{,S}ROS

Securing ROS over the wire, in the graph,
and through the kernel

By Ruffin White and Morgan Quigley

What is SROS?

“An addition to the ROS API and ecosystem to support modern cryptography and security measures”

Encryption

- Native TLS support for all socket level communication
- X.509 PKI certificates for chains of trust, authenticity and integrity
- Keyserver for key pair generation and certificate customisation

Access Control

- Definable namespace globbing for node restrictions and roles
- Audit graph network through security logs and events
- User constructed and/or auto trained access control policies

Process Profiles

- Harden node processes on using Linux Security Modules in kernel
- Quarantine a node’s file, device, signal, and networking access
- Reusable AppArmor profile library for ROS

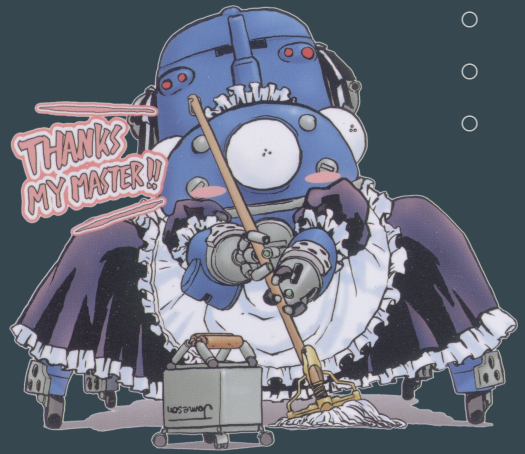
Why SROS?

“Robots provide a vector for cyber threats to manifest into real-world risks.”

ROS’s clear text transport
- Packet Sniffing: *Confidentiality*
- Man-in-the-middle: *Integrity*

ROS’s anonymous graphs
- Message Spoofing: *Identification*
- Rouge Nodes: *Authorization*

ROS’s runtime process
- Code injection: *Compromised Execution*
- Zero Day Exploits: *Altered Permissions*

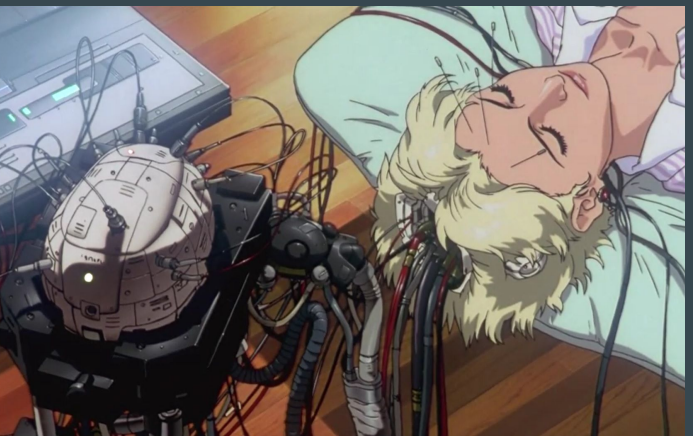


Relevant robotic sectors:
- Industrial Automation
- Autonomous Vehicles
- Home Automation
- Internet of Things, etc.

Why not use VPNs or SDNs?

Pros:

- Tunneling over distant networks
  - ROS allocates ports over runtime; ephemeral port firewall = *whack-a-mole*
- Professional routers and switches
  - Cryptographic overhead can be offloaded to dedicated hardware
- Existing Infrastructure
  - A mature technology; well documented and understood



Cons:

- Session Hijacking
  - ROS is still exposed to attacks from within the local network
- Stack Complexity
  - Additional system layers to configure and regulation
- Fine Grain Control
  - Difficult to integrate and segregate subdomains of ROS graph


SECURING ROS OVER THE WIRE, IN THE GRAPH, AND THROUGH THE KERNEL
{,S}ROS
CAUTION
BY:
RUFFIN
WHITE,
MORGAN
QUIGLEY

Encryption

X.509 Public Key Infrastructure (PKI)

