Cave Canem—An extensible DDS-based monitoring and Intrusion Detection System

Getting Started Guide

1. Introduction

Performance and health monitoring are essential to ensure that a system meets its operational needs—especially in distributed systems. Currently, there are plenty of solutions addressing those issues, however, they tend to serve a single purpose (e.g., performance monitoring vs. intrusion detection) and to be limited in scalability (e.g., based on a centralized client-server model).

In this context, we propose Cave Canem, a C++ open distributed framework for the collection and integration of events, alerts, and information regarding the system status. To achieve this goal, Cave Canem builds a Common Operational Picture (COP) to combine information from disparate sensor classes. Its COP normalizes all the relevant information from those sensors into a distributed operational model, making the information available for observation in a seamless and efficient manner.

To construct the COP, Cave Canem exploits the benefits of the Object Management Group (OMG) Data Distribution Service for Real-Time Systems (DDS) standard. DDS defines a data-centric publish-subscribe (DCPS) communication model to exchange information. It allows the middleware infrastructure to reflex the essential aspects of the information model, cache the required information, provide content- and time- based filtering and deliver information to the applications with the correct quality-of-service (QoS).

2. Installation

2.1. RTI Data Distribution Service

This version of Cave Canem includes the RTI Data Distribution Service libraries for:

- GNU/Linux 32 and 64 bits.
- Solaris i286 and sparc.

Therefore, not further configuration is needed.

2.2. Compiling Cave Canem

To compile Cave Canem you can use the makefiles provided in each directory. Each makefile detects automatically the version of the operating system running and compiles the application to work on it. To run the makefile just type:

```
$ make
```

If you want to compile Cave Canem for debugging, use:

```
$ make DEBUG=1
```

By default, Cave Canem uses GCC (g++) and Make for compiling. However, you can change that configuration by editing the `CC` and `MAKE` parameters in *shared/config.mk*.

3. Running Cave Canem

To run Cave Canem, a BASH script called *cavecanem.sh* is provided. It can be used as follows:

```
$ ./cavecanem.sh       # runs the application
$ ./cavecanem.sh start # runs Cave Canem in background
$ ./cavecanem.sh stop  # stops Cave Canem
```

Note: If you want to deploy Cave Canem on different architectures, please take into account that a new compilation will overwrite binaries and libraries previously compiled.

4. Configuring Cave Canem

By default, Cave Canem will load the following plugins:

- CPU—publishes CPU usage and load in the host where Cave Canem is running.
- Memory—gathers information about the usage of memory and swap.
- Disk—provides information regarding the usage of file systems.
- Net Load—publishes information about the configuration, usage, and errors of network interfaces.
- Host Info—provides information about the operating system running, as well as the uptime of the machine.
- Proc—gathers information about each process running on the host.
- Proc Stat—provides information regarding the general status of the processes running on a host —number of processes, threads, etc.

You can add new plugins specifying their path and name in *config/cavecanem.xml*, which also includes the DDS domain ID where data will be published, as well as the rate the plugin manager will poll for new data.

The configuration folder contains also *cavecanem_qos.xml*, which includes a set of QoS libraries and policies to tune the dissemination of the information provided by the plugins.

4.1. Configuring QoS libraries and profiles

Cave Canem provides two QoS profiles—within two QoS libraries—in *cavecanem_qos.xml* to tune the dissemination of information using DDS.

The first one, called testing, can be used for testing purposes. It defines a reliable communication to ensure that the messages are received by the subscriber.

The second one, called deployment, is the one that should be used in real deployments. It defines reliable communication, and configures DDS DataWriters to keep the data generated for sending it to late-joiner subscribers trying to get monitoring information.

QoS libraries and profiles are defined both in *cavecanem.xml* and in the plugins' configuration files. QoS libraries and profiles in *cavecanem.xml* are related to the DDS Domain Participant and the DDS Publisher, whereas QoS libraries and profiles defined in the plugins' configuration files are related to the DDS DataWriters. Plugins' configuration files allow also the in-line definition of QoS for the DDS DataWriter. It can be done adding the `<datawriter_qos>` tag—containing QoS definitions using the correct syntax—within the `<dds_properties>` tag and removing `<dds_qos_library>` and `<dds_qos_profile>` within that same tag.

4.2. Configuring publishing rates

Cave Canem allows different publication rates in each plugin. There are two configuration files involved. First, in *config/cavecanem.xml* the `<publishing_period_sec>` tag defines the rate Cave Canem's plugin manager will poll for new data—is set to 1 second by default. On the other hand, each plugin has a `<publishing_period_sec>` tag in its configuration file, which defines the rate for gathering and sending monitoring information. By default it is set to 300 seconds (5 minutes), so Cave Canem will publish information every 5 minutes—checking whether a plugin is ready to publish every second.

4.3. Configuring Snort plugin

Snort plugin is not enabled by default. It collects and publishes alerts generated by the Instruction Detection System (IDS) Snort using its CSV output. Therefore, an *alert.csv* file must exist and its path must be configured accordingly in the plugin's configuration file—*plugins/snort/snort.xml*.
To configure Snort for using the CSV output plugin you must add the following line to the Snort's configuration file—usually located at */etc/snort/snort.conf*:

```
output alert_csv: /var/log/snort/alert.csv default
```