18.6. Remote Logging: Server

The **rsyslog** server can listen for three connection types. Verify that the **module** configuration is similar to:

- **UDP**
  ```
  $ModLoad imudp
  $UDPServerRun 514
  ```
- **TCP**
  ```
  $ModLoad imtcp
  $TCPServerRun 514
  ```
- **RELP transport**
  ```
  $ModLoad imrelp
  $RELPServerRun 601
  ```

Each of the module definitions requires an option for the port number that will be listened to. Be careful with **firewall** and **SELinux** settings. The output in the server's system log will now show messages from the client.

Having all the messages in one log file may not be the best. You can use property-based filters to separate the information. Here is an example:

```
#### RULES ####
```

```
#### RULES ####
:fromhost, isequal, "kvm.onapinglake.com"  /var/log/messages-kvm
& ~
```

This rule is placed at the top of the rules section, so it is the first rule processed.

The first line checks the client **`hostname`** for a match; if positive, it writes a log message, if not, it continues processing.

The second line of the rule stops further processing if the previous line was positive, if not, the message will be processed again by the rules following and the log message will be written twice.

18.2. Learning Objectives

By the end of this session, you should be able to:

- **Explore the configuration options for system logging.**

18.3. rsyslog

The **r**ocket-fast **sys**tem for **log** processing (**rsyslog**) is a system logging application that accepts messages from various sources, separating the process that generated the log from the process that records the event message.

The default configuration in most **Linux** distributions has **rsyslog** accepting all the system-related messages and recording events in local files, usually in the `/var/log` directory.

The configuration files for **rsyslog** are `/etc/syslog.conf` and the `/etc/rsylog.d/` directory.

In summary, **ryslog** offers high performance, excellent security features, accepts many input types and outputs to a variety of targets, provides filters for any message type and has configurable output formats.

18.4.a. rsyslog Modules

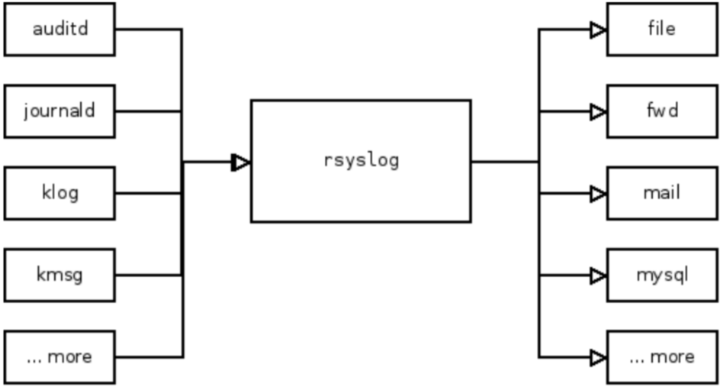

Figure 18.1: rsyslog

THE LINUX FOUNDATION

18.4.b. rsyslog Modules (Cont'd)

rsyslog uses modules for input and output. An example of some input module configurations are:

```
$ModLoad imuxsock # provides support for local system logging

$ModLoad imjournal # provides access to the systemd journal

$ModLoad imklog   # provides kernel logging support
```

The default action for output is to write to one or more files. Standard output configuration rules have the following format:

```
<facility.priority>       <action>
```

See the **man** page on `rsyslog.conf` for additional details. Some standard output rule examples are:

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none         /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                       /var/log/secure

# Log all the mail messages in one place.
# Log all the mail messages in one place.
mail.*                                                -/var/log/maillog
```

18.5. Remote Logging: Client

There are different options for sending system log messages to a remote host. The option of the **UDP** transport is the oldest and has the inherent challenges associated with **UDP**: an unreliable transmission.

The **TCP** protocol option resolves the packet delivery reliability issue. These two protocols require the receiver, the **syslog** server, to be ready to receive packets or else the **syslog** information may be lost.

The third option (**RELP**) can be configured to prevent loss of messages in the case of the server being unavailable. **RELP** has options of temporary disk queue, saving outstanding messages on shutdown, and retry counters that can be engaged. This makes the **RELP** transport the most reliable and robust of the communication methods.

There is a **rsyslog** module required for these transports, **UDP** and **TCP** are usually built-in. However, for the **client** to send messages via **RELP** an additional module, **omrelp.so**, must be installed and configured. Look in your distribution for additional packages: **rsyslog-relp** and maybe **rsyslog-doc**.

To enable the **relp** module for the client, add:

```
$LoadModule omrelp
```

to the modules section of `/etc/rsyslog.conf`.

The client can send log messages three ways:

The client can send log messages three ways:

- **UDP:** `*.* @192.168.0.1`
- **TCP:** `*.* @@192.168.0.1`
- **RELP transport:** `*.* :omrelp:192.168.0.1:601`