Overview

- **Background**
  - Summarized security issues in ROS1

- **Related Work**
  - What prior approaches have been proposed

- **Current Work**
  - Present development for SROS2

- **Future Work**
  - TODOs and action items for SROS2

- **Conclusions**
  - Closing remarks and observations
Background | Robotic Frameworks

- **ROS**
  - Market Expected to Reach US$ 402.7Mn by 2026.
  - +10 years development, +13.4K downloads 2017.
- **Other Examples**
  - Player, YARP, Orocos, CARMEN, Orca, MOOS, Microsoft Robotics Studio, LabVIEW Robotics, MATLAB Robotics Toolbox.


Cited 4,430 times as of 2018, up 26% from 2017.
Background | ROS

Robotic Operating System (ROS)

- **Plumbing**: Middleware for process communication
- **Tooling**: Introspective debugging and visualization
- **Capabilities**: Reusable domain specific modules
- **Ecosystem**: Collaborative open source communities
Background | ROS1

- Peer-to-peer pub/sub model formulates **anonymous** computational graph.
- Processes communicate through common APIs over **clear text** transport via topics and services.
- Master provides namespace resolution and **centralized** key-value parameter storage.
- APIs subsystems are unregulated and provide **unauthenticated** access to any connection.
Background | ROS1

Illustrated subscription in ladder diagram

1. Node A sends topic subscription request
2. Master returns publisher list in callback
3. Node A negotiates transport method
4. Node B returns transport specifics
5. Node A connects, receives topic data
Related Work | ROS-RV

Redirecting ROS traffic through MITM mediator

- **Pros**
  - **Runtime Verification**: of message data in flight
  - **Compatibility**: Maintains application API

- **Cons**
  - **Unencrypted**: Transport level still exposed
  - **SPOF**: RVMaster adds a Single Point of Failure
  - **Scalability**: Added overhead from Monitor
  - **Access Control**: Limited to IP level
  - **Flexible**: Not suitable for dynamic networks
  - **Subsystems**: Not all APIs are protected

Enabling private and authentic remote connectivity

- **Pros**
  - **Secure Transport**: via authenticated encryption
  - **Compatibility**: Maintains application API
  - **Dynamic**: Authenticator updates Access Control

- **Cons**
  - **SPOF**: Authenticator adds a Single Point of Failure
  - **Access Control**: authentication but no *authorization*
  - **Subsystems**: Not all APIs are protected
  - **Limited Scope**: Non-native ROS clients only

---

Related Work | ROS-ALG

Application Level Gateway for key distribution

- **Pros**
  - **Dynamic**: DataBase updates Access Control
  - **Accounting**: Enables auditing of events
  - **Compatibility**: Maintains application API

- **Cons**
  - **SPOF**: AA node adds a Single Point of Failure
  - **Custom Crypto**: Rolls own transport encryption
  - **Subsystems**: Not all APIs are protected

---

Related Work | Secure-ROS-Transport

Application Level Gateway for key distribution

- **Pros**
  - Secure Transport: for topics at least
  - ABI: No client library modification

- **Cons**
  - Compatibility: divergent application API
  - SPOF: AA node adds a Single Point of Failure
  - Custom Crypto: Rolls own transport encryption
  - Subsystems: Not all APIs are protected
  - Access Control: authentication but no authorization

Related Work | ROS-AES-Encryption

Decentralised authentication for transport

- **Pros**
  - **Secure Transport**: via authenticated encryption
  - **Standard Crypto**: Use of TLS libraries
  - **Compatibility**: Maintains application API
  - **No SPOF**: Distributed access control
  - **More QoS**: Support DTLS over UDP

- **Cons**
  - **Subsystems**: Not all APIs are protected
  - **Access Control**: authentication but no *authorization*
  - **Coupling**: Identity and permissions are conjoined

Related Work | Secure ROS

Decentralised authentication and authorization

- **Pros**
  - **Secure Transport**: via authenticated encryption
  - **Standard Crypto**: Use of IPSec libraries
  - **Compatibility**: Maintains application API
  - **No SPOF**: Distributed access control
  - **Coupling**: Identity/permissions are loosely conjoined

- **Cons**
  - **Access Control**: Limited to IP level
  - **Flexible**: Not suitable for dynamic networks
  - **Less QoS**: TCP only, so no UDP multicasting

Related Work | SROS1

Decentralised authentication and authorization of full API

- **Pros**
  - **Secure Transport**: via authenticated encryption
  - **Standard Crypto**: Use of TLS libraries
  - **Compatibility**: Maintains application API
  - **Access Control**: Fine grained permissions
  - **Subsystems**: All APIs are guarded
  - **No SPOF**: Distributed access control
  - **Accounting**: Enables auditing of events

- **Cons**
  - **Context Leaking**: Access criteria embedded in identity cert publicly disclosed from TLS handshake
  - **Coupling**: Identity and permissions are conjoined
  - **Less QoS**: TCP only, so no UDP multicasting

---

R. White, M. Quigley, and H. Christensen, “SROS: Securing ROS over the wire, in the graph, and through the kernel,” in Humanoids Workshop: Towards Humanoid Robots OS. Cancun, Mexico, 2016.
Related Work | SROS

