Logging with syslog-ng, Part Two

By Forrest Hoffman

The previous "Extreme Linux" column (available online at http://www.linux-mag.com/2005-11/extreme_01.html) described the use of *syslogd* and *syslog-ng* for managing system logs within *Linux* clusters and networks of workstations. While system logs are often considered to be of little value, they can explain configuration problems, foretell of impending hardware failures, and inform system administrators of mischievous user or cracker behavior. However, to be of much utility, logs from a large cluster of compute nodes or from a collection of widely dispersed clusters in a grid must be managed in an efficient and secure manner.

The syntax for configuring *syslogd* was presented last time, along with instructions for setting up centralized log collection on a single (master) node. The previous column also included instructions for building, installing, and configuring *syslog-ng* on an out-of-the-box *Fedora Core 4* distribution.

The example discussed last time used *syslog-ng* running on a master node, receiving log entries from client/compute nodes running regular, old *syslogd*. [You can find the configuration files for that example and for the examples shown this month at http://www.linux-mag.com/downloads/2006-01/extreme.]

The example also demonstrated how *syslog-ng* on a master cluster node can forward a copy of these logs to a central grid host (a multi-site or enterprise-level log host also running *syslog-ng*) over the Internet. Forwarding was accomplished by adding a new *destination* to the *syslog-ng* configuration file as follows (the entry should appear on a single line, but is wrapped here due to the narrow column width):

```
destination gridhost
  { tcp("123.45.67.89") port(514)); };
```

This `destination` section sends log entries to the machine at `123.45.67.89` via *TCP* to port `514`.

However, there are a number of problems with this scheme. First, TCP port 514 was originally used for *rsh* (something no site should ever run), so port 514 is probably firewalled by institutional policy. To make the `destination` rule work, a firewall exception at the remote (central grid host) site is needed. Second, the log messages traverse the Internet in the clear, so anyone can read them and learn all kinds of things about your machines, users, and system configurations.

These problems can all be overcome by forwarding messages over an encrypted connection between two machines residing on two different site networks. Encrypted channels can be established either by using *ssh* to create a reverse tunnel from the central grid host, or by using *stunnel* on both nodes to create a tunnel between the two systems using SSL (secure sockets layer) certificates. Both of these methods, along with their advantages and disadvantages, are described in detail in this month's column.

Log Forwarding via ssh

Configuring *ssh* to establish a reverse tunnel from the central grid machine back to the master cluster node is relatively easy. It requires only that port 22 be accessible through the firewall at the cluster's site — something that's likely already working if remote users can login to the cluster.

To make this connection work without human intervention, *ssh* must be configured so that the *root* user on the central grid host can login as the *root* user on the master cluster node without a password. This is accomplished by generating a key on the central grid host with an empty passphrase, as shown in *Figure One*.

Press only the Return key in response to the three prompts from *ssh-keygen*. Now, place the public RSA key from the grid host (contained in the file */root/.ssh/id_rsa.pub*) into a file called */root/.ssh/authorized_keys* on the master cluster node. On the master cluster node, be sure to set the modes of */root.ssh* to `0700` and */root/.ssh/authorized_keys* to `0600`.

Next, test that *root* on the grid host can execute a command on the master cluster node. You can run a simple command, like *w*, as shown in *Figure Two*.

This test is important not only to demonstrate that the keys work, but also because the RSA fingerprint must be accepted (by typing `yes`) manually the first time. When the grid host is configured to establish a reverse tunnel automatically, responding to such a prompt is not possible.

