [auth] [client 129.67.101.23] user
foobar not found
```

/var/log/messages:

```
Feb  9 15:11:50 netlab1 [Thu Feb 09 15:11:50 2012]: [auth] [client 129.67.101.23] user
foobar not found
```

Example from PHP manual

```
// open syslog, include the process ID and also send
// the log to standard error, and use a user defined
// logging mechanism

openlog("myScriptLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// some code

if (authorized_client()) {
    // do something
} else {
    // unauthorized client!
    // log the attempt

    $access = date("Y/m/d H:i:s");

    syslog(LOG_WARNING, "Unauthorized client: $access {$_SERVER['REMOTE_AD
DR']} ({$_SERVER['HTTP_USER_AGENT']})");

}

closelog();
```

Yes, overly colourful, but we get the idea

Syslog’s ranking of things

***Facility*: a channel describing the message source**

LOG_AUTH	security/authorization messages (use LOG_AUTHPRIV instead in systems where that constant is defined)
LOG_AUTHPRIV	security/authorization messages (private)
LOG_CRON	clock daemon (cron and at)
LOG_DAEMON	other system daemons
LOG_KERN	kernel messages
LOG_LOCAL0 ... LOG_LOCAL7	local use, not available in Windows
LOG_LPR	line printer subsystem
LOG_MAIL	mail subsystem
LOG_NEWS	USENET news subsystem
LOG_SYSLOG	messages generated internally by syslogd
LOG_USER	generic user-level messages
LOG_UUCP	UUCP subsystem

Syslog's ranking of things

***Priority* level, helps select particular log file. Old name is severity level.**

LOG_EMERG	system is unusable
LOG_ALERT	action must be taken immediately
LOG_CRIT	critical conditions
LOG_ERR	error conditions
LOG_WARNING	warning conditions
LOG_NOTICE	normal, but significant, condition
LOG_INFO	informational message
LOG_DEBUG	debug-level message
LOG_NONE	exclude all, it is out of bounds

Syslog likes to choose items using syntax of *facility.priority* where *priority* often means "at or above" this level

The muddle: configuration

This is where complexity resides

- **/etc/syslog.conf**
- **/etc/syslog-ng/syslog-ng.conf**
- **/etc/rsyslog.conf**

syslogd was relatively simple, is now history

syslog-ng full of embellishments, used by SLES/D

rsyslog over embellished, can buffer network writes, can write to databases etc. Used in OpenSUSE

/etc/syslog.conf, old style

```
# all email-messages in one file
#
mail.*                  -/var/log/mail
mail.info               -/var/log/mail.info
mail.warning            -/var/log/mail.warn
mail.err                 /var/log/mail.err
#
# save the rest in one file
#
*.*;mail.none;news.none       -/var/log/messages
```

Format is *facility.level* destination_log

- means don't flush to disk for each write

; means join many commands into one

***Level* of none means exclude all**

Syslog-ng sources

internal()	Messages generated internally in syslog-ng
file()	Opens the specified file and reads messages
pipe(), fifo	Opens the specified named pipe and reads messages
program()	Opens the specified application and reads messages from its standard output
sun-stream(), sun-streams()	Reads from STREAMS device on Solaris systems
syslog()	Listens for incoming messages using the new IETF-standard syslog protocol
tcp(), tcp6()	Listens on the specified TCP port for incoming messages using the BSD-syslog protocol
udp(), udp6()	Listens on the specified UDP port
unix-dgram()	Listens on internal unix socket in SOCK_DGRAM mode
unix-stream()	Listen on internal unix socket in SOCK_STREAM mode

Syslog-ng, typical sources

```
source src {
        #
        # include internal syslog-ng messages
        # note: the internal() source is required!
        #
        internal();

        #
        # the default log socket for local logging:
        #
        unix-dgram("/dev/log");                 Unix socket (aka kernel buffers)

        #
        # uncomment to process log messages from network:
        #
        #udp(ip("0.0.0.0") port(514));            Network, UDP, not active here
};
```

UDP is the least reliable network transport, rsyslog addresses this issue

Syslog-ng destinations

file()	To the specified file
fifo(), pipe()	To the specified named pipe
program()	Forks specified program, send messages to its stdin input
sql()	To an SQL database
syslog()	To the specified remote host using the IETF-syslog protocol. Supports UDP, TCP, and TLS
tcp() and tcp6()	To a remote host using the BSD-syslog protocol
udp() and udp6()	To a remote host using the BSD-syslog protocol
unix-dgram()	To an internal unix socket in SOCK_DGRAM style (BSD).
unix-stream()	To an internal unix socket in SOCK_STREAM style
usertty()	To terminal of a user, if the user is logged in

Example for network:

```
destination my_logger { tcp( “1.2.3.4” port(514)  ); };
```

Output filenames can employ a date macro to have files automatically named by date (no logrotate involved)

Syslog-ng.conf, man page

It can connect *sources* and *destinations* using the *log* statement plus *filters* to select message types:

```
log  {  source S1; source S2;... filter F1; filter F2;...
        destination D1; destination D2;... };
```

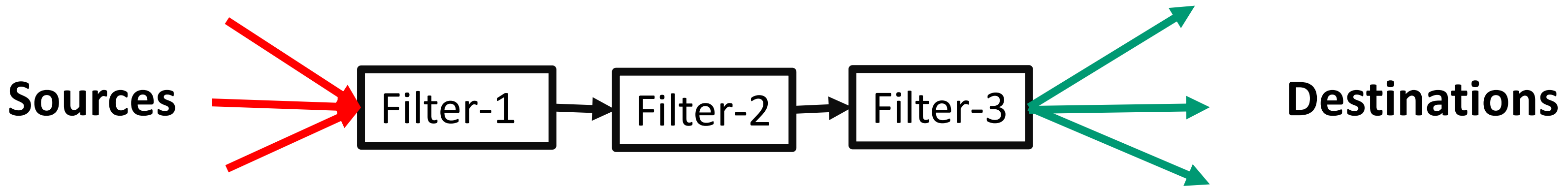
Note that filters are ANDed together

This does seem messy at first, but it works well

http://www.balabit.com/ is the source site, has docs

Another way of expressing this

**log { source S1; source S2;... filter F1; filter F2;...
destination D1; destination D2;... };**



Filter selections, in series

Syslog-ng snippets

Filters choose the messages to be processed

```
filter  f_mailinfo   { level(info)        and facility(mail); };
filter  f_mailwarn   { level(warn)      and facility(mail); };
filter  f_mailerr    { level(err, crit)    and facility(mail); };
filter  f_mail         { facility(mail); };

# Mail-messages in separate files:

destination mailinfo { file("/var/log/mail.info"); };
log { source(src); filter(f_mailinfo); destination(mailinfo); };

destination mailwarn { file("/var/log/mail.warn"); };
log { source(src); filter(f_mailwarn); destination(mailwarn); };
```

These verbose pairs do two things:

- more precise filtering than the "at or above" *level* style of old syslog
- the source() clause can select sources rather than "all" of old syslog

Syslog-ng filter examples

```
filter f_iptables   { facility(kern) and match("IN=") and match("OUT="); };
filter f_console   { level(warn) and facility(kern) and not filter(f_iptables)
                    or level(err) and not facility(authpriv); };
filter f_mailinfo    { level(info)      and facility(mail); };
filter f_mailwarn   { level(warn)   and facility(mail); };
filter f_mailerr   { level(err, crit)  and facility(mail); };
filter f_mail       { facility(mail); };
filter f_cron       { facility(cron); };
```

I added this to get rid of cron splatter

```
filter f_messages   { not facility(news, mail) and not filter(f_iptables) and not
     facility(cron); };
##filter f_messages   { not facility(news, mail) and not filter(f_iptables); };
filter f_warn       { level(warn, err, crit) and not filter(f_iptables); };
filter f_alert       { level(alert); };
```

These choose what to choose or ignore

Syslog-ng rewriting messages

Declaration:

```
rewrite <name_of_the_rule>
{subst("<string or regular expression to find>", "<replacement string>",
  value(<field name>) type() flags());};
```

Example:

```
rewrite r_rewrite_subst{subst("IP", "IP-Address", value("MESSAGE"),
   flags("global"));};
```

The docs discuss many ways patterns can be detected and rewriting done

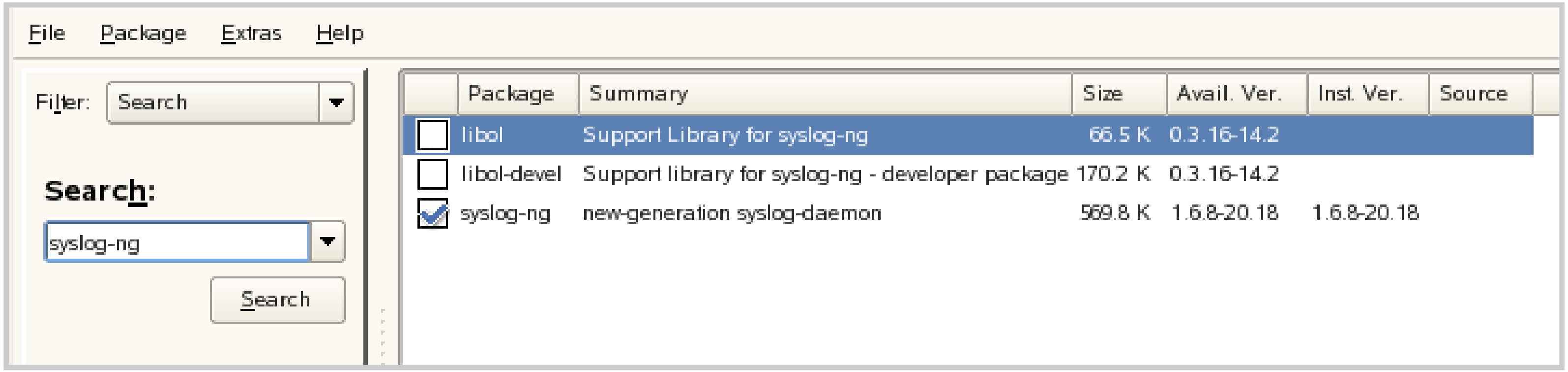
Purists might worry about "re-writing history" to cover tracks

Syslog-ng, a bit of taming

Reduce the log file clutter by delaying mindless syslog-stats messages for one full day, rather than the default of once per hour

