

Next Generation System Logging: Syslog-NG

Flight Recording Box

Logging system events is a challenge for any administrator. And the weaknesses of the traditional syslog daemons are particularly apparent in larger networks – Syslog-NG is a worthy replacement.

BY CHRISTIAN SCHMITZ

Syslog allows admins to collect the logging data for the systems they manage in a uniform manner for a complete network. That makes the task of saving, analyzing and processing logfiles a lot easier. But what people expect of a logging system has certainly changed over the last few years – and the traditional syslog daemon simply cannot deliver. Syslog-NG [1] looks set to close this gap.

Logs are traditionally used to check on a system's health. Many admins didn't even bother looking at the logs unless they had to troubleshoot a system. But today, it is also a question of improving reliability, that is, using the system as an early-warning system to prevent things going too badly astray. The integrity of system messages is also more important than ever, as it allows admins to build up their defenses based on reliable data. Also, admins typically look for more flexibility with respect to configuration and networking.

Several projects have set out to achieve the goal of extending the traditional syslog daemon. One of the most widespread is Syslog-NG (Syslog Next Generation), which was released under the GPL. Many Linux distributions have already adopted Syslog-NG. Some of the other contenders are Reliable Syslog – in its first implementation, SDSC Secure Syslog [5], and Syslog Sign. The latter has not yet been released for use.



Problems with BSD Syslog

The traditional syslog daemon was introduced way back in September 1983 at the University of California (Berkeley). There were no design papers, and the software was sparsely documented. Some 18 years passed before BSD syslog was finally documented in RFC 3164 [7].

Syslog developed into a de-facto standard. The service is easy to configure using a central configuration file called *syslog.conf*. But there are a few good reasons not to be satisfied with syslogd's functionality:

Lack of authentication methods

Syslogd cannot distinguish multiple hosts. If the daemon is launched with the *-r* flag, it accepts messages on UDP port 514 no matter where they originated. This allows an attacker to flood the logging host with UDP packets, or transmit manipulated messages. Apart from simple firewall functionality, there

is no way to protect the logging host.

- **Messages in cleartext**

Syslog always uses cleartext to transmit messages across the wire. That makes it easy to sniff messages, and thus gain access to privileged information.

- **Inflexible configuration**

The Syslog configuration uses a rigid system with 20 different origins and eight priorities. This can be an obstacle on large networks or for server systems with multiple services.

- **Inconsistent use of origins and priorities**

For most applications, admins do not have an option to log messages below a specific origin. In some cases you can set an option when compiling the application, but there are very few runtime options available.

- **Does not quote the original source**

When a message is forwarded across multiple logging hosts, it is impossible to discover the original source of the message. Syslog does not store the FQDN

