SROS 2

Mikael Arguedas
IROS 2018, Madrid
SROS2

- What is ROS 2
- Interfacing DDS-Security to the ROS 2 stack
- Use the sros2 command line interface
- Run some basic examples
ROS as we know it
Characteristics of systems initially targeted by ROS in 2007

- Research applications
- High-volume sensors
- Complex kinematics
- Lots of computation power
- Ideal network connectivity
Characteristics of small robotic systems today

- Multi-robot system
- Distributed system
- Small processors
- Battery power
- Unreliable network connectivity
Goals of ROS 2

Support multi-robot systems involving unreliable networks

Remove the gap between prototyping and final products

“Bare-metal” micro controller

Support for real-time control

Cross-platform support

http://design.ros2.org/articles/why_ros2.html
ROS 2

Plumbing + Tools + Capabilities + Ecosystem
ROS 2

ROS usability

Plumbing + Tools + Capabilities + Ecosystem

less time spent here means more time to spend here
ROS 2 Releases

December 2017

June 2018
Architectural overview

User code

ROS client library API
Architectural overview

User code

ROS client library API

DDS implementation

= discovery + serialization + transport
Architectural overview

User code

ROS client library API

- DDS impl A or DDS impl B or ...

open robotics
Architectural overview

User code

ROS client library API

ROS middleware API

DDS impl A or DDS impl B or ...

DDS agnostic

ROS agnostic
Architectural overview

User code

ROS client library API

ROS middleware API

 DDS impl A or DDS impl B or ...

 DDS agnostic

 ROS agnostic
Architectural overview

User code

ROS client library API

ROS middleware API

RMW impl A

RMW impl B

... or

DDS impl A

or

DDS impl B

or

...
“Hour Glass” Pattern

- rclcpp
- rclpy
- rclcs
- rcljava

rcl

rcl impl

rmw

{rmw impl}

{DDS vendor}
“Hour Glass” Pattern

- rclcpp
- rclpy
- rclcs
- rcljava

rcl

rcl impl

rmw

rmw_fastrtps_cpp

rmw_connext_cpp

eProsima
Fast-RTPS

RTI
Connext
Where does SROS 2 live?

User code should **not** change

Plugin instantiation

DDDS-Security implementation

---

19
Where does SROS 2 live?

Environment variables checking

ROS_SECURITY_ENABLE → Should we look for security artifacts?

ROS_SECURITY_STRATEGY → Should we prevent unauthenticated nodes from being created?

ROS_SECURITY_ROOT_DIRECTORY → Where to look for artifacts
Where does SROS 2 live?

**Environment variables checking**

- **ROS_SECURITY_ENABLE**: true/false
- **ROS_SECURITY_STRATEGY**: Permissive/Enforce
- **ROS_SECURITY_ROOT_DIRECTORY**: <path/to/keystore>
Where does SROS 2 live?

Environment variables checking
+ Keystore node structure checking

Extract node security directory path
Where does SROS 2 live?

Retrieve security artifacts + Instantiate plugins accordingly

- rclcpp
- rclpy
- rcl
- rcl impl
- rmw
- rmw_fastrtps_cpp
- rmw_connext_cpp
- eProsima
  - Fast-RTPS
- RTI
  - Connext
Where does SROS 2 live?

Retrieve security artifacts

rmw

rmw_fastrtps_cpp

rmw_connext_cpp

```cpp
bool get_security_file_paths(
    std::array<std::string, 6> & security_files_paths, const char * node_secure_root) {
    // here assume only 6 files for security
    const char * file_names[6] = {
        "identity_ca.crt.pem", "cert.pem", "key.pem",
        "permissions_ca.cert.pem", "governance.pem", "permissions.pem"
    };
    size_t num_files = sizeof(file_names) / sizeof(char *);
    std::string file_prefix("file://");
    for (size_t i = 0; i < num_files; i++) {
        rctools_allocator_t allocator = rctools.get_default_allocator();
        char * file_path = rctools_join_path(node_secure_root, file_names[i], allocator);
        if (!file_path) {
            return false;
        }
        if (rctools_is_readable(file_path)) {
            security_files_paths[i] = file_prefix + std::string(file_path);
        } else {
            allocator.deallocate(file_path, allocator.state);
            return false;
        }
        allocator.deallocate(file_path, allocator.state);
    }
}```
Where does SROS 2 live?

Instantiate security plugins

rmw

rmw_fastrtps_cpp

rmw_connext_cpp