```
# Global options.
#
options { long_hostnames(off); sync(0); perm(0640); stats(86400); };
##options { long_hostnames(off); sync(0); perm(0640); stats(3600); };
```

Syslog-ng extras, from YaST

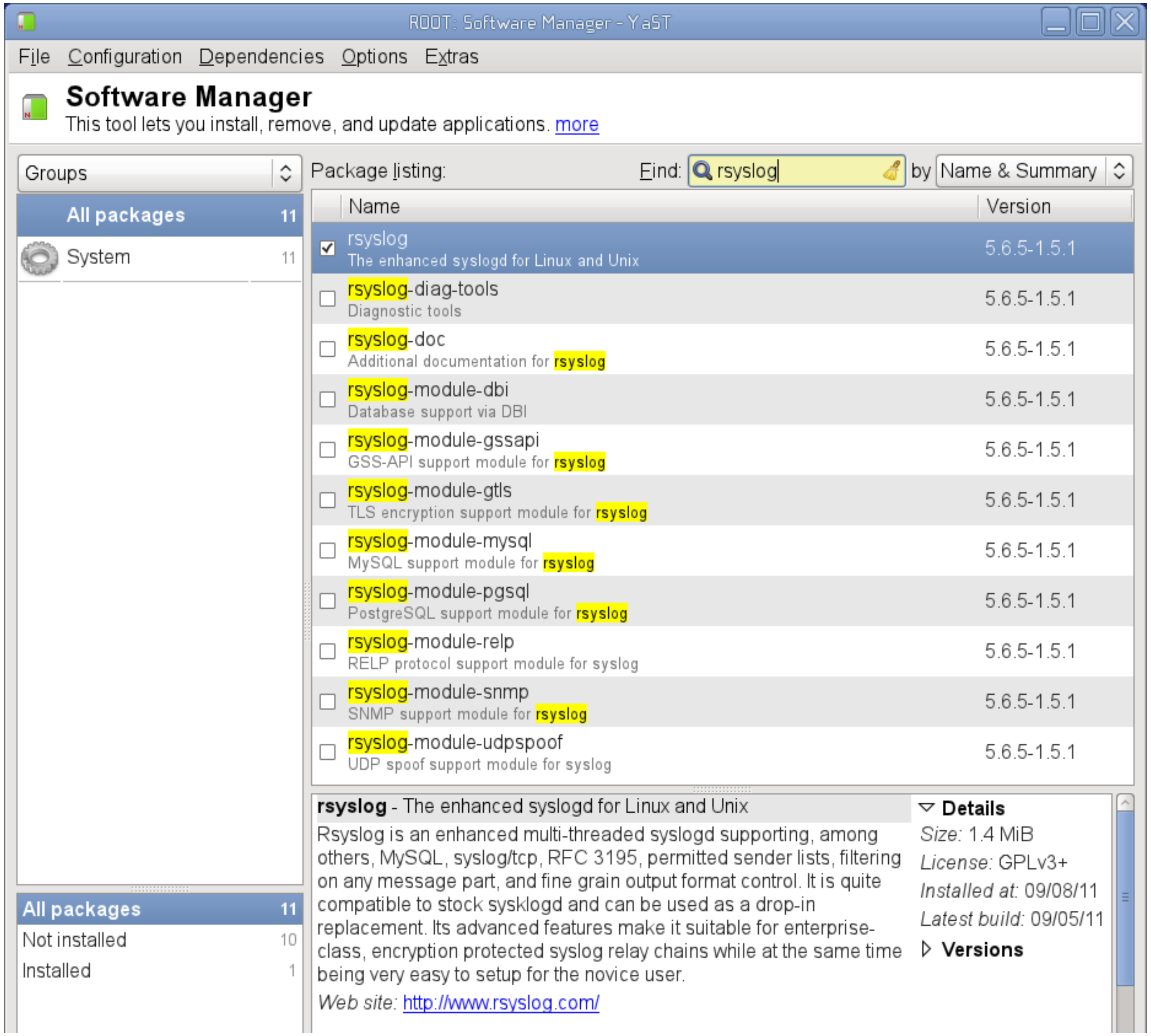


	Package	Summary	Size	Avail. Ver.	Inst. Ver.	Source
☐	libol	Support Library for syslog-ng	66.5 K	0.3.16-14.2		
☐	libol-devel	Support library for syslog-ng - developer package	170.2 K	0.3.16-14.2		
✓	syslog-ng	new-generation syslog-daemon	569.8 K	1.6.8-20.18	1.6.8-20.18	

Extras: Slim to none, and Slim just left town

Extra functionality is available in the commercial edition

Rsyslog, has many helpers


ROOT: Software Manager - YaST
File Configuration Dependencies Options Extras
Software Manager
This tool lets you install, remove, and update applications. more
Groups
All packages 11
System 11
Package listing:
Find: rsyslog
by Name & Summary
All packages 11
Not installed 10
Installed 1

	Name	Version
✓	rsyslog The enhanced syslogd for Linux and Unix	5.6.5-1.5.1
☐	rsyslog-diag-tools Diagnostic tools	5.6.5-1.5.1
☐	rsyslog-doc Additional documentation for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-dbi Database support via DBI	5.6.5-1.5.1
☐	rsyslog-module-gssapi GSS-API support module for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-gtls TLS encryption support module for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-mysql MySQL support module for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-pgsql PostgreSQL support module for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-relp RELP protocol support module for syslog	5.6.5-1.5.1
☐	rsyslog-module-snmp SNMP support module for rsyslog	5.6.5-1.5.1
☐	rsyslog-module-udpspoof UDP spoof support module for syslog	5.6.5-1.5.1

rsyslog - The enhanced syslogd for Linux and Unix

Rsyslog is an enhanced multi-threaded syslogd supporting, among others, MySQL, syslog/tcp, RFC 3195, permitted sender lists, filtering on any message part, and fine grain output format control. It is quite compatible to stock sysklogd and can be used as a drop-in replacement. Its advanced features make it suitable for enterprise-class, encryption protected syslog relay chains while at the same time being very easy to setup for the novice user.

Web site: http://www.rsyslog.com/

Details

Size: 1.4 MiB
License: GPLv3+
Installed at: 09/08/11
Latest build: 09/05/11

Versions

Rsyslog input modules

These three listen for network connections

imudp	**UDP syslog, send and forget, can lose data**

```
$ModLoad imudp
$UDPServerRun 514
```

514 is a UDP port number

imtcp	**Plain TCP syslog, loss if network connection vanishes**

```
$ModLoad imtcp
$InputTCPServerRun 514
```

imrelp	**RELP TCP protocol, prevents message loss by buffering**

```
$ModLoad imrelp
$InputRELPServerRun 2514
```

im* for input module

Rsyslog more input modules

imgssapi	Input plugin for plain TCP and GSS-enable syslog
immark	Support for mark messages (hourly chime spam)
imklog	Kernel logging

```
$ModLoad imklog
```

Please note that the klogd daemon is no longer used

imuxsock	Unix domain sockets, including the system log socket

```
$ModLoad  imuxsock                                   load the module (once only)
$InputUnixListenSocketHostName  jail1.example.net    change name in logs
$AddUnixListenSocket  /jail/1/dev/log                socket to listen upon
```

Rsyslog output modules

omsnmp	SNMP trap output module
omgssapi	Output module for GSS-enabled syslog
ommysql	Output module for MySQL
ompgsql	Output module for PostgreSQL
omrelp	Output module for the reliable RELP protocol (prevents message loss by buffering).

Examples -

