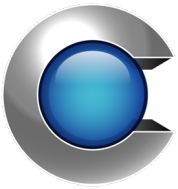

Manual

Cryptify Log Module

CAB-18:006, Rev C. 2021-03-01

Table of Contents

Scope

This document describes how to install and configure the Cryptify Log Module (CLM). It assumes the reader is familiar with the Ubuntu, RedHat Enterprise Linux or SUSE Linux Enterprise Server operating systems.

Prerequisites

The CLM requires a server running 64-bit versions of Ubuntu 16.04 or 20.04 LTS, RedHat Enterprise Linux 7.6 or SUSE Linux Enterprise Server 12 SP1.

The CLM should be installed on each server running Cryptify software that is to be monitored.

Basic concepts

The basic operation of the CLM is to gather data from a set of sources and transfer the data elsewhere, to a data sink.

Supported sources include:

- CRS traces, usually viewed using the `crs-cli trace` family of commands.
- Plain text log files, such as the CRS log (normally written to `/var/log/crs/crs.log`).

Supported sinks include:

- The system log, as viewable via `journalctl`.
- A plain text file.

Administrative tasks

Installation

To install the CLM for the first time, simply use the package installer for your operating system:

```
Ubuntu$ sudo dpkg -i clm-3.x.y-amd64-ubuntu.deb
RHEL$ sudo rpm -i clm-3.x.y-x86_64-rhel.rpm
SLES$ sudo rpm -i clm-3.x.y-x86_64-suse.rpm
```

where the `.deb/.rpm` installation package is provided by Cryptify.

When installing the CLM, a new user – `clmuser` – is added to the system and used to run the service.

After installing the CLM, the service must be configured and then started before it can be used.

Upgrading

To upgrade the CLM, simply use the package installer for your operating system:

```
Ubuntu$ sudo dpkg -i clm-3.x.y-amd64-ubuntu.deb
```

```
RHEL$ sudo rpm -U clm-3.x.y-x86_64-rhel.rpm
SLES$ sudo rpm -U clm-3.x.y-x86_64-suse.rpm
```
where the `.deb/.rpm` installation package is provided by Cryptify.

Diagnostic logging

The purpose of the CLM is to transfer logging data from various sources to any number of sinks. During normal operation, however, the CLM itself generates very little diagnostic output.

The diagnostic output that is generated, however, is automatically written to the standard system log provided by `systemd`, which is accessed using `journalctl`.

Some common commands to read the log file include:

- Read the entire log file:
  `$ sudo journalctl -u clm`
- Read the end of the log file and wait for new messages:
  `$ sudo journalctl -f -u clm`
- Read messages logged within the last hour:
  `$ sudo journalctl -u clm --since "1 hour ago"`
- Read the last 100 messages and wait for new messages:
  `$ sudo journalctl -u clm -f -n 100`

Note that by default, the `systemd` journal is *not* persisted past reboots. To persist the log file past reboots on a default system installation, run:

```
$ sudo mkdir -p /var/log/journal
$ sudo systemd-tmpfiles --create --prefix /var/log/journal
$ sudo systemctl restart systemd-journald
```

Starting and stopping

The CLM service is controlled using the standard `systemd` commands.

Some common commands include the following.

- Starting the service:
  `$ sudo systemctl start clm`
- Stopping the service:
  `$ sudo systemctl stop clm`
- Checking the status of the service:
  `$ sudo systemctl status clm`

Uninstallation

To uninstall the CLM, use the package manager for your operating system:

```
Ubuntu$ sudo dpkg -r clm
RHEL$ sudo rpm -e clm
SLES$ sudo rpm -e clm
```

Uninstalling the CLM does not remove the configuration files.

Configuration

The CLM is configured using the included `clm-cli` program. After changing the configuration, the CLM must be restarted for the changes to take effect:

```
$ sudo systemctl restart clm
```

Configuring sources

Adding sources

To add a source that the CLM will monitor, use:

```
$ clm-cli source create <identifier> <type>
```

where *<identifier>* is a name, which will later be used to reference the source, and *<type>* is one of:

- `file` *<path>*
  - Monitors the plain text file at *<path>* for new lines.
- `local-crs` *<trace-type>*
  - Contains the data displayed in "`crs-cli trace` *<trace-type>*" on the local machine.

For each added source *<identifier>*, there is also a virtual source named *<identifier>*`.error` that contains any errors encountered during data collection (such as failing to connect to the CRS or failing to read from the file).

Removing sources

To stop the CLM from monitoring a source, use:

```
$ clm-cli source delete <identifier>
```

where *<identifier>* is the name of the source to stop monitoring.