```c
#include <std::string, std::array path, security_files_paths>

if (get_security_file_paths(security_files_paths, security_options->security_root_path)) {
    eprosima::fastrtps::rtps::PropertyPolicy policy;
    using Property = eprosima::fastrtps::rtps::Property;
    policy.properties().emplace_back(Property("dds.sec.auth.plugin", "builtin.PKI-DH");
    policy.properties().emplace_back(Property("dds.sec.auth.ssl_certificate", security_files_paths[1]));
    policy.properties().emplace_back(Property("dds.sec.auth.ssl_private_key", security_files_paths[2]));
    policy.properties().emplace_back(Property("dds.sec.crypto.plugin", "builtin.AES-GCM-GMAC");
    policy.properties().emplace_back(Property("dds.sec.access.builtin.Access-Permissions.permissions_ca", security_files_paths[3]));
    policy.properties().emplace_back(Property("dds.sec.access.builtin.Access-Permissions.permissions", security_files_paths[4]));
    policy.properties().emplace_back(Property("dds.sec.access.builtin.Access-Permissions.permissions", security_files_paths[5]));
    participant_rtps.properties = policy;
} else if (security_options->enforce_security) {
    return msg("couldn't find all security files!");
}
```
How to generate SROS 2 artifacts?

- Setting up your environment:
  - $ source /opt/ros/bouncy/setup.bash

- Create a keystore:
  - $ ros2 security create_keystore my_keystore

```
root@9db8b460bb4f:~# ll my_keystore
total 36
drwxr-xr-x  2 root root  4096 Sep  8 19:54 .
drwxr-x---  4 root root  4096 Sep  8 19:54 ..
-rw-r--r--  1 root root  477 Sep  8 19:54 ca.cert.pem
-rw-r--r--  1 root root  241 Sep  8 19:54 ca.key.pem
-rw-r--r--  1 root root 1112 Sep  8 19:54 ca_conf.cnf
-rw-r--r--  1 root root   75 Sep  8 19:54 ecdsaparam
-rw-r--r--  1 root root 3325 Sep  8 19:54 governance.p7s
-rw-r--r--  1 root root 1783 Sep  8 19:54 governance.xml
-rw-r--r--  1 root root   0 Sep  8 19:54 index.txt
-rw-r--r--  1 root root   4 Sep  8 19:54 serial
```
How to generate SROS 2 artifacts?

- Create key and wildcard permissions for a node:
  - `$ ros2 security create_key my_keystore my_node`

```
root@9db8b460bb4f:~# ls my_keystore/my_node/
cacert.pem governance.p7s permissions.xml
cert.pem key.pem req.pem
ecdsaparam permissions.p7s request.cnf
```
How to generate SROS 2 artifacts?

- Create policies files for a node:

```python
1 nodes:
2   my_node:
3     topics:
4       my_pub_topic:
5         allow: p
6       my_sub_topic:
7         allow: s
```
How to generate SROS 2 artifacts?

- Create permission files for a node:
  - `$ ros2 security create_permissions ./my_keystore my_node ./my_node_policies.yaml`
Let’s try it!

$ docker run -it --rm osrf/ros2:bouncy-desktop
# source /opt/ros/bouncy/setup.bash
# mkdir ~/my_ros2_ws && cd ~/my_ros2_ws
# ros2 security create_keystore demo_keys
# ros2 security create_key demo_keys talker
# ros2 security create_key demo_keys listener

Now let’s run our secure nodes:
# export ROS_SECURITY_ROOT_DIRECTORY=~/my_ros2_ws/demo_keys
# export ROS_SECURITY_ENABLE=true
# export ROS_SECURITY_STRATEGY=Enforce

# ros2 run demo_nodes_cpp talker &
# ros2 run demo_nodes_py listener
Let’s try it!

Access Control:

Create ~/my_ros2_ws/pub_sub_policies.yaml with:

```yaml
nodes:
  listener:
    topics:
      chatter:
        allow: s # can subscribe to chatter
  talker:
    topics:
      chatter:
        allow: p # can publish on chatter
```
Let’s try it!

Create the permissions:

# ros2 security create_permission demo_keys talker pub_sub_policies.yaml
# ros2 security create_permission demo_keys listener pub_sub_policies.yaml

# ros2 run demo_nodes_cpp talker &
# ros2 run demo_nodes_py listener
Let’s try it!

Let’s remap the topic on which talker publishes:

```
# ros2 run demo_nodes_cpp talker chatter:=my_chatter
```

```
[SECURITY Error] Error checking creation of local writer 9a.dd.34.40.e7.49.82.27.af.91.a8.29|0.0.1c.3 (rt/my_chatter) topic not found in allow rule. (/tmp/binarydeb/ros-bouncy-fastrtps-1.6.0/src/cpp/security/accesscontrol/Permissions.cpp:1085)
  -> Function register_local_writer
[PARTICIPANT Error] Problem creating associated Writer -> Function createPublisher
```
Seeing in wireshark (clear text)
Seeing in wireshark (encrypted)