```
*.*  :omrelp:server:port
*.*  :omrelp:192.168.0.1:2514
```

omlibdbi	Generic database output module (Firebird/Interbase, MS SQL, Sybase, SQLite, Ingres, Oracle, mSQL)

om* for output module

Logging to remote rsyslog

Everything to 192.0.2.1 port 10514 using TCP:

```
*.*  @@192.0.2.1:10514

# if you need to forward to other systems as well, just
# add additional config lines:

*.*  @@other-server.example.net:10514
```

Syntax is:

```
@host          uses UDP
@@host         uses TCP
:omrelp:host   uses buffered TCP
```

(:om… means output module, the reliable protocol in this case)

Logging <u>from</u> remote rsyslog

$ModLoad imtcp	**Load TCP input module, once only**
$InputTCPServerRun 10514	**Listen on TCP port 10514**

```
# do this in FRONT of the local/regular rules

if  $fromhost-ip startswith  '192.0.1.'  then  /var/log/network1.log   &~
```

& means "and do this" , ~ means discard
thus the message is not seen by lines below

```
if  $fromhost-ip startswith '192.0.2.'  then  /var/log/network2.log   & ~

# local/regular rules, like
*.*  /var/log/syslog.log
```

***facility.level*   destination log**

Disk spill buffer to queue traffic

```
$WorkDirectory /rsyslog/work          # default location for work (spool) files

$ActionQueueType LinkedList           # use asynchronous processing
$ActionQueueFileName dbq              # set file name, also enables disk mode
$ActionResumeRetryCount -1            # infinite retries on insert failure


# for PostgreSQL replace  :ommysql  by  :ompgsql  below
*.*      :ommysql:hostname,dbname,userid,password;
```

If message traffic is intense, and/or the destination is slow, then use a disk spill buffer as a queue

MindWorks UK

Forwarding to two destinations

```
$ModLoad imuxsock                      # local message reception

$WorkDirectory /rsyslog/work           # default location for work (spool) files

# start forwarding rule 1
$ActionQueueType LinkedList            # use asynchronous processing
$ActionQueueFileName srvrfwd1          # set file name, also enables disk mode
$ActionResumeRetryCount -1             # infinite retries on insert failure
$ActionQueueSaveOnShutdown on          # save in-memory data if rsyslog shuts down
*.*      @@server1:port
# end forwarding rule 1

# start forwarding rule 2
$ActionQueueType LinkedList
$ActionQueueFileName srvrfwd2          # a different filename for this queue
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
*.*      @@server2
# end forwarding rule 2
```

I dislike this construction. Actions are not bracketed to refer to a particular forward.

Message properties, can exploit

msg

the MSG part of the message (aka "the message" ;))

rawmsg

the message excactly as it was received from the socket. Should be useful for debugging.

hostname

hostname from the message

source

alias for HOSTNAME

hostname of the system the message was received from (in a relay chain, this is the system immediately in front of us and not necessarily the original sender). This is a DNS-resolved name, except if that is not possible or DNS resolution has been disabled.

fromhost-ip

The same as fromhost, but always as an IP address. Local inputs (like imklog) use 127.0.0.1 in this property.

syslogtag

TAG from the message

programname

the "static" part of the tag, as defined by BSD syslogd. For example, when TAG is "named[12345]", programname is "named".

and so on through a long list

Pluck string from contents

```
:msg, contains, "error"  /var/log/error.log  & ~
```

If a message contains string “error” then send it to error.log. Then discard it (do not pass msg to following rules)

Note the commas to separate key words

Rewriting messages can use regular expressions, and the rules can be very elaborate

Writing to MySQL

```
$ModLoad ommysql                    Load MySQL module, once only
```

Syntax is *facility.level* :ommysql:blah,blah as in this skeleton

```
*.*     :ommysql:database-server,database-name,database-
userid,database-password
```

Example command:

```
mail.*     :ommysql:127.0.0.1,syslog,syslogwriter,topsecret
```

Oh dear, a password is in plain sight.
But then we presume this is a secure server holding the database.

What we may choose to do

Review the syslog configuration file

Untangle the ways a single message might be logged to multiple files, simplify things

Optionally include exporting messages to a central log server

Silence noise makers to reduce rubbish entries

MindWorks UK

ind
orks
IT Training & Event Specialists

MindWorks Inc. Ltd
210 Burnley Road
Weir
Bacup
OL13 8QE UK

Telephone:	+44 (0) 170 687 1900
Fax:	+44 (0) 170 687 8203
Web:	www.mindworksuk.com
Email:	training@mindworksuk.com