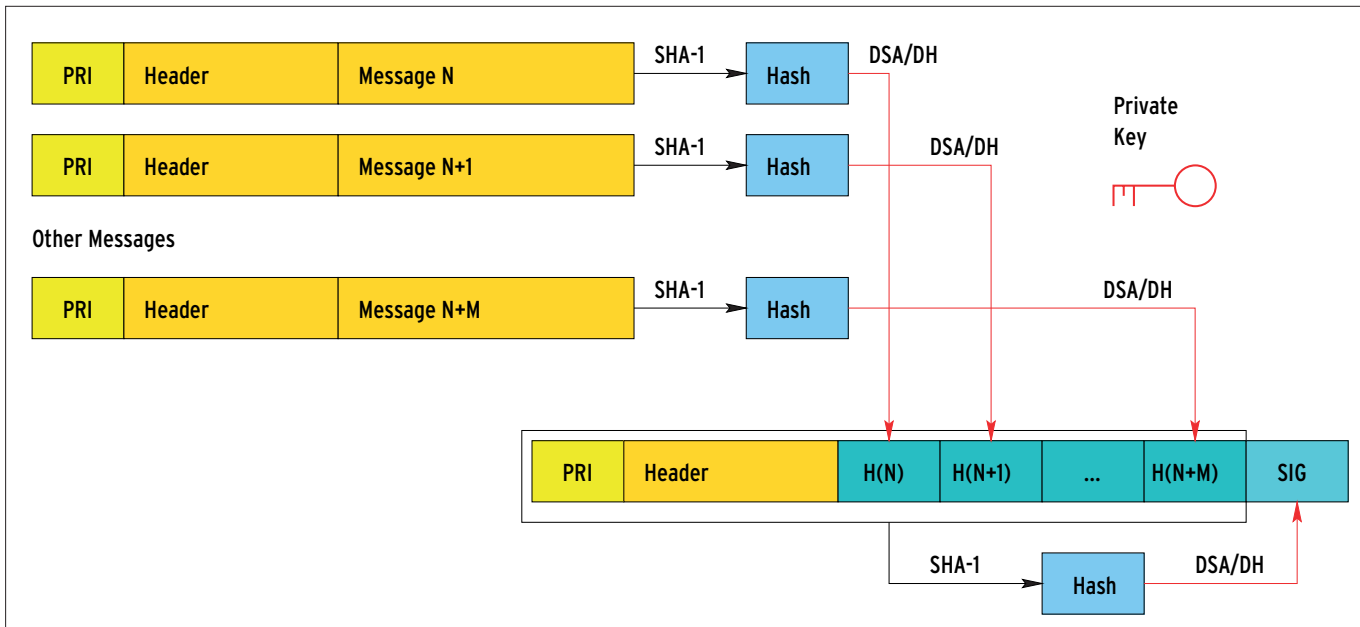


Figure 1: Syslog-Sign creates hashes for each message that is sent. It then transfers the digests for a group of messages, notifying the admin of the transmission and additionally signing the messages

(Fully Qualified Domain Name) of the host. Each host that forwards the message modifies the IP address logged with the message.

- **Uses connectionless UDP to transfer messages**

Syslogd can only use the connectionless User Datagram Protocol (UDP) to transfer messages. If a packet is dropped due

Syslog-Sign

Syslog-Sign is an extension of RFC 3164 [7] that retains downward compatibility to BSD Syslog. As the name suggests, this new development uses digital signatures to protect messages against manipulation and replay attacks. On boot-up, each host creates and asymmetric key pair that it uses to sign the message digests. The host then transmits a group message digests to the log server in a message of its own (see Figure 1).

This method is particularly useful to protect stored messages. As it retains downward compatibility to RFC 3164, Syslog-Sign uses UDP to transfer messages. Messages can go astray on the network. The messages themselves are not encrypted. An attacker sniffing them might gain access to privileged information.

Key verification is still not clear. An attacker might be able to distribute manipulated keys, as Syslog-Sign uses a normal message to transfer the public key. Work is in progress on a FreeBSD implementation [6], which is currently beta.

to network problems or the like, the message will never reach the recipient.

Syslog-NG

A few groups are working on removing these issues, and developing a state-of-the-art logging system (see the “Syslog-Sign” and “Reliable Syslog” boxes). Syslog-NG is the best of the bunch at present. This development builds up on the traditional Syslog Daemon adding new features without sacrificing compatibility to RFC 3164. Although it currently lacks features such as digital signatures and encryption, the developers do intend to add encryption to future versions. At present, stunnel [4] is required to handle encryption.

You can switch Syslog-NG from UDP to TCP to provide more reliability in message delivery. The tool uses port 514 by default in this case, although 514 is actually reserved for Rlogin. If you want to run both, you will need to re-configure the server.

Many distributions already include Syslog-NG. If you want to use the successor to syslog on SuSE Linux, add the following entry in `/etc/sysconfig/syslog` and then re-launch the daemon.

```
SYSLOG_DAEMON="syslog-ng"
```

The central configuration file `/etc/syslog-ng/syslog-ng.conf` is slightly more

complex than that of the traditional syslogd. Instead of original and priority pairs, it contains so-called logpaths that comprise the source, filter, destination.

