The 4th International KLEE Workshop on Symbolic Execution (15–16 April 2024)

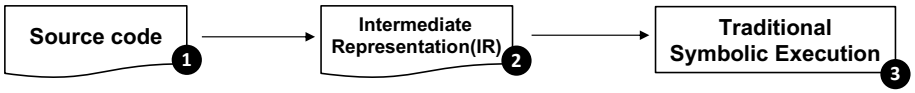
FastKLEE: Faster Symbolic Execution via Reducing Redundant Bound Checking of Type-Safe Pointers

Haoxin Tu, Lingxiao Jiang, Xuhua Ding, and He Jiang

SMU SINGAPORE MANAGEMENT UNIVERSITY

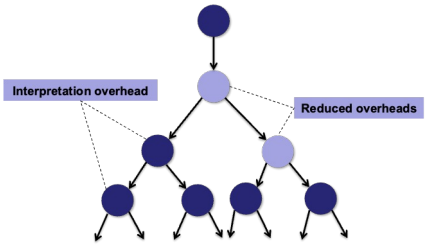
INTRODUCTION

- Bugs in software are inevitable and many tools (e.g., symbolic execution engines) are designed to detect bugs effectively and efficiently.
- Intermediate Representations (IR)-based symbolic execution engines (e.g., KLEE) are prevalent and widely used for software testing.
- The efficiency of symbolic execution is seldom explored in the literature.
- **Objective:** we aim to design and implement a fast symbolic execution engine to improve the efficiency of symbolic execution.

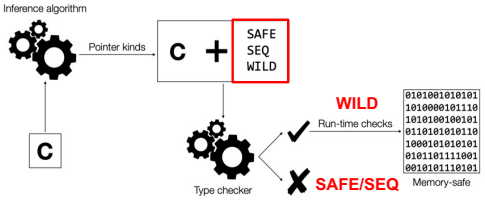


MOTIVATION

- **Observations**
  - ➢ The number of interpreted instructions tends to be huge
  - ➢ Overheads in current symbolic execution: all instructions are equal



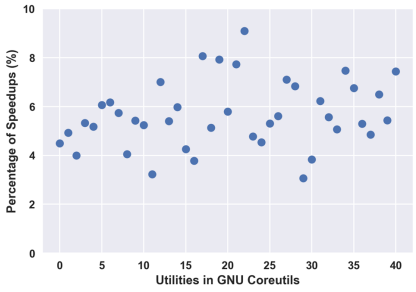
- **Key Insights**
  - ➢ Only a small portion of memory-related instructions need bound checking
  - ➢ Reduce interpreting overhead of most frequently interpreted ones (i.e., load/store instructions)
  - ➢ Inspired by *Type Inference* system [2]



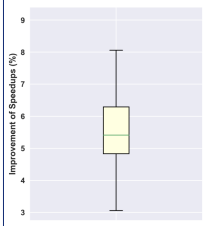
PRELIMINARY RESULTS

Evaluation Criteria

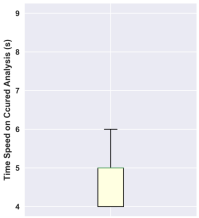
- ➢ **Speedups**
  - ➢ the time spent on exploring the same number of instructions



Scatter plot of the improvement in speedups



Box plot of the improvement in speedups



Box plot of the time spent on type inference

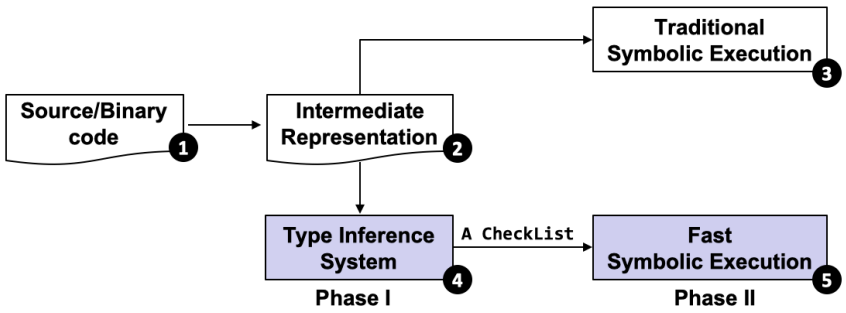
PROPOSED SOLUTIONS

Phase I: Introduce a type inference system to classify memory-related instruction types

- ➢ Unsafe memory instructions will be stored in CheckList

Phase II: Conduct customized memory operation for fast symbolic execution

- ➢ Only perform checking for *unsafe* memory instructions during interpretation



High level idea of FastKLEE

FUTURE WORK

- ➢ **Motivation**
  - ➢ Current heuristics for path exploration are not vulnerability-oriented
- ➢ **Main idea**
  - ➢ Use the unsafe-pointer to guide the path exploration
  - ➢ Extend FastKLEE for unsafe-pointer-oriented path exploration

Contact

Email: haoxintu.2020@phdcs.smu.edu.sg
GitHub: https://github.com/haoxintu
Twitter: @tuhaoxin

ACKNOWLEDGEMENT & REFERENCES

This research/project is supported by the National Research Foundation, Singapore and the National Satellite of Excellence in Trustworthy Software Systems (NSoE-TSS) award number NSOE-TSS2019-04.

[1] Haoxin Tu, Lingxiao Jiang, Xuhua Ding, and He Jiang. "FastKLEE: Faster Symbolic Execution via Reducing Redundant Bound Checking of Type-safe Pointers." In ESEC/FSE, pp. 1741-1745. 2022.
[2] George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. CCured: Type-safe Retrofitting of Legacy Software. ACM Trans. Program. Lang. Syst. 27, 3, 477–526.

SCHOOL OF COMPUTING AND INFORMATION SYSTEMS