Next, the *syslog-ng* configuration files on both the grid host and the cluster master node should be modified to use the domain name server to look up hostnames and to use fully-qualified domain names in hostnames. This assures that log entries contain the complete hostname and domain name, so that their origins can be determined. In the *options* section of each file, set the following:

```
# Old
#use_dns (no);
#use_fqdn (no);

# New
use_dns (yes);
use_fqdn (yes);
```

Next, add the following line to the *source* `s_sys` section of the configuration file on the central grid host:

```
# New
tcp(port(5140) max-connections(100));
```

This causes *syslog-ng* on the grid host to listen on TCP port `5140` for logs. Since the grid host may be used to receive logs from a large number of remote clusters, the maximum number of connections is raised to `100` (from the usual default of 10). Then, on the master cluster node, add a new *destination* section…

```
# New
destination d_gridlogport
   { tcp("127.0.0.1" port(5140)); };
```

… and a new log section:

```
# New
log { source(s_sys);
   destination(d_gridlogport); };
```

This causes all logs on the master cluster node to be sent to itself (*localhost*) via TCP port `5140.` Once the reverse tunnel is established, the effect is that anything destined for TCP port 5140 on the master node is encrypted and sent via *ssh* to the central grid host where it's then forwarded to TCP port 5140. Once these configuration file changes are in place, *syslog-ng* should be restarted on both the grid host and the master node (using `service syslog-ng restart`).

Finally, the reverse tunnel needs to be established from the grid host. Since this should happen automatically and since it should be re-established if it ever gets dropped, it's best to have *init* handle it. This is easily accomplished by adding the following line (wrapped here intentionally to fit the column) to */etc/inittab* on the central grid host:

```
log1:35:respawn:/usr/bin/ssh -nNTx -R
5140:gridhost:5140
master.someschool.edu > /dev/null 2>&1
```

FIGURE ONE: Generating an *RSA* key on the central grid host

```
[root@gridhost root]# ssh-keygen -trsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
84:b1:c7:4d:94:67:d4:3d:64:05:86:ff:3c:b3:5c:24
  root@gridhost
```

Whenever the grid host is in run states 3 or 5, *init* executes the `ssh` command, respawning the command if it ever dies. The options on `ssh` redirect *stdin* from */dev/null* (`-n`); do not execute a remote command (`-N`); disable pseudo-tty allocation (`-T`); disable X11 forwarding (`-x`); and specify that port `5140` on `master.someschool.edu` should be forwarded to port `5140` on `gridhost` (`-R`).

Once this line is added to */etc/inittab* on the central grid host, send a HUP signal to *init* (using `killall -HUP init`) to start everything working. Log entries from `master.someschool.edu` should begin appearing in the log files of `gridhost`. It may be convenient to route these remotely generated log entries to one or more separate files so that they're not interspersed with the log entries of `gridhost` itself.

While this setup is relatively secure (since the correct key must be presented to the master cluster node and logs are sent encrypted over the Internet), it means that anyone who can become *root* on the central grid host also has complete control of the cluster master node (and probably the entire cluster). Moreover, if the central grid host is ever compromised, then the cluster master node (and probably the entire cluster) should be considered compromised as well.

One way of restricting privileges on the cluster and reducing the impact of a security compromise is to have *syslog-ng* run as a non-*root* user on the master node. Nevertheless, some additional security risk is incurred as a result of trusting the central grid host.

FIGURE TWO: Testing an *ssh* configuration

```
[root@gridhost root]# ssh master.someschool.edu w
The authenticity of host 'master.someschool.edu (89.67.45.123)'
can't be established.
RSA key fingerprint is
66:65:7e:00:e9:ab:35:21:89:37:17:9d:06:64:d6:30.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master.someschool.edu, 89.67.45.123'
(RSA) to the list of known hosts.
 20:50:56 up 1 day,  3:54,  1 user,  load average:
   0.03, 0.03, 0.00
USER   TTY    FROM             LOGIN@   IDLE    JCPU   PCPU  WHAT
bob    pts/1 schmoe.someschool 17:21   19.00s  1.15s  1.15s -bash
```

Log Forwarding via stunnel

An alternative mechanism for sending encrypted system logs to a central grid host over the Internet is to employ *stunnel*, the universal SSL tunnel program. Operating

much like the reverse tunnel created with *ssh* described above, *stunnel* accepts local connections and encrypts and forwards data to a remote host. The *stunnel* wrapper is frequently used to provide SSL-encrypted transport for non-SSL-aware daemons.

