Coverage-Guided Fuzz Testing for Cyber-Physical Systems

Sanaz Sheikhi  Edward Kim  Parasara S. Duggirala  Stanley Bak

ICCPS 2022

Stony Brook University  University of North Carolina at Chapel Hill
Motivation

CPS properties:
• Hardware and software space
• Complex protocols

Challenges:
• Does the CPS works correctly?
• How to generate test cases?
An automated software testing method injecting invalid, malformed, or unexpected inputs into a system to reveal bugs and vulnerabilities.
Fuzz Testing CPSs

Challenges of CPS fuzz testing:

- Continuous states,
- Inputs that change over time
CPSFuzz

• **Novel coverage notion** to evaluate fuzz testing methodology effectiveness for CPS.

• **Customized power schedule**: leverages coverage score to select promising inputs to find failures in new system states.

• **Customized mutation strategy**: reasons with the causal nature of a CPS.
CPS Execution Model

- Black-box simulator model:
  \[ f : X \times U \times W \rightarrow Y \]

- Black-box software controller:
  \[ g : Y \rightarrow U \]

- **Goal**: find external input sequences, \( w_0, w_1, \ldots w_T \), that cause errors
CPS Coverage Metric

Designe properties:

- Adding more events never decreases the metric
- Identical events do not increase the metric
- Similar events have a lower impact than dis-similar events

- Input: sensed states at events
- Output: scalar coverage score

\[ S : \text{Set}[Y] \rightarrow \mathbb{R} \]
CPS Coverage Metric

- **Objective Space Projection Function**: maps sensed state to a $o$-dimensional Euclidean space:
  \[ \mathcal{P} : Y \rightarrow \mathbb{R}^o \]

- **Objective Space Exploration Limits**: box bounds within the objective space:
  \[ B \in \mathbb{R}^{2o} \]

- **Kernel function**: measures the similarity of states in the objective space using $o$-dimensional normal distribution
  \[ \mathcal{N}(\mu, \sigma^2) \]

  - $\mu$: a point in the objective space of each event
  - $\sigma$: a fixed hyper-parameter
CPS Coverage Metric

**Metric computation:**

- Map each event to the objective space,
- Apply kernel functions to measure states similarity,
- Integrate the maximum of the kernels

\[
S(\text{Set}[Y]) = \int_B \max_{y \in \text{Set}[Y]} \mathcal{N}(P(y), \sigma^2)(b) \, db
\]
CPS Coverage Metric (Example)

Two Events (score: 2.0)

Three Events (score: 2.595)
CPSFuzz Architecture

- **CPS**: execution or simulation.
- **Seed**: initial inputs for mutation.
- **Population**: set of all inputs, and test results.
- **Seed Manager**: maintains the population.
CPSFuzz Architecture

- **Power Schedule**: selects a seed based on seeds’ energy.
- **Energy**: probability that a seed will be picked.
- **Mutator**: performs various operations on a valid seed.
CPSFuzz Power Schedule

- Problem of generic power schedule:
  - Waste testing cycles on duplicate seeds.
  - Deprives promising seeds.

- CPSFuzz solution:
  - Finds a subset of the objective state space with minimum CPS coverage score.
  - Picks a seed that improves the coverage of the subset.
CPSFuzz Mutator

- Problem of generic mutation:
  - Blind input modification
  - Fine-grained operations

- CPSFuzz’s Mutation:
  - Maps subset of state space to an interval in input sequence.
  - Employs coarse-grained mutations at control command level.
Evaluation

Case study:
- F1TENTH autonomous racing competition
- Stress test overtake maneuvers
- Perturb the adversarial agent behavior
- Interesting events: collisions

Comparison:
- Hypothesis
- Atheris
- Random approach
Evaluation

(a) CPSFuzz

(b) Random approach

**DBScan**: measuring uniqueness of failures by spatial clustering.
Evaluation

Median scores during five runs of test case generation, one million frames at each run

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th># Test cases</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPSFuzz</td>
<td>361</td>
<td>21.06</td>
</tr>
<tr>
<td>Atheris</td>
<td>635</td>
<td>3.28</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>562</td>
<td>13.54</td>
</tr>
<tr>
<td>Random</td>
<td>499</td>
<td>16.34</td>
</tr>
</tbody>
</table>
Conclusion

- CPSFuzz: a framework for fuzz testing CPSs
- Notion of objective state space coverage
- [https://github.com/sanazsheikhi/CPSFuzz/tree/master](https://github.com/sanazsheikhi/CPSFuzz/tree/master)