RoboFuzz: Fuzzing Robotic Systems over Robot Operating System (ROS) for Finding Correctness Bugs

ESEC/FSE 2022

Seulbae Kim, Taesoo Kim
Motivation: Robotic systems are intriguing targets

- Robots: One type of Cyber-Physical Systems

TurtleBot3 “Burger”

[SENSING]
360° Laser distance sensor
Camera sensor
Sensor malfunction, spoofing attack, ...

[ACTUATION]
Dynamixel motors
Driver firmware errors, physical attacks, ...

[SOFTWARE]
Object recognition, Differential driving algo.
SW errors, vulnerabilities, ...

Driver firmware errors, physical attacks, ...
Challenges of testing robotic systems

1. **Systems are heterogeneous**
   - Factories, surgical robots, drones, autonomous cars, ...
   - Req) Need to focus on common properties

2. **Input space is humongous – as big as the physical world**
   - Robots operate in different conditions and environments
   - Req) Need to efficiently explore the search space

3. **Physical processes are noisy**
   - Sensors and actuators are noisy as they interact with the real world
   - Req) Cyber-physical discrepancy must be considered
Tackling challenge #1 (heterogeneity)

- Robot Operating System (ROS) is a de facto standard for robot development
Robot development using ROS

- ROS-based robotic application: node + topic + msg

- camera
  - publish
  - front_image
  - subscribe

- LiDAR
  - publish
  - laser_dist
  - subscribe

- object_detector
  - publish
  - subscribe

- objects
  - msg: (human, 5m)
  - subscribe

- path_planner
  - msg: (human, 5m)
  - subscribe

- hazard_logger
  - msg: (human, 5m)
The behavior of ROS-based systems can be summarized as the data (message) flow among distributed nodes.
Tackling challenge #2 (huge input space)

- Feedback-driven fuzzing to the rescue

Coverage feedback

coverage map

input

target system

buggy code
A need for a new feedback mechanism

General software programs

- Diverse, linear code paths
- More code paths $\approx$ more bugs found

Robotic systems

- Distributed system
- Behavior is driven by state changes in a loop, not by code paths
Semantic feedback for robotic systems

• Fundamental questions
  • How do we determine if the robotic system is approaching an undesirable state?
  • What indicates that the robotic system is being driven towards buggy states?

Semantics of the execution can be utilized as feedback!
e.g., Redundant sensor inconsistency as feedback

- The case of PX4 flight controller
  - Pixhawk 4 has two Inertial Measurement Units (IMU)
    - IMU consists of an accelerometer (measures linear acceleration)

IMU1: ICM-20689 of TDK
IMU2: BMI-055 of Bosch

Diff. of measured acceleration (stable operation) Average diff < 0.01 m/s²

Diff. of measured acceleration (crashed mission) diff = 76 m/s²
Tackling challenge #3 (noise)

• Key intuition
  • It is impossible to perfectly model the physical world
  • There will always be cyber-physical discrepancy to some degree
  • Let’s use the discrepancy to our advantage

We can simultaneously execute a robotic system in a simulator and in the real world
Tackling challenge #3 (noise): Hybrid execution

• States and events from both worlds are monitored, e.g.,
  • Phy: location (gps), collision (camera), battery state, motor temp.
  • Sim: location (gps), collision (script), non-existent, non-existent

Some states diverge, which is an important execution feedback

Some states exist only in the physical world

(a) PX4 drone in the real world
(b) PX4 drone in Gazebo simulator
Which types of bugs are we looking for?

- A new class of bugs in robotic systems: **correctness bugs**

Classic software bugs

Robotic correctness bugs
Which types of bugs are we looking for?

- A new class of bugs in robotic systems: **correctness bugs**

  - Violation of physical laws
  - Violation of specification
  - Cyber-physical discrepancy
Overview of RoboFuzz

Test Target -> System Inspector -> Node-Topic Graph

System Inspector -> Message Mutator

Message Mutator -> Hybrid Executor

Hybrid Executor -> Simulator

Simulator -> Real world

Feedback Engine

Feedback Engine -> BUG? [No]


BUG? [No] -> Oracle Handler
System Inspector

- Generates a node-topic graph
- Select a topic to inject mutated messages

Node-Topic Graph

- camera: publish to front_image
- LiDAR: publish to laser_dist
- object_detector: subscribe to laser_dist, publish to objects
- path_planner: subscribe to objects
- hazard_logger: subscribe to objects

Fuzz these nodes

Inject here
Message mutator

• Structure-aware mutation
  • ROS messages are structured

An example message

```plaintext
uint32 height  # image height
uint32 width   # image width
string encoding # encoding of pixels
uint8[] data    # actual matrix data of pixels
```
Hybrid executor

• Set up a pair of **simulated** and **physical** test beds
  • Identical environment
  • Robots subscribe to the same topic
  • Publish mutated messages to the topic
  • Both robots receive the message and take corresponding actions
Oracle handler

- Collects and merges states from hybrid execution
- Allows developers to declare and apply custom correctness oracles
- Reports if any violation is found
- See paper for our specialized oracles
  - for two ROS internals and four ROS-based robots
Feedback engine

• If no bug is found, calculates the feedback score
  • Using the semantic feedback metrics
    • e.g., redundant sensor inconsistency
  • Users can register custom feedback metrics
• Favorable inputs are enqueued
  • Further mutated in the subsequent fuzzing rounds
Evaluation

• Environment
  • Laptop machine running Ubuntu 20.04
  • Intel i7-8850H 2.6Ghz, 16GB RAM, Quadro P2000 Mobile GPU

• Six fuzzing targets
  • ROS 2 internals:
    • ① Type system (ROSIDL), ② Client library (rclpy/rclcpp)
  • ROS 2-based robots:
    • ③ PX4, ④ TurtleBot3, ⑤ MoveIt2, ⑥ Turtlesim
Overall effectiveness of RoboFuzz

- RoboFuzz fuzzed each target for 12 hours
- Found 30 new correctness bugs (25 acknowledged, 6 fixed)
  - ROS 2 Internal layers
    - 8 in ROSIDL (1)
    - 5 in rclpy/rclcpp (2)
  - Applications
    - 8 in PX4 drone (3)
    - 5 in TurtleBot3 (4)
    - 2 in MoveIt2 (5)
    - 2 in Turtlesim (6)

→ Utilized hybrid fuzzing for 3 & 4

→ affects any robot built upon ROS 2
Demo – TurtleBot3 spec. violation

RoboFuzz Demonstration

Bug #9
- Motor driver HW impl. doesn't match the spec.
- Maximum linear velocity is smaller than documented
  - Spec: 0.22 m/s, actual: 0.21 m/s

Maximum linear velocity
- Spec : 0.22 m/s
- Actual: 0.21 m/s

BUG: Achievable velocity is smaller than documented due to a float handling bug in motor driver

MANIFESTATION:
Simulated robot can move at 0.22 m/s
The physical robot cannot

https://youtu.be/MB5iCiYLBCI
Effectiveness of semantic feedback

- Fuzzing PX4 with and without semantic feedback for 12 hr
  - 9 bugs with semantic feedback
  - 2 bugs without feedback
Summary

• Targeted **correctness bugs** in ROS and ROS-based robots
• **Semantic feedbacks** are defined and registered to efficiently explore the input space
• Utilized **hybrid execution** model to collect and compare the states of both cyber and physical robots
• Found 30 new correctness bugs in multiple robotic systems
• Open-sourced at [https://github.com/sslab-gatech/robofuzz](https://github.com/sslab-gatech/robofuzz)
  • Artifact evaluated: available (✅), evaluated & reusable (✅)