In this case, you want to establish a persistent connection between specific ports on two hosts. The easiest and most secure way to set this up is to create SSL certificates on each host, then provide the public portions of these certificates to the other host. On the master cluster node (again assuming it is running Fedora Core 4), an appropriate certificate can be created using the commands in *Figure Three*.

Then using an editor, remove the private section from *syslog-ng-client.pem.4server*. This file now contains the public portion of the certificate needed by the central grid host to verify the identity of the master node in establishing the encrypted tunnel.

On the central grid host, a similar certificate can be created, as shown in *Figure Four*.

Again, with an editor, remove the private section from *syslog-ng-server.pem.4clients*. This file now contains the public portion of the certificate needed by the cluster master node to verify the identity of the grid host in establishing the tunnel.

The *syslog-ng-client.pem.4server* file from the master node should be appended to a file called */etc/stunnel/syslog-ng-client.pem* on the central grid host. This file contains all the public certificates from all of the clients that have *syslog-ng stunnel* connections. Similarly the *syslog-ng-server.pem.4clients* file from the grid host should be copied to a file called */etc/stunnel/syslog-ng-server.pem* on all master cluster nodes.

Now that the certificates are all in place, a configuration file for *stunnel* is needed. This file should be called */etc/stunnel/stunnel.conf* on both the master node and the central grid host. On the grid host, the file should contain the following:

```
cert = /etc/stunnel/syslog-ng-server.pem
CAfile = /etc/stunnel/syslog-ng-client.pem
verify = 3
[5140]
accept = gridhost:5140
connect = 127.0.0.1:514
```

The `cert` entry points *stunnel* to the server's PEM-encoded certificate (which contains both the private and public sections). The `CAfile` entry points to the file containing the public portion of the master node's certificate. The `verify` entry is set to 3, which tells *stunnel* to verify the client's identity using the locally installed certificate (contained in the `CAfile` file). Next, a configuration section labeled `[5140]` contains two entries: `accept` and `connect`.

The `accept` entry tells *stunnel* to accept connections from its external IP address on port `5140`. The `connect` entry tells *stunnel* to connect to itself (*localhost*) on port `514`. Therefore, data coming in from the Internet on port 5140 will be decrypted and sent to port 514. Make sure that *rsh* is not enabled or this could cause problems.

FIGURE THREE: Creating an SSL certificate on the master node

```
[root@master ~]# cd /etc/pki/tls/certs
[root@master certs]# make syslog-ng-client.pem
[root@master certs]# mv syslog-ng-client.pem
/etc/stunnel/
[root@master stunnel]# cp syslog-ng-client.pem
syslog-ng-client.pem.4server
```

FIGURE FOUR: Creating an SSL certificate on the central grid host

```
[root@gridhost ~]# cd /etc/pki/tls/certs
[root@gridhost certs]# make syslog-ng-server.pem
[root@gridhost certs]# mv syslog-ng-server.pem
/etc/stunnel/
[root@gridhost stunnel]# cp syslog-ng-server.pem
syslog-ng-server.pem.4clients
```

On the cluster master node, */etc/stunnel/stunnel.conf* should contain the following:

```
client = yes
cert = /etc/stunnel/syslog-ng-client.pem
CAfile = /etc/stunnel/syslog-ng-server.pem
verify = 3
[5140]
accept = 127.0.0.1:514
connect = gridhost.centralgrid.edu:5140
```

The `client` entry tells *stunnel* on the master node that it is a client. Again, the `cert` and `CAfile` entries point to the PEM-encoded (public and private) certificate of the master node and the public certificate of the grid host, respectively. The `verify` entry assures that the public certificate of the grid host is verified. In the `[5140]` section, the `accept` entry causes *stunnel* to listen for traffic on port `514` on *localhost*, and the `connect` entry tells *stunnel* to encrypt that traffic and forward it to *gridhost* on port 5140.

