Combining ROS with seL4 for Trustworthy Autonomous Systems

Cynthia Irvine

Naval Postgraduate School

September 2019
A Reason to Care

Subversion

The covert and methodical undermining of internal and external controls over a system lifetime to allow unauthorized or undetected access to system resources and/or information. [1]
Early Subversion Experiments

Karger and Schell (1974)

- Early proponents of trustworthy systems
- Conducted vulnerability analysis of Multics [2]
- Described early episodes of subversion
- Software Trapdoor in Multics
  “Upgrade” to fielded systems
- Possible firmware or hardware “upgrade”
- Both “unexpected”
  Sailed through configuration control board
- Required “damage control”
Subversion Popularized

Elaborated on subversion suggestion made in Karger and Schell paper: Subvert the compiler.

Ken Thompson


- Compiler contained two artifice constructors
  - Compiler artifice installer
  - Unix artifice installer
- When compiler compiles compiler source, both installers are installed
- When compiler compiles Unix source, Unix artifice is installed

Was this real? Evidence of subversion backed out of system? Maybe.
UAV Swarms

Advanced Robotic Systems Engineering Laboratory (ARSENAL), a team of students at the Naval Postgraduate School in Monterey, California, successfully launched a swarm of 50 drones

NPS Zepher II

Basic system

- Developed mostly using COTS hobby equipment
  - Wingspan: 145 cm
  - Takeoff weight: 2.5 kg
  - Flight endurance: 50 min
  - Cruise speed: 18 m/sec

- COTS used for
  - Flight
  - Avionics
  - Navigation
  - Communications
Deliberative Planning and Control System
- ODroid companion computer
- Ubuntu 14.04

Implemented as independent ROS nodes
- Controlled by separate swarm control node

Inter-component communication on companion computer relies on ROS services & message topics

Autopilot-bridge node provides direction to Pixhawk Autopilot using MAVLink (Micro Air Vehicle Link) protocol
- Serial link

Swarm behavior implemented using ROS
ARSENAL Companion-computer C2 Architecture

Diagram shows network messages flowing through different components:

- Network Messages
  - network
    - set_subswarm
    - run_behavior
    - pause_behavior
    - swarm_control
      - set
      - run
      - pause
      - user-defined behavior
  - MAVLINK Messages
    - payload_wpt
    - recv_pose
    - recv_red_pose
    - red_tracker
      - red_uav_states
    - swarm_tracker
      - swarm_uav_states

Key:
- ROS Message Topic
- ROS Message Topic
- ROS Service
Simulation System

- Enhanced Software-in-the-Loop (SITL) simulation system [5]
- Development and testing of single- and multi-UAV algorithms
- Realistic testing with actual vehicle software in rigorous, physically-based simulation

- Can simulate lossy communications environments as expected in real-world
Functional Success!!
But What is Wrong With This Picture?

No Consideration of Security

Just Trying to Make Them Function
- Communications
- Separation of Critical Processing
- Separation of High Integrity Data
- Audit
Hypothesis

seL4 could provide a foundation to support a better architecture.
Potential Uses of seL4 - Classroom

Problem
Students need exposure to real highly trustworthy systems

How can we make seL4 projects accessible to beginning and intermediate-level students?
Labtainers Objectives

**Consistent and Fair**
- Students execute labs in identical environments
- Instructors see consistent results and assess students on their work rather than environmental effects

**Parameterizable**
- Labs configured so each student’s work can be unique
- Labs are same level of difficulty for all students
- Expected results are parameterized to streamline grading

**Support for Automatic Assessment**
- Collected student work is parsed for specific outputs
- Tools may be developed to support assessment of particular aspects of exercise
Pre-packaged cyber labs

- Multiple Linux-based computers per lab
  - Spin up >10 computers in seconds on a laptop
  - Independent computers with virtual networks
- No provisioning/admin required of student
  - Consistent configurations via Docker containers
- Automated student assessment
  - Did student accomplish defined goals of the lab?
- Individualized labs to discourage “sharing”
Example Labtainers-based Distributed System

- Each container has own `init` & `systemd`
- Virtual networks; config files; packages . . .
- We instrument containers for artifacts
- Selected `stdin/stdout` & logs `timestamped`
More than 45 Existing Labs

- Software vulnerabilities, e.g., buffer overflow
- Networking, e.g., arp-spoof, DNS-spoof, snort
- Operations, e.g. ACLs, logging, authentication
- Web security, e.g., cross site scripting
- Cryptography, e.g., hashing, VPNs
- Industrial control systems (PLCs)
Roles in the World of Labtainers

**Designer**
SME who works with instructor to create labs based on learning objectives. Fine tunes and updates labs. May support auxiliary assessment tools.

**Instructor**
Defines learning objectives. Works with (or is) designer. Ensures student readiness to perform labs and conducts assessments.

**Student**
Performs lab exercise. Learns! Delivers results to instructor for assessment.
Student Experience

- Download a single fully provisioned Linux VM, e.g., into virtual box
- Issue command to run lab, e.g., *labtainers ssl* for the SSL lab
- Framework automatically pulls images
- Docker containers start per the lab design
- Student then sees virtual terminals & GUIs
- When done, student runs *stoplab*
- Directed to send zip file to instructor, e.g., LMS
Instructor Experience

- Assign lab (each has a lab manual)
- Collect zips, e.g., via LMS bulk download
- Run “gradelab ssl”
- Framework recognizes and unpacks LMS bulk
- Instructor sees table of students & goals
- Can to deep dive into specific student artifacts
Automated Assessment

- Lab designer defines measurable goals
- Might want to know: did `nmap` output indicate the correct ports were open during a single configuration of `iptables`?
- Identify specific artifacts, e.g., `stdout` of `nmap`
- Express intended values, e.g., `80 tcp/open`
- Support for temporal expressions
- Evaluate set of results delimited in time by configuration actions, e.g., running `iptables`
Managed as Project

- Open source project with GitHub presence
  - https://github.com/mfthomps/Labtainers
- Developed in Python
- Suite of regression tests for stability
- SimLab automated lab performance tool
- Simulate keystrokes for testing labs
- Our website: http://my.nps.edu/web/c3o/labtainers
- Initial development sponsored by NSF (DUE-1438893).
- Continued support from NSF (SaTC 1932950) and NSA.
Why Did I Tell You All of This?

Student exposure to highly trustworthy systems

- **Do not** need to convert them into formal methodists
- **Do want** them to understand benefits of high assurance systems
- Need a story, otherwise “What, me worry?”
- Lab exercise should not be torture
- Illustrate high assurance as an appropriate investment
- Can interesting seL4 projects be accessible to intermediate-level students?
Where is the convincing evidence that clearly demonstrates how a system founded on seL4 will save lives?


Questions

Cynthia Irvine
Naval Postgraduate School
irvine@nps.edu