Introduction of Configuration Awareness Into an Evaluation Framework for Fuzz Testing

Arnab Dev, Meah Ahmed Tahmeed, and Shiyi Wei

Department of Computer Science, University of Texas at Dallas

INTRODUCTION

Fuzzing is a key tool used to reduce bugs in production software

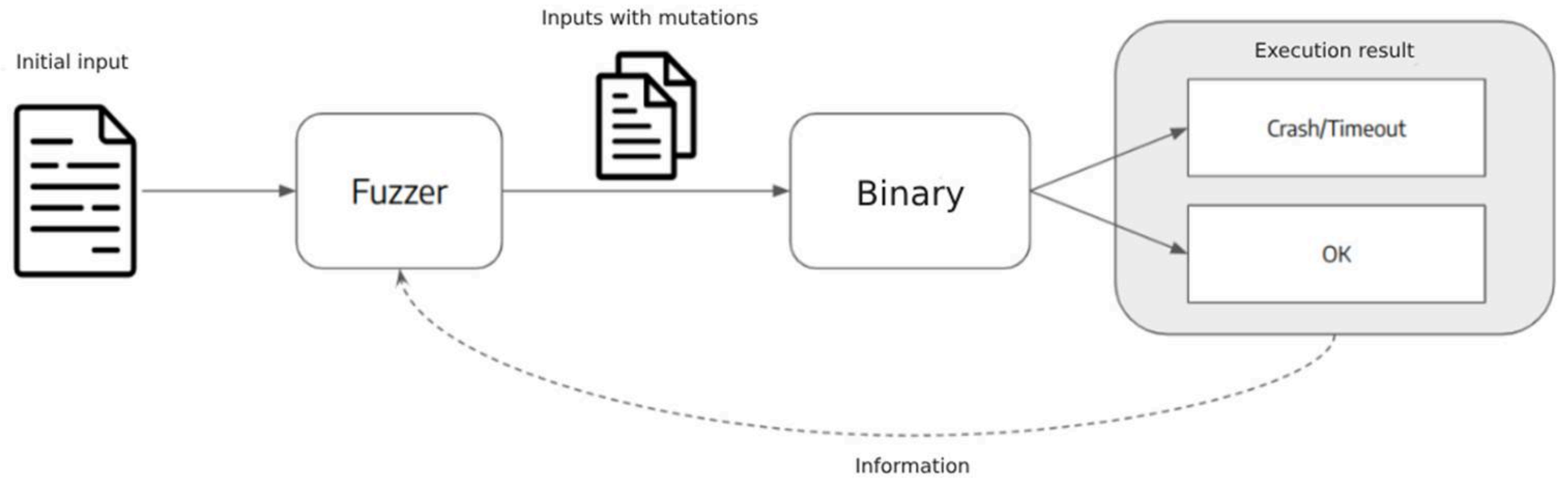


Fig. 1 The fuzzing process

Configuration-aware fuzzer actively analyzes and varies configuration options during fuzzing

The Problem
Existing fuzzing benchmarks, most notably FuzzBench, evaluate fuzzers on a single, fixed configuration.

Our Solution
We extend FuzzBench to support configuration-aware fuzzing by enabling runtime configuration control.

Motivation
The growing body of evidence demonstrating that configuration-aware fuzzers consistently outperform traditional fuzzers in both bug discovery and code coverage

Our Approach

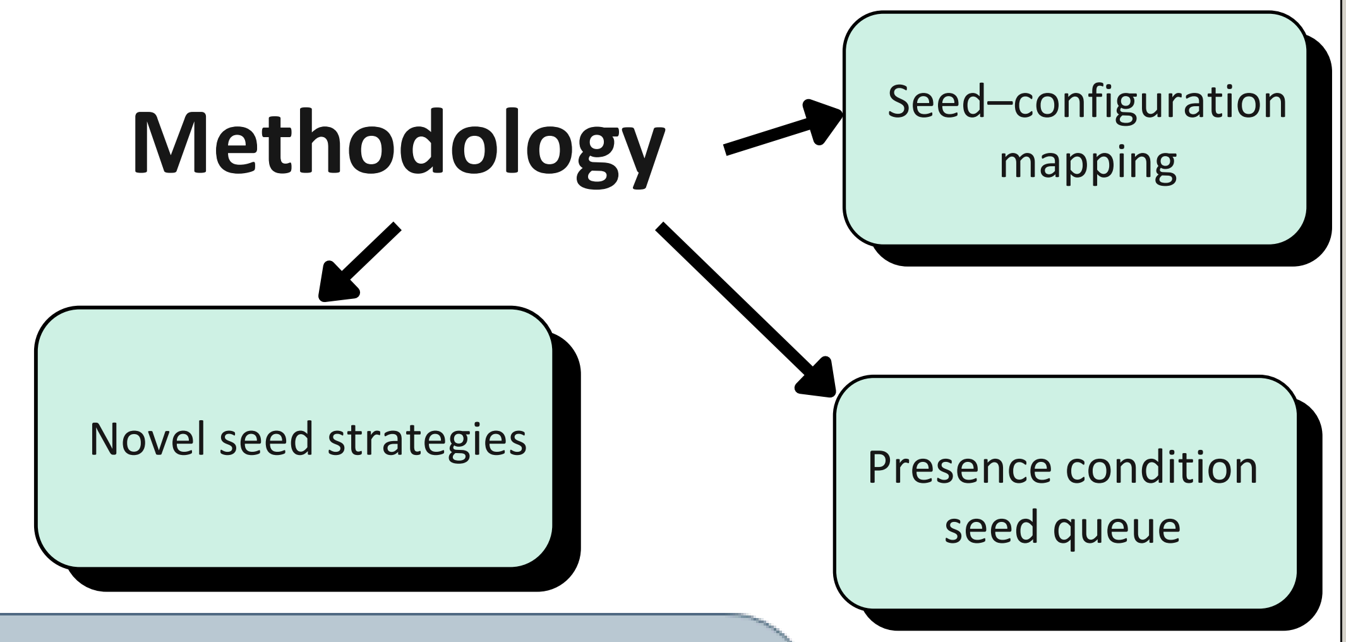
Modifications to existing framework → **Tests Configuration Aware Fuzzers** → **Produces relevant reports**

Key Goal
Allow fuzzers that manipulate configurations during fuzzing to be evaluated under standard, reproducible conditions.