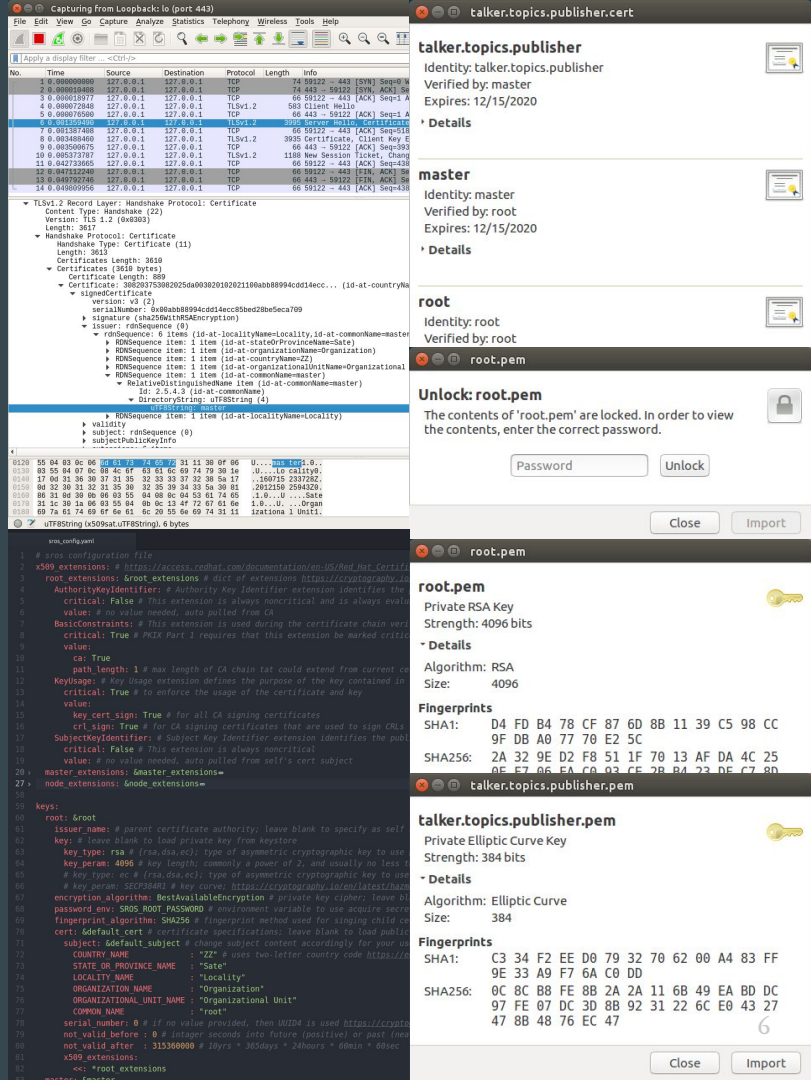
- DSA, RSA, Elliptic Curve Keysigning
- Leverage chains of trust to verify validity and authenticity

Transport Layer Security (TLS)

- Wrap XMLRPC and TCPROS communication
- Leverage socket encryption to verify integrity and privacy

Keyserver and Keystores

- API to auto generate and distribute ciphered key pairs
- Customize X.509 certificate extensions and attributes



Access Control

Defining Namespaces Permissions

- Permit/exclude graph permissions via ROS namespaces
- Similar Apparmor globbing syntax using wildcards

Auditing and Security Logs

- Granular logging of access events and violation attempts
- Audit/deny ROS API usage with rule modifiers

Building Policies

- Auto-generate/train policies from demonstration or logs
- Human readable yaml, leverage variables & node anchors