Showing sources

To show the current sources, use:

```
$ clm-cli source list
```

To get more information about a specific source, use:

```
$ clm-cli source show <identifier>
```

Configuring sinks

Creating sinks

To create a sink that the CLM can output data to, first create it using

```
$ clm-cli sink create <identifier> <type>
```

where *<identifier>* is a name, which will later be used to reference the sink, and *<type>* is one of:

- `file` *<path>*
  - Writes the collected data to a plaintext file at *<path>*.
- `syslog`
  - Writes the collected data to the system log.

Initially, a sink has no sources configured and no data will be written to it until one or more sources has been added to it.

Modifying sink sources

For a source to be written to the sink, it must be added using:
`$ clm-cli sink modify <sink-identifier> add source <source-identifier>`
where *<sink-identifier>* is the sink to modify and *<source-identifier>* is a source that should be written to the sink. Similarly, sources can be removed using `remove source`.

Showing sources

To show the current sinks, use:
`$ clm-cli sink list`
and to show information about a specific sink, use:
`$ clm-cli sink show <identifier>`

Example: Write CRS logs to the system log

First, we create the necessary sources:

```
$ clm-cli source create crs-attach-trace local-crs attach
$ clm-cli source create crs-connection-trace local-crs connection
$ clm-cli source create crs-session-trace local-crs session
$ clm-cli source create crs-event-trace local-crs event
$ clm-cli source create crs-message-trace local-crs  message
$ clm-cli source create crs-object-trace local-crs object
$ clm-cli source create crs-log file /var/log/crs/crs.log
```

We then create a sink, named “main” and written to the system log, and add the newly created sources to it:

```
$ clm-cli sink create main syslog
$ clm-cli sink modify main add source crs-attach-trace
$ clm-cli sink modify main add source crs-connection-trace
$ clm-cli sink modify main add source crs-session-trace
$ clm-cli sink modify main add source crs-event-trace
$ clm-cli sink modify main add source crs-message-trace
$ clm-cli sink modify main add source crs-object-trace
$ clm-cli sink modify main add source crs-log
```

To aid in troubleshooting, we also configure so that errors encountered during data collection are written to the same sink:

```
$ clm-cli sink modify main add source crs-attach-trace.error
$ clm-cli sink modify main add source crs-connection-trace.error
$ clm-cli sink modify main add source crs-session-trace.error
$ clm-cli sink modify main add source crs-event-trace.error
$ clm-cli sink modify main add source crs-message-trace.error
$ clm-cli sink modify main add source crs-object-trace.error
$ clm-cli sink modify main add source crs-log.error
```

We can then verify the configuration:

```
$ clm-cli sink show main
Name: main
Type: syslog
Sources:
    - crs-attach-trace (type local-crs attach)
```

```
- crs-connection-trace (type local-crs connection)
- crs-session-trace (type local-crs session)
- crs-event-trace (type local-crs event)
- crs-message-trace (type local-crs message)
- crs-object-trace (type local-crs object)
- crs-log (type file /var/log/crs/crs.log)
- crs-attach-trace.error (errors of crs-attach-trace)
- crs-connection-trace.error (errors of crs-connection-trace)
- crs-session-trace.error (errors of crs-session-trace)
- crs-event-trace.error (errors of crs-event-trace)
- crs-message-trace.error (errors of crs-message-trace)
- crs-object-trace.error (errors of crs-object-trace)
- crs-log.error (errors of crs-log)
```

Finally, we restart the CLM in order for the changes to take effect:

```
$ sudo systemctl restart clm
```

When running redundant CRSes, the commands above are repeated on the server running the second CRS instance as well.

Example: Maintain a persistent CRS call log

This example achieves roughly the same as `crs-cli trace session >> /home/clmuser/crs-session.log 2>>/home/clmuser/crs-session.err`, except that the event trace is restarted as needed:

```
$ clm-cli source create crs-session-trace local-crs session
$ clm-cli sink create call-log file /home/clmuser/crs-
session.log
$ clm-cli sink modify call-log add source crs-session-trace
$ clm-cli sink create call-log-errors file /home/clmuser/crs-
session.err
$ clm-cli sink modify call-log-errors add source crs-session-
trace.error
$ sudo systemctl restart clm
```

To write the log elsewhere, ensure that the user `clmuser` has permission to open a file at the specified path for appending, for instance by first creating the file as root and then changing the owner of the file:

```
$ sudo touch /var/log/crs-session.log
$ sudo chown clmuser /var/log/crs-session.log
```

Warning

When using file sources and sinks, it is important to ensure that a loop is not created. That is, the same file should not be used as both a sink and a source, since doing so creates an infinite message loop.