Variability Aware Fuzzer (VAFuzz)

Overview
Development of **Variability-Aware fuzzer** to establish importance of the framework

A type of a **configuration aware fuzzer** where the fuzzer it explores multiple feature combinations to uncover bugs hidden across the configuration space and hence enhance configuration space exploration.



Methodology

- Seed–configuration mapping
- Novel seed strategies
- Presence condition seed queue

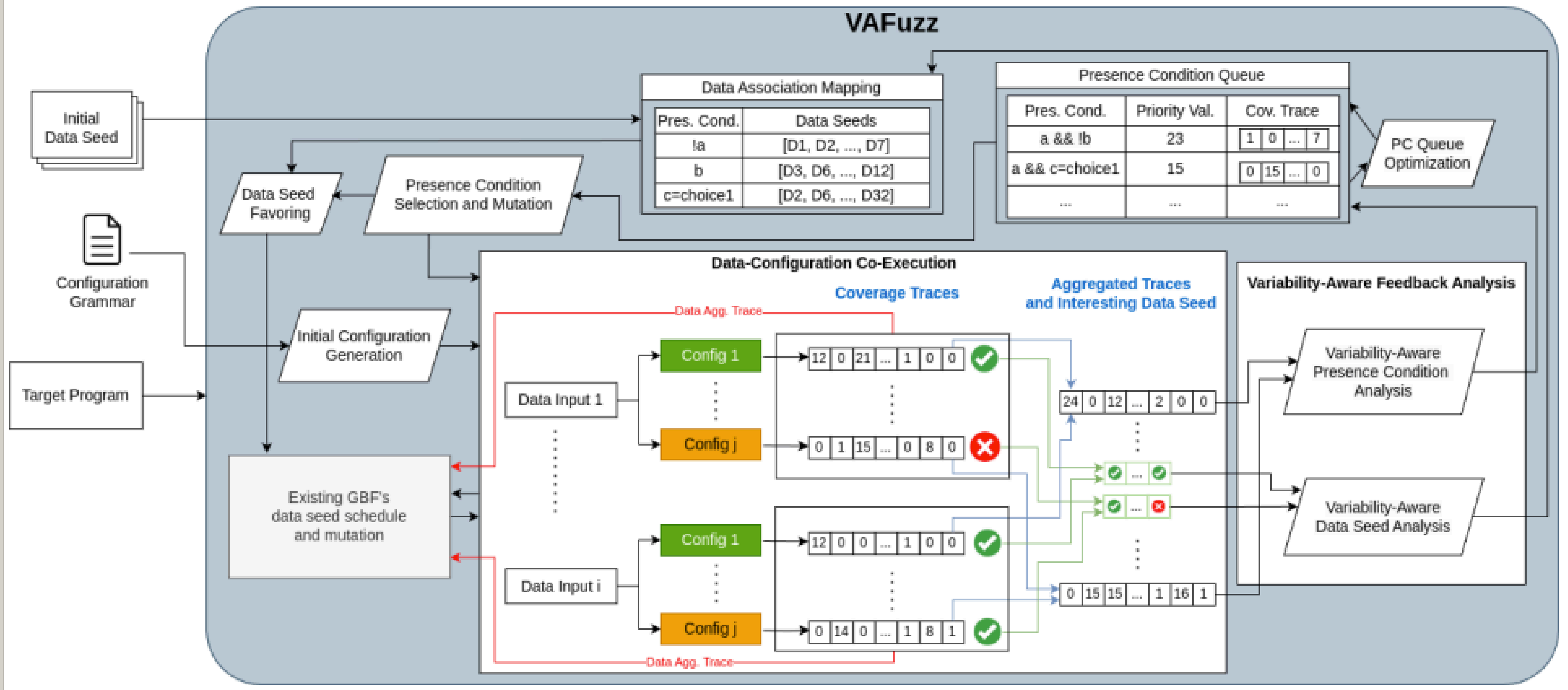


Fig. 2
Workflow of VAFuzz

Results

Programs	VAFuzz		ZigZagFuzz		AFL++	
	Coverage	Bugs	Coverage	Bugs	Coverage	Bugs
cjpeg	4162	0	3680	0	1143	0
jpegtran	5292	0	2254	0	1302	0
fax2ps	2275	1	1903	1	1835	0
fax2tiff	1586	0	1485	0	821	0
tiff2pdf	2087	0	1592	0	1075	0
tiff2ps	1522	0	1380	0	1179	0
tiffcp	2648	0	207	0	1050	0
gif2png	414	2	413	2	320	2
nasm	4268	3	3388	0	750	0
ndisasm	877	1	808	0	385	0
nm	291	2	287	0	101	0
objdump	2951	7	2652	0	1302	0
xmlcatalog	4271	0	3821	1	1778	0

Table 1 Coverage and bug detection results for different fuzzers