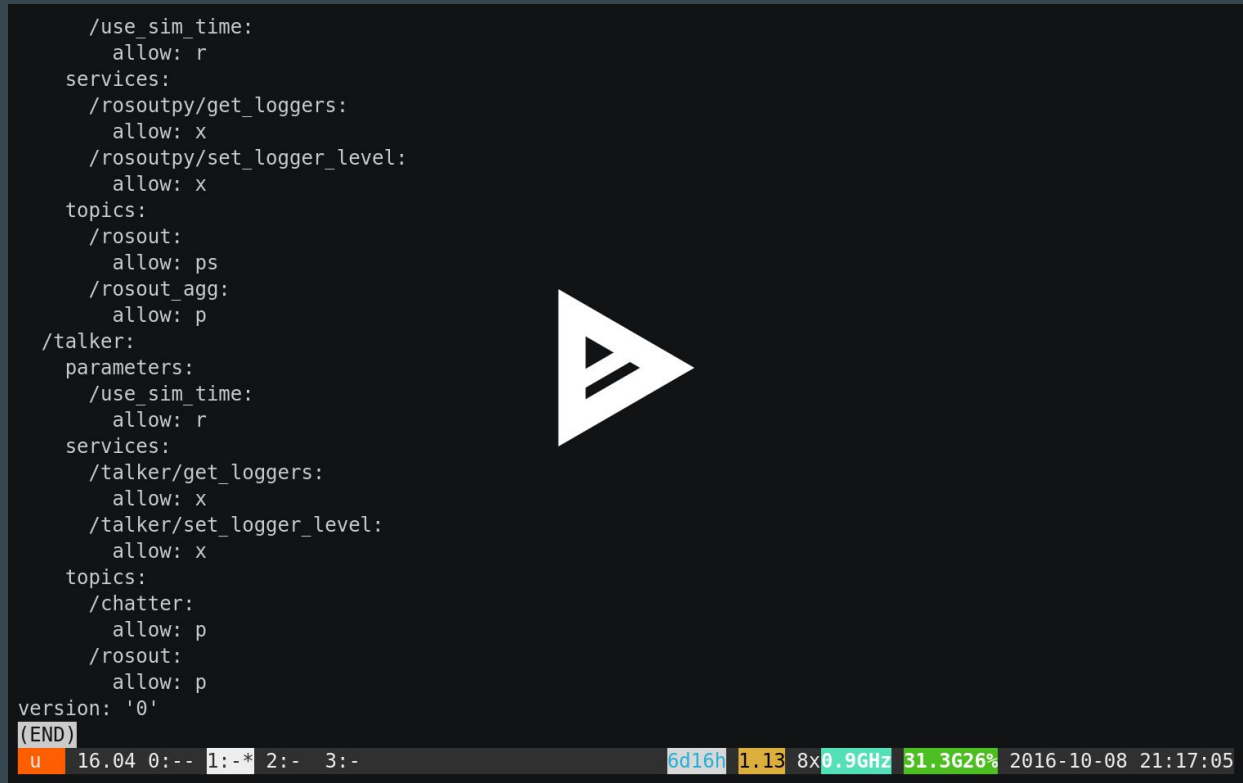
policy_config.yaml

```yaml
nodes:
  /listener:
    parameters:
    services:
    topics:
      /chatter:
        allow: s
      /rosout:
  /roslaunch:
    parameters:
      /rosdistro:
        allow: w
      /roslaunch/uris/*:
        allow: w
      /rosversion:
        allow: w
      /run_id:
        allow: rw
  /rosoutpy:
    parameters:
      /enable_statistics:
        allow: r
      /tcp_keepalive:
        allow: r
      /use_sim_time:
        allow: r
    services:
      /rosoutpy/get_loggers:
        allow: x
      /rosoutpy/set_logger_level:
        allow: x
    topics:
      /rosout:
        allow: ps
      /rosout_agg:
        allow: p
  /talker:
    parameters:
    services:
      /talker/get_loggers:
        allow: x
      /talker/set_logger_level:
        allow: x
    topics:
      /chatter:
        allow: p
      /rosout:
        allow: p
version: '0'
```

policy_config.yaml 63:1

Encryption & Access Control Demo

https://asciinema.org/a/88519



Process Profiles

Linux Security Modules

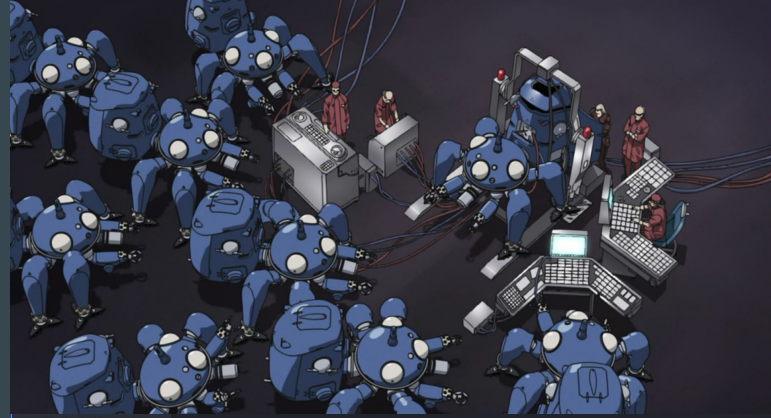
- Leverage pre-existing kernel level security features
- AppArmor: easy to use and well documented

Quarantine ROS Process

- Limit node's file, device, signal and networking permissions
- Preemptively defend against zero day exploits