Now that the encrypted tunnel is configured, *syslog-ng* on each host needs to be reconfigured to take advantage of this tunnel. The `use_dns` and `use_fqdn` options should be set to `yes` as described above on both hosts. On the central grid host, comment out (or remove) the entry used for *ssh* and add an entry for port 514 as follows in the *source* `s_sys` section of the *syslog-ng.conf* file.

See Extreme, *pg. 63*

Extreme, *from pg. 52*

```
# New
#tcp(port(5140) max-connections(100));
tcp(ip(127.0.0.1) port(514));
```

This causes *syslog-ng* to listen to itself (*localhost*) on TCP port `514`. On the cluster master node (the client in this case), remove or comment out the the `destination` and `log` sections used for *ssh*, and add new ones for *stunnel* forwarding as follows:

```
# New
#destination d_gridlogport
   { tcp("127.0.0.1" port(5140)); };
destination d_stunnel
   { tcp("127.0.0.1" port(514)); };
…
# New
#log { source(s_sys);
   destination(d_gridlogport); };
log { source(s_sys);
   destination(d_stunnel); };
```

These new sections will cause *syslog-ng* to send all its logs to itself (localhost) on port 514, in addition to all the other destinations (files) where they are sent.

With this configuration, *syslog-ng* on the master node sends logs to *localhost* on port 514 where *stunnel* receives them, encrypts them, and sends them to the central grid host via TCP on port 5140. Then, *stunnel* on the grid host receives the encrypted logs on port 5140, decrypts them, and sends them to *localhost* on port 514, where *syslog-ng* receives them and processes them according to the local configuration.

Finally, *stunnel* should be started on both systems, if it's not already in use. In Fedora Core 4, it appears that no *init*/startup files are provided for *stunnel*. Such a file could be created so that *stunnel* is started whenever the system boots, or the command `/usr/sbin/stunnel` could be added to */etc/rc.d/ rc.local*, which gets executed at boot time.

Once *stunnel* is running on both systems and *syslog-ng* has been restarted (using `service syslog-ng restart`) on both systems, logs should begin flowing from the master node to the grid host.

Using the *stunnel* method of forwarding, the logs are sent encrypted across the Internet without requiring system administrators at either site to have *root* access (or even non-root access) to the other system. However, since traffic flows between the two sites on port 5140, this port cannot be blocked by the firewall at either site or by the firewall configuration on either the master node or the grid host. Firewall rules should be (re)written to accept TCP traffic on port 5140.

At institutions where the bureaucratic barriers to getting a firewall exception are significant, it may be more expedient to employ the *ssh* method, assuming port 22 is already open, until a thorough analysis of risks and benefits is complete. The best method depends on the security policy needed for individual systems and site networks.

Send Forrest Hoffman email at forrest@climate.ornl.gov.

Advertisers' Index

The Advertisers' Index lists each company's Web address and advertisement page. To advertise in Linux Magazine, please contact adsales@linux-mag.com for a media kit containing an editorial schedule, rate card, and ad close dates.

Linux Magazine (ISSN 1536-4674) is published monthly by InfoStrada LLC at 330 Townsend St. Suite 112, San Francisco, CA 94107. The U.S. subscription rate is $29.95 for 12 issues. In Canada and Mexico, a one-year subscription is $59.95 US. In all other countries, the annual rate is $89.95 US. Non-US subscriptions must be pre-paid in US funds drawn on a US bank. Periodicals Postage Paid at San Francisco, CA and at additional mailing offices.

POSTMASTER: Send address changes to Linux Magazine, P.O. Box 55731, Boulder, CO 80323-5731.

Article submissions and letters should be e-mailed to editors@linux-mag.com. Linux Magazine reserves the right to edit all submissions and assumes no responsibility for unsolicited material. Subscription requests should be e-mailed to linuxmag@neodata.com or visit our Web site at www.linuxmagazine.com.