There are eight different source drivers (see Table 1). The *internal* driver is mandatory. Syslog-NG uses this special source to transmit messages that concern the daemon itself.

Reliable Syslog

Reliable Syslog [8], which is specified in RFC 3195, uses BEEP (the Block Extensible Exchange Protocol) to transfer messages. This application layer protocol is based on TCP. It is connection-oriented, has authentication and verification mechanisms, and provides protection against replay and sniffing attacks.

Reliable Syslog uses two message formats: RAW mode is compatible to the RFC 3164 style syslog daemon, while COOKED mode uses XML format messages. COOKED messages also store additional attributes, such as IP addresses, FQDNs and device types. The 1023 byte restriction no longer applies – message length is now arbitrary. SDSC (San Diego Supercomputer Center) uses a Reliable Syslog implementation in its BSD-licensed Secure Syslog [5]. This version is downwardly compatible to BSD Syslog. It requires the Roadrunner BEEP libraries, OpenSSL, Glib, and Pkg-Config. In addition to the RAW and COOKED profiles, it also supports so-called security profiles.

Table 1: Source drivers

| Source | Description |
|-------------|--|
| internal | Driver for the daemon's own messages. Mandatory. |
| unix-stream | Opens the specified Unix socket in <i>SOCK_STREAM</i> mode and listens for messages. |
| unix-dgram | Opens the Unix socket in <i>SOCK_DGRAM</i> mode and receives messages through it. |
| file | Opens the specified file. |
| pipe, fifo | Opens the specified named pipe and reads messages. |
| udp | Listens for messages on the specified UDP port. |
| tcp | Listens for messages on the specified TCP port. |
| sun-stream | Opens the specified stream device (Solaris only). |

Sources

Warning: The *file* and *pipe* drivers are not to be confused with the *file* and *pipe* actions of *syslogd*. Syslog-NG uses them as sources from which the daemon reads messages, and not as goals to which messages are forwarded. The *file* driver takes over the role of *klogd*, for example, reading kernel messages from */proc/kmsg*.

Each of these drivers has one or more options, which can be specified in brackets following the driver name. For example, *tcp* and *udp* need to know the port number to listen on, and *file* needs the filename. Linux systems use the *unix-stream* driver locally; the driver uses a connection-oriented Unix domain socket. Each open connection requires a process of its own, a fact that an attacker might exploit to perform a DoS (Denial of Service) attack.

max-connections can help prevent this by specifying the maximum number of concurrent connections to the syslog service. This is set to 10 by default. The Reference Manual [2] contains a complete list of source driver options.

Filters and Destinations

Filters describe how Syslog-NG should handle the messages it receives from the various sources. They are one the strongest points of the new logging system. Admins can use filters to sort messages and pass them on to the appropriate targets.

Filter functions (see Table 2) can be linked using Boolean operators (*and*, *or*,

not and brackets). To apply a filter, the result of the operation must be *true*. Some filter functions can even handle regular expressions as options.

Destinations specify where, and by what means, messages are forwarded and processed. Just like for sources, a number of destination drivers are available – each of them can have different options. Table 3 provides an overview of the available drivers.

Syslog-NG only calls the program driver once and then keeps the driver running until the daemon receives a SIGHUP signal and terminates. This makes the driver very efficient. If Syslog-NG were to launch the external program for each incoming message, an attacker could launch multiple processes, which would be tantamount to a DoS attack against the system.

Syslog-NG also has a number of global options. For example, *chain_hostname* and *keep_hostname* specify how Syslog-NG handles hostnames, when forwarding messages across multiple logging hosts. This allows admins to discover where a message originated. The Reference Manual [2] contains a complete list of global options.

Configuration Example

To illustrate the structure of *syslog-ng.conf*, the following sections look at a simple configuration file section by section. If you are interested in how complex and expansive the file can be, take a look at [3].