ROS Profile Library

- Quickly build custom profiles using ROS module primitives
- Includes minimal permissions necessary for core ROS features



roslaunch_talker_listener_profile

```
#include <tunables/global>
#include <tunables/ros>

profile ros/roslaunch @{ROS_INSTALL_BIN}/{,s}roslaunch {
  #include <ros/nodes/roslaunch>

  @{ROS_INSTALL_BIN}/roslaunch rix,
}

profile ros/rosmaster @{ROS_INSTALL_BIN}/rosmaster {
  #include <ros/base>
  #include <ros/node>
  #include <ros/python>

  @{ROS_INSTALL_BIN}/rosmaster rix,
}

profile ros/roscore @{ROS_INSTALL_BIN}/{,s}roscore {
  #include <ros/nodes/roslaunch>

  @{HOME}/.rnd r,
  @{ROS_INSTALL_BIN}/roslaunch rix,
  @{ROS_INSTALL_BIN}/{,s}roscore rix,
}

profile ros/rosout @{ROS_INSTALL_LIB}/rosout {
  #include <ros/base>
  #include <ros/node>
```

Process Profiles Demo

https://asciinema.org/a/88531

TODO:

SROS related REPs

- PKI practices, regester OIDs, extensions

More client libraries and transports

- roscpp, rosjava; UDPROS, etc

Harden all Master & Slave API calls

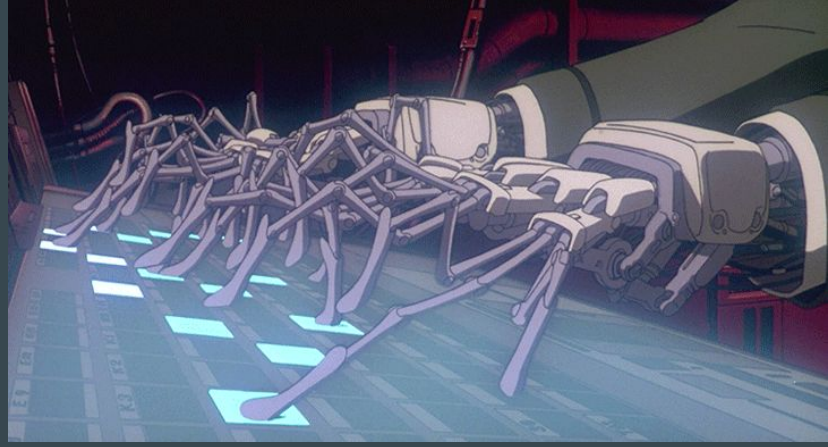
- Check caller privilege/identity of before response

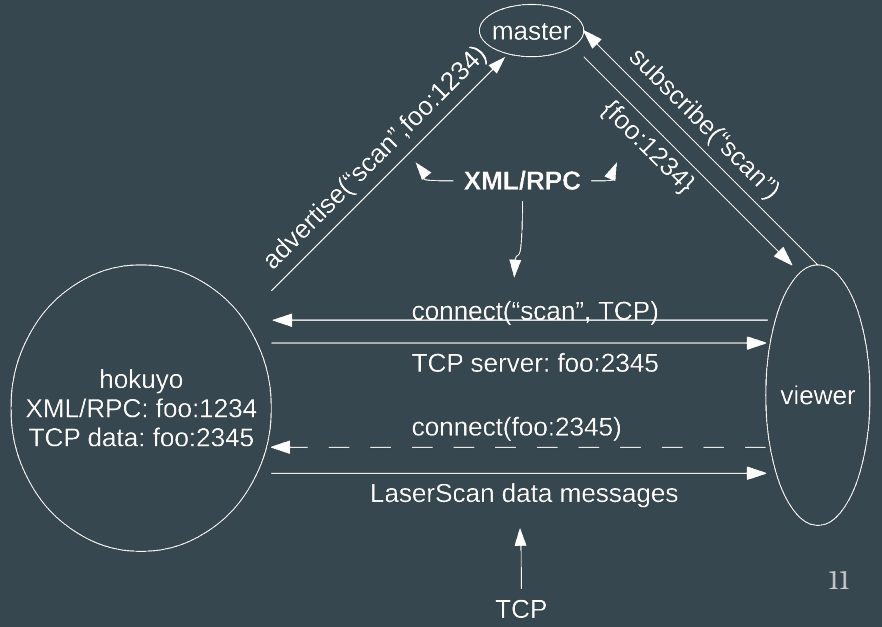
Tests and code coverage

- Unit Test all the things!

