2017 Codebreaker Challenge Task 0 Setup

For the 2017 Codebreaker Challenge you'll want to set up an instance of the test system using the files provided to you in Task 0. These instructions will walk you through this setup, and ultimately help you complete this task.

Background

The test system consists of multiple command line applications and binary files which work together to emulate a rudimentary SCADA sensor network. The two most important applications are the agent and the agent_controller – the agent runs on various entities in the network and waits for commands from the agent_controller. These two applications talk to one another via MQTT (a light-weight publish/subscribe messaging protocol) messages passed via an intermediary server. To successfully run the agent and agent_controller binaries, you will first need to setup an MQTT server.

For your test system, you will set up an instance of the open-source Mosquitto MQTT server inside a docker container (the instructions below will walk through this process.) The system uses a custom authentication plugin on top of MQTT, and the compiled library file that provides this functionality has been provided. Once your server is running, you can begin using the agent and agent_controller binaries.

One action that the agent_controller can perform on a given agent is to install various modules into the agent that add new functionality. Two modules are provided for use in your test system – one module that listens for messages from the agent_controller and writes these messages to the terminal window, and another module that tests publishing messages from the agent. The instructions below will demonstrate how to load these binaries.

Step 0. Setup a Linux VM to use for the Challenge

The challenge binaries this year only work on Linux, so you'll need to set up a Linux environment to use them in. We recommend setting up a virtual machine using Ubuntu 16.04 as the operating system, available from the link below.

http://releases.ubuntu.com/16.04/ubuntu-16.04.3-desktop-amd64.iso

Note: The SHA1 sum of the downloaded binary should equal
da21a922bd19b3b87931ea44095a02efba2decfe

There are multiple virtual machine software packages that you can use, and there are multiple online guides that will walk you through setting one up. One such guide can be found at:

https://linus.nci.nih.gov/bdge/installUbuntu.html

Note: You only need to work through the 'Download', 'Create Virtual Machine', and 'Install Ubuntu' portions of the tutorial linked to above. Also, although the tutorial walks through installing Ubuntu 14.04, we recommend installing Ubuntu 16.04 instead.

Step 1. Download Challenge Files

Download the challenge files provided as part of Task 0 into your virtual machine and unzip the files.

Step 2. Download mosquitto-docker.zip

Download the zip file that you'll need to setup your local MQTT server. Unzip the files.

Step 3. MQTT Server Setup

To make setup easy for you, we've provided a docker file which will create an instance of the server on your local system.

Open a terminal window and change directories into the unzipped challenge files.

```
cd Downloads/t0_test_system
```

Copy libauth_plugin.so into the mosquitto-docker folder that was created when you unzipped mosquitto-docker.zip

```
cp libauth_plugin.so ~/Downloads/mosquitto-docker/
```

Install docker by running the following command:

```
sudo apt-get install docker.io
```

Build the server and run it with the following commands:

```
sudo docker build --rm . -t mosquitto
```

```
sudo docker run --rm -p 127.0.0.1:8883:8883 mosquitto
```

The following output indicates that your MQTT server is successfully up and running:

```
<timestamp>: mosquitto version 1.4.14 (build date 2017-09-14 03:28:33+0000) starting
<timestamp>: Config loaded from /etc/mosquitto/conf.d/local.conf.
<timestamp>: Opening ipv4 listen socket on port 8883.
```

Leave the MQTT server running in that terminal window and continue on to Step 4 below.

Step 4. Install Packages to Allow 32-bit Applications to Run

In your VM, ensure that you can run 32-bit applications by installing the 32-bit C library:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386
```

Step 5. Run the agent and agent_controller Binaries

From a new terminal window, make the agent and agent_controller binaries executable:

```
chmod 755 agent agent_controller
```

Run the agent binary

```
./agent
```

You should see a message that says:

```
Successfully connected to client with OTP key #1
```

In a separate terminal window, run through the following sequence of commands:

1. Load the log module

   ```
   ./agent_controller -n log -i agent_module_log.so
   ```

2. Send a command to log the “hello world!” message

   ```
   ./agent_controller -m "hello world!"
   ```

3. Uninstall the log module

   ```
   ./agent_controller -u log
   ```

4. Shutdown the agent

   ```
   ./agent_controller -d
   ```

The agent binary should have successfully printed the “hello world!” log message and exited.

T0 Solution:

After handling a command, the agent prints the following message to the terminal:

```
Agent <Agent ID> successfully handled message
```

Submit your <Agent ID> to the website to verify that you’ve completed Task 0.