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

Table 2: Filter Functions

| Filter | Description |
|-----------------|--|
| facility | Refers to messages that originate with the specified facility. |
| level, priority | Refers to messages with the specified priority. |
| program | Filters messages where the program name field contains the specified regular expression. |
| host | Filters messages where the hostname field contains the specified regular expression. |
| match | Applies the specified regular expression to the whole message. |
| filter | Calls another filter rule. |

Table 3: Destination Drivers

| Destination | Description |
|-------------|--|
| file | Writes the message to the specified file. |
| pipe, fifo | Passes the message to the specified named pipe. |
| unix-stream | Forwards the message to the <i>SOCK_STREAM</i> Unix socket |
| unix-dgram | Forwards the message to the <i>SOCK_DGRAM</i> Unix socket |
| udp | Sends the message to the specified UDP port. |
| tcp | Sends the message to the specified TCP port. |
| usertty | Sends the message to the console of the specified user, if the user is logged on. |
| program | Launches the specified program and sends the message to the programs standard input (Stdin). |

```
file("/proc/kmsg");
};
```

The source specified here is identified by its name *local* and comprises a number of local sources. The source drivers are *internal* (mandatory), the *unix-stream* driver, which uses the device *file /dev/log*, and a *file* driver that reads Kernel messages from */proc/kmsg*.

The next section creates a network source:

```
source remote {
    tcp(
        ip(192.168.0.24) port(3333)
        max-connections(10)
    );
};
```

This source is referred to as *remote*; its driver is *tcp*. Syslog-NG will now listen for messages on TCP port 3333. Even if the computer has multiple IP addresses, the server will only listen on 192.168.0.24. The *max-connections* option is set to 10; the computer will accept a maximum of ten concurrent connections to Syslog-NG. It is not always necessary to specify a unique source for remote messages. Many admins create a single source that handles all the drivers. Having said that, separate sources do provide a cleaner structure.

Selecting Messages

Now for the filters. Our first example handles messages whose log levels correspond to the values *warn*, *err*, *crit*:

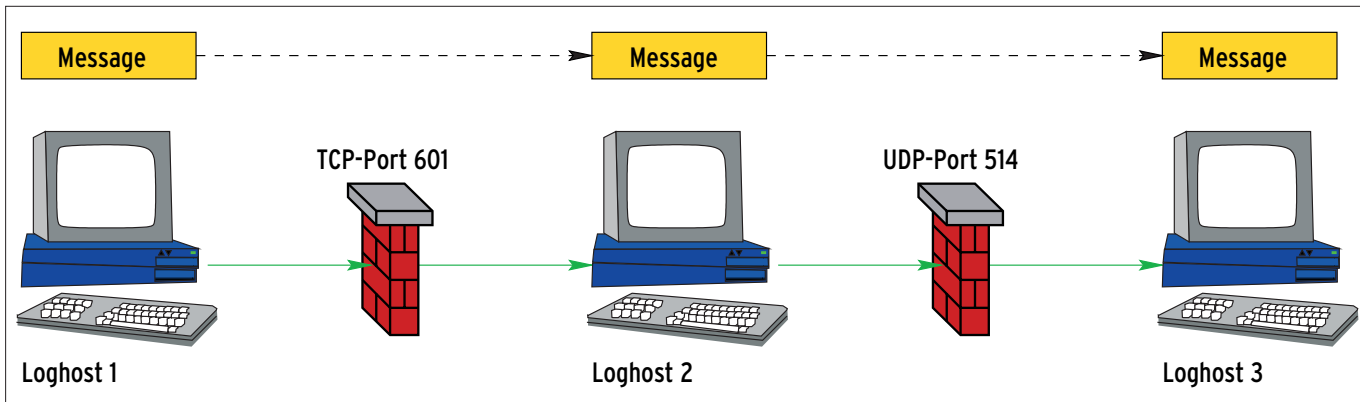


Figure 2: Depending on the configuration, Syslog-NG uses either TCP or UDP to forward messages across the network. This allows you to customize Syslog-NG for use in segmented environments with multiple firewalls

```
filter warning {
    level(warn, err, crit);
};
```

This simple rule, which applies priority-based filters, is quite common with the traditional syslogd. The major advantage of the Syslog-NG variant is its use of regular expressions. Various books on Perl contain useful introductions to this technique, as does the documentation for the Logsurfer [9] program. This automatic logfile analysis tool relies heavily on the use of regular expressions and provides a 50-page reference.

