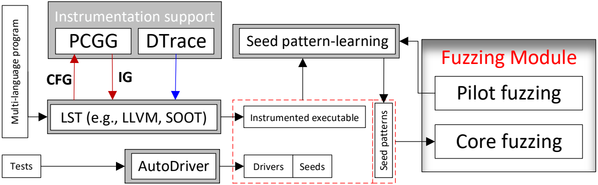
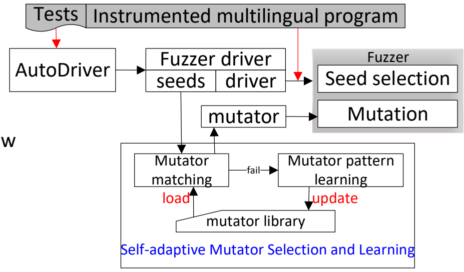
Overview



CFG: control flow graph
IG: instrumentation guidance
PCGG: path-coverage guarantee guidance
LST: language-specific tool

1 Overview



Tests

Instrumented multilingual program

AutoDriver

Fuzzer driver

seeds

driver

Fuzzer

Seed selection

mutator

Mutation

Mutator matching

fail

Mutator pattern learning

load

update

mutator library

Self-adaptive Mutator Selection and Learning

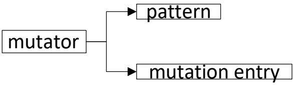
AutoDriver: a tool for generating seeds and drivers automatically.

2 Self-adaptive Mutator Selection

Basic Ideas

1. structural pattern: identify crucial bytes through pilot fuzzing, use these crucial bytes to construct structural pattern. (e.g., Python list, URL, JSON).
2. character pattern: identify the character type of bytes through character analysis to construct character pattern. (e.g., the valid input is the displayable character set).
3. Mutator learning: Build a tool to automatically identify the two patterns, Then construct a corresponding mutator following the three strategies:
   a> mutate following the patterns
   b> mutate against the patterns
   c> mutate randomly
4. Mutator selection: Select a mutator through pattern matching on the seeds and bind it to the seed set and driver.
5. Start fuzzing with the selected mutator.

Mutator definition



mutator

pattern

mutation entry

Mutator Library

A list of mutators

Mutator Library Construction

Manually Construction

E.g.,
URL: https://github.com/google/AFL

Pattern: http[s]?://[a-zA-Z0-9\.\-/]+
Mutation: -> syntax-maintained mutation
-> length-varied mutation
-> random mutation

Automatic Pattern Learning

1. Pilot fuzzing:
   a> Define threshold P as the least length of valid path .
   b> Mutate the byte one by one and get the length of execution path as *p.*
   c> iff p <= P, consider the byte as crucial byte and further identify its range. Done? d> : a>
   d> Collect the crucial bytes to construct pattern

2. Character analysis:
   a> Analyze the bytes one by one and get the char types
   b> Collect all char types of the tests as character pattern

3. Construct mutator:
   a> Construct template according to the two patterns
   b> Construct mutator based on the template
   c> Store the learned mutator to the mutator library