Abstract security modules as plugins

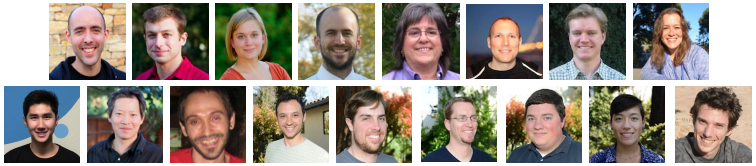
- Allow user to define custom policy evaluation plugins





A tip of the hat to OSRF & CogRob

Open Source Robotics Foundation



"...to support the development, distribution, and adoption of open source software for use in robotics research, education, and product development."

DARPA NASA NSF Sandia National Laboratories
YUJIN ROBOT BOSCH rethink robotics. ROS Industrial consortium
QUALCOMM ATAP SICK Sensor Intelligence.

CONTEXTUAL ROBOTICS INSTITUTE
UC San Diego

"...to advance contextual robotics through relevant grand challenge research, to educate and train students who are prepared to catalyze future developments in robotics; and to provide the talent and innovation to establish San Diego as a leading robotics hub."

Resources

SROS Documentation:

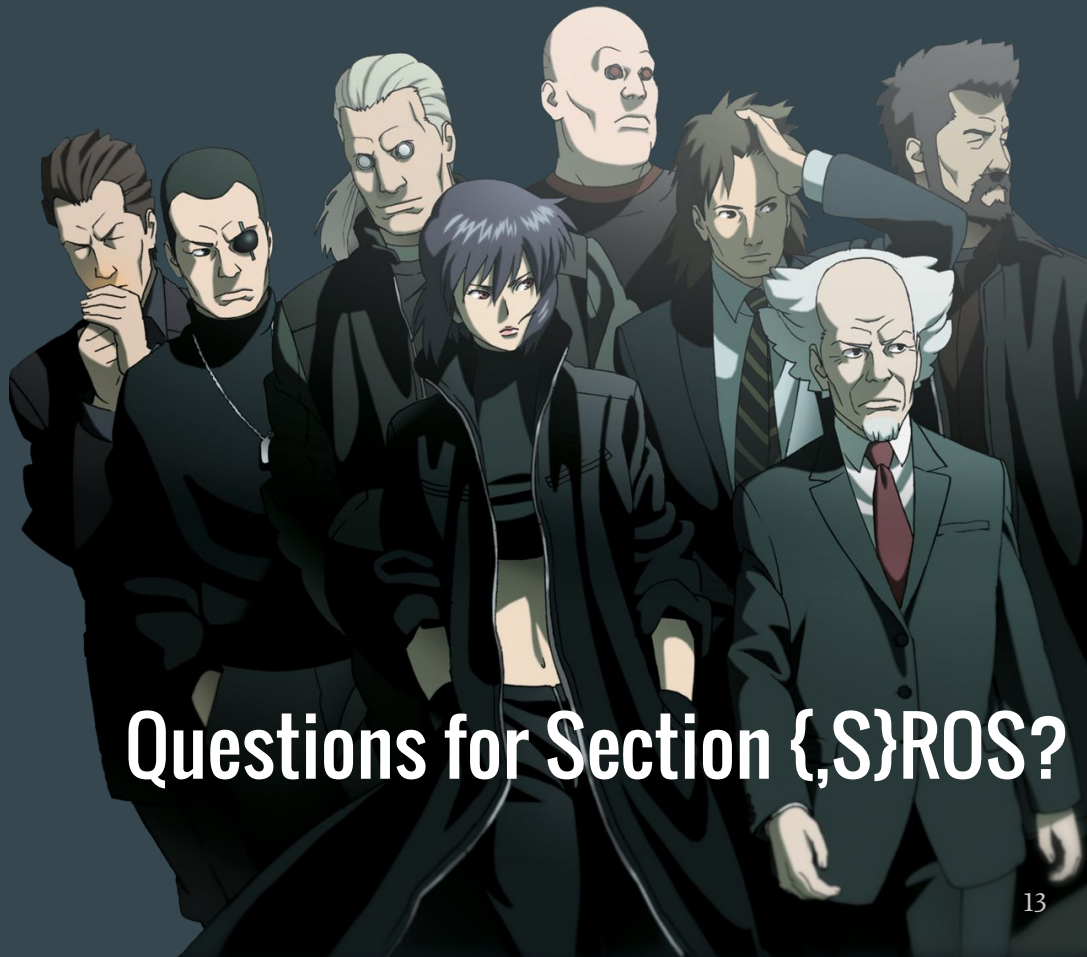
- wiki.ros.org/SROS

SROS Docker Image:

- hub.docker.com/r/osrf/sros

More about me:

- about.me/ruffin



Questions for Section {,S}ROS?