An extremely simple regular expressions searches for all messages that have anything to do with FTP. It is true if *ftp* occurs anywhere within a message:

```
filter ftp { match("ftp"); };
```

A filter rule that only allows critical messages to pass is also useful for output to TTY10. Console output should not be too verbose.

```
filter console {
    level(err) and not
    facility
    (authpriv)
    or level(warn) and
    facility
    (kern);
};
filter email {
    facility(mail);
};
```

The *console* filter uses functions, values and Boolean expressions. This

rule handles any error messages that do not originate from the *authpriv* facility, and all Kernel warnings. The *email* filter rule only allows messages from the mail system to pass.

On Target

All we need now, are a few destinations to send the messages to. The following configuration entries tell Syslog-NG to write messages to the logfile */var/log/mail* (destination name: *email*) or to the console */dev/tty10*.

```
destination email {
    file("/var/log/mail");
};
destination console {
    file("/dev/tty10");
};
```

A logpath describes the complete path from the source through the filter to the

target. It is a rule that comprises the name of a source, a filter and a destination entry. The first of the following *log* rules reads messages from the *local* source and sends entries that match the *console* filter rules to the destination *console*. The second log rule stores messages from the local mail system in */var/log/mail*.

```
log {
    source(local); filter(console);
    destination(console);
};
log {
    source(local); filter(email);
    destination(email);
};
```

If a host needs to forward messages received from network sources to another loghost, you simply configure an appropriate destination for Syslog-NG. It does not matter where the message originates, not even if the source uses UDP and a different port.

```
destination loghost {
    udp( ip(172.16.0.
    33)port(514) );
};
log {
    source(remote);
    filter(ftp);
    destination(loghost);
};
```

The important thing is that the log server, *loghost*, has a source that listens to TCP port 514 defined for the IP

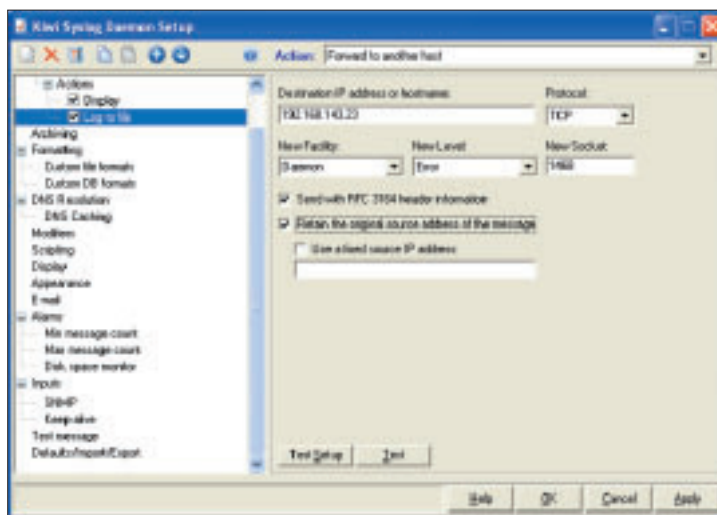


Figure 3: The Syslog Daemon from Kiwi Enterprises allows admins to include Windows computers in their logging setup

address 172.16.0.33 in its configuration file. This simple address conversion method can be extremely useful (see Figure 2).

Global Options

To complete the configuration file, we now need the global options, which are listed at the start of the file. Some options specify how Syslog-NG will handle hostnames in messages, when forwarding messages to other logging hosts. Enabling *keep_hostname* tells Syslog-NG to keep an existing name.

