ROS 2 Security Working Group Meeting Notes

Working Group Calendar URL | Google Group for invites | ROS Security Meeting

Please join the Google group to get calendar invites to the meeting and be able to comment on this doc.
Want to help facilitate this working group? Contact tmoulard@amazon.com

2019-10-29

Agenda:

- Security tooling for ROS 2-- any desires? Monitoring tools?
- How should security vulns against ROS be reported? An email list? Who should be on it? Disclosure policies?

2019-09-17 (next meeting)

Meeting Announcement (Discourse) | Meeting Recording | Chime Link

Agenda

- Announcements (5 min)
- Roslaunch2 sandboxing (10-15 min)
- Discussion DDS participants (10-15 min)

Announcements

- Canonical will start chairing the Security WG
- Joe McManus will lead the ROS 2 security WG

roslaunch2 sandboxing

https://github.com/aws-robotics/launch-ros-sandbox/tree/master/examples

- **Simple**
    - Execute ROS 2 apps w/o installing ROS2 - you just need to install roslaunch2 with PIP and you can start any ROS 2 on any platform (OS X, Windows, Linux)[^]
- **Safe**

    - Your application run in one or more Docker containers sandboxing nodes from each other and from the host system.
    - Use Docker resource limits to protect your nodes and budget the resources allocated to every one of them
- **Distribute your nodes easily as Docker images**
    - No more Jenkins & Bloom hell, build your image once, upload it to a Docker repo and let anyone use your code.
    - Rely on ROS2 OSRF images and Docker/GitHub integration to setup your CD pipeline!
- **Build ROS2 robot applications easily**
    - Large ROS2 (or ROS1) apps are currently expressed through ROS packages.
    - Cumbersome - each node the app relies on needs to be a dependency in the package.xml. If compiling from source, can take a very long time.
    - Some cloud CI services such as Travis have a hard timeout on builds, avoid it by pulling images instead!
    - Instead, your app can become a single, self-contained Python file.
    - Pull all nodes as Docker images instead
    - Store in your application which version of which node you want to run by specifying an image tag (impossible to do currently with Bloom+APT)
    - No need to necessarily Bloom release everything (can use custom images - just need to publish the image in some repo like Dockerhub).
    - Don't wait X months for updates anymore! Update your images to a newer version your dependencies instantaneously!
- **Build advanced e2e and integration tests easily**
    - Build non-flaky, meaningful integration tests. If your integration test depends on e.g. robot_state_publisher and you pull it using test_depend, if it breaks, you break! Use a Docker image to pull robot_state_publisher from a ROS2 release easily. And pin the image to avoid E2E test flakyness!
    - Write cross-distro tests (talker is a Crystal node, listener is a Dashing node)
    - Write cross-RMW tests (talker is fastrtps, listener is RTI Connext)
    - Write e2e tests simulating multiple architectures with qemu and Docker (listener is ARMHF, talker is X86)
    - Use Docker tools such as Pumba to simulate network latency in your e2e tests (a la Chaos Monkey) - are you sure your application is safe when the WiFi quality drops?
    - Use Docker resource limits to starve your nodes - how will your robot react when it will run of memory?

[^] Bloom release is planned, release through PIP should be possible but is not scheduled yet.

Discussion DDS participants

https://github.com/ros2/design/pull/250
https://discourse.ros.org/t/reconsidering-1-to-1-mapping-of-ros-nodes-to-dds-participants/10062

iRobot Performance analysis:
https://github.com/irobot-ros/ros2-performance/tree/master/performances/benchmark

2019-08-21

Meeting Announcement (Discourse) | Meeting Recording | Chime Link

ROS 2 Access Control Policies

@ruffsl

PR for access control policies

- Contains the design as well as the schema
    - Can grant general permissions and then revoke
    - Evaluated in order; can interleave allow and deny
- Supports topics, services, actions (parameters TBD)
- Independent from the DDS layer
- Templates
- Uses XML over YAML for parsing/validating
- “Color” profiles; use to whitelist communications between other profile

Feedback

- Runtime errors if two nodes try using mismatched DDS?
- Composability might be an issue due to not knowing about Pub/Sub relationship
    - Know graph if policies are part of ROS Launch
- Encryption might have a performance impact
    - Off by default?

ROS 2 Node Sandboxing

Zach (AWS)

The objective is to extend roslaunch with a syntax allowing nodes to be sandboxed using various methods. We are looking at implementing a policy relying on Docker containers as an extension to ROS Launch.

Feedback

- Might need to bootstrap ROS launch for running in a container
- Nodes might be looking for configs at specific locations
- How to handle physical devices attached to robots

    - GPUs
    - Lidar

2019-08-01

Meeting Announcement (Discourse) | Meeting Recording | Chime Link

Calendar Invites:

- Working Group Calendar URL
- Google Group for invites

Generic Node Interface Description

@kyrofa

https://docs.google.com/presentation/d/1-Mdy9VE0nKgUbOUQY9S8ELCoyMFBtx-mRTfX8q3eyal/edit?usp=sharing

Feedback:

- Dynamic behavior is hard to cover.
    - How to define an interface if nodes are spawned on the fly?
    - Maybe grouping nodes together is a way to simplify the problem?
- Useful to populate ROS 1 like node API documentation pages
- Where to put the info?
    - package.xml?
    - Probably need to be installed (distributed as part of the Debian package)
    - One single repository for all nodes?
- Are policies needed if you have good node interface?
    - Can generate policy through your build system.
- What about having the IDL the input and consume it from the C++ side?

DDS-XML spec: https://www.omg.org/spec/DDS-XML/About-DDS-XML/
Example of use for a talker listener:
https://www.omg.org/spec/DDS-XML/20180501/dds-xml_applications_example.xml