Illustrated subscription in ladder diagram

1. Node A starts TLS handshake with master, verifying API permissions before sending topic subscription request.

2. Master returns sanitized publisher list in callback that Node A has permissions to.


4. Node A connects over separate TLS session and receives topic data.
<table>
<thead>
<tr>
<th>Approach</th>
<th>Encryption</th>
<th>Authentication</th>
<th>Authorization</th>
<th>Compatibility</th>
<th>Subsystems</th>
<th>SPOF</th>
<th>QoS</th>
<th>Scalability</th>
<th>Dynamic</th>
<th>Flexible</th>
<th>Accounting</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS-RV</td>
<td>None</td>
<td>IP</td>
<td>API</td>
<td>Topic Only</td>
<td></td>
<td></td>
<td>Rel &amp; Best?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Rosauth</td>
<td>TLS</td>
<td>Token</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td>Rel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>ROS-ALG</td>
<td>Cstm+SSH</td>
<td>Pass</td>
<td>PKI</td>
<td>Topic Only</td>
<td>Rel &amp; Best?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Secure-ROS-Transport</td>
<td>Cstm</td>
<td>PKI</td>
<td>ABI</td>
<td>Topic Only</td>
<td>Rel &amp; Best?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>ROS-AES-Encryption</td>
<td>TLS/DTLS</td>
<td>PKI</td>
<td>PKI</td>
<td>Topic Only</td>
<td>Rel &amp; Best</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tight</td>
</tr>
<tr>
<td>Secure ROS</td>
<td>IPSec</td>
<td>PKI</td>
<td>IP</td>
<td>API</td>
<td>Rel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Loose</td>
</tr>
<tr>
<td>SROS1</td>
<td>TLS</td>
<td>PKI</td>
<td>PKI</td>
<td>All</td>
<td>Rel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tight</td>
</tr>
</tbody>
</table>
Related Work | Comparison

- SROS1 was demonstrated as most secure initiative tested previously
- Given it extensive security layer was designed to envelop the entire API surface
- However, it languished from slow performance, as it was only ever ported to rospy

- Data leaked by each initiative on requests made inside a secured ROS network.

<table>
<thead>
<tr>
<th></th>
<th>ROS</th>
<th>SROS</th>
<th>ROS-AES-Encryption</th>
<th>Secure-ROS</th>
<th>Secure-ROS-Transport</th>
<th>Rosauth</th>
</tr>
</thead>
<tbody>
<tr>
<td>rostopic list</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rosnode list</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rosservice list</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rosnode kill</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rostopic echo</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Current Work | SROS2

Utilizing the DDS Security Specification

- **Pros**
  1. **Secure Transport**: via authenticated encryption
  2. **Standard Crypto**: Use of AES-GCM-GMAC
  3. **Compatibility**: Maintains application API
  4. **Access Control**: Fine-grained permissions
  5. **Subsystems**: All APIs are guarded
  6. **No SPOF**: Distributed access control
  7. **Accounting**: Enables auditing of events
  8. **Coupling**: Identity/permissions are loosely conjoined
  9. **Flexible**: Suitable for dynamic networks
  10. **More QoS**: Support Sign vs Encrypt + existing QoS for DDS
  11. **Plugins**: Customizable for swapping or adding features

- **Cons**
  1. **RMW Specific**: These security features are specific to RMW implementations using DDS; however, security specification is standardized across DDS vendors to facilitate interoperability
Future Work | SROS2 Profiling

- Determining optimum configurations for specific robotic deployment scenarios

- Profiling and Engineering tradeoffs
  - **Power** - energy conservation
  - **Performance** - latency
  - **Bandwidth** - network overhead
  - **Security** - cryptographic strength

- **SROS2 + DDS Security**
  - Embedded devices
  - Real Time systems
  - Wireless links
  - Resilient orchestration

Future Work | SROS2 Action Items

- Finer ROS2 subsystem security
  - Instance level parameter access control
  - Preparing for upcoming ROS2 Actions
  - Hierarchical comms: robot -> fleet -> swarm

- Development and Debug Tooling
  - Assistive permission policy generation
  - Static and runtime security profiling
  - Descriptive connectivity manifests

- Management and Orchestration
  - Procedural provisioning security artifacts
  - Expressive security policy definitions
  - Generation, deployment, revocation of PKI

- Auditing and Logging
  - Distributed logging over networks
  - Recording Security Events levels
  - Cryptographically immutable records

- Trusted Execution Environments
  - Secure DDS support using TEE
  - Sealing/storage of private PKI keys
  - Protecting runtime session keys
  - E.g. Intel SGX, ARM TrustZone

- Security Testing
  - Adding additional automated CI tests
  - Static verification and code analysis

- Upstream DDS Security Issues
  - Leaking of permissions: DDSSEC12-13
  - Data-tag expressions: DDSSEC12-19
  - Instance-Level AC: DDSSEC12-12
  - Lightweight permission serialization
  - Instance vs monolithic permission exchange
Conclusions

Robots, as cyber physical systems, present a host of new security issues. However, the robotic middleware itself needn’t always be a primary issue.

However, this residual security issue in robotics originate and persist from present deficiencies in:

- **Tooling**
  - Making security accessible
- **Usability**
  - Encourage user adoption
- **Standardization**
  - Facilitate interoperability