If *keep_hostname* is disabled, *chain_hostnames* (alias: *long_hostname*) decides what to do with the name. Without chaining, the log-server will insert its own hostname, with chaining it will add its name to the existing hostname. This allows admins to trace the path of a message back to its source.

```
options {
  keep_hostname(no);
```

```
  chain_hostnames(yes);
  sync(0);
};
```

The *sync* option specifies the number of lines Syslog-NG will cache, before writing them to a file. A larger value improves performance, but it does increase the risk of losing messages if a system crashes.

Listing 1 contains the complete configuration file. This simple example illustrates how much more advanced Syslog-NG is with respect to flexibility and scalability, in comparison to its predecessor.

Almost Perfect

Filter functions allow admins to customize logging to reflect more complex network infrastructures. The configuration is clear despite its flexibility. You can even compensate for the lack of encryption and authentication facilities – if you are prepared to put in some extra

work [4]. If you require integrated encryption, authentication and anti-replay facilities, you should take a look at SDSC Secure Syslog. However, this server does require more management.

At the time of writing, none of the syslog developments we looked at fulfill all of our requirements. Having said that, Syslog-NG comes closest to doing so, as you can customize it to support your existing infrastructure with relatively little effort. ■

Syslog and Windows

Many networks comprise both Linux and Windows PCs. Under ideal circumstances, a central logging solution should handle all of these systems. However, some extra software is required to allow Microsoft computers to send and receive syslog messages.

The Syslog Daemon by Kiwi Enterprises <http://www.kiwisyslog.com> is one example of a program that provides this service. It supports Windows 9x, NT, 2000, and XP. Besides the freeware version, there is also a commercial version with added functionality.

The free version is normally adequate for simply adding Windows machines to a Linux syslog environment. It is GUI configurable and can be run as a loghost and/or client. Just like Syslog-NG, the Kiwi software can use TCP or UDP to send messages. It also has a few nice, albeit superfluous, bells and whistles such as colored bar charts and other statistical displays.

Listing 1: Syslog-NG Configuration

```
01 # Global Option
02 options { keep_hostname(no); chain_hostnames(yes); sync(0); };
03
04 # Sources
05 source local { internal(); unix-stream("/dev/log");
06   file("/proc/kmsg"); };
06 source remote { tcp(ip 192.168.0.24) port(3333) max-connections(10);
07   };
07
08 # Filters
09 filter warning { level(warn, err, crit); };
10 filter email { facility(mail); };
11 filter ftp { match("ftp"); };
12 filter console {
13   level(err) and not facility(authpriv)
14   or level(warn) and facility(kern);
15 };
16
17 # Send critical messages to TTY10
18 destination console { file("/dev/tty10"); };
19 log { source(local); filter(console); destination(console); };
20
21 # Write mail messages to a file
22 destination email { file("/var/log/mail"); };
23 log { source(local); filter(email); destination(email); };
24
25 # Forward messages to another network
26 destination loghost { udp(ip(172.16.0.33) port(514)); };
27 log { source(remote); filter(ftp); destination(loghost); };
```

INFO

- [1] Syslog-NG homepage: http://www.balabit.com/products/syslog_ng/
- [2] Syslog-NG manual: http://www.balabit.com/products/syslog_ng/reference/book1.html
- [3] Sample configuration: <http://www.campin.net/syslog-ng.conf>
- [4] Syslog-NG encryption howto: <http://venus.ece.ndsu.nodak.edu/~jezerr/linux/secure-remote-logging.html>
- [5] Secure Syslog by SDSC: <http://security.sdsc.edu/software/sdsc-syslog/>
- [6] Syslog-sec: <http://sf.net/projects/syslog-sec/>
- [7] RFC 3164, "The BSD Syslog Protocol": <http://www.ietf.org/rfc/rfc3164.txt>
- [8] RFC 3195, "Reliable Delivery for Syslog": <http://www.ietf.org/rfc/rfc3195.txt>
- [9] Logsurfer: <http://www.cert.dfn.de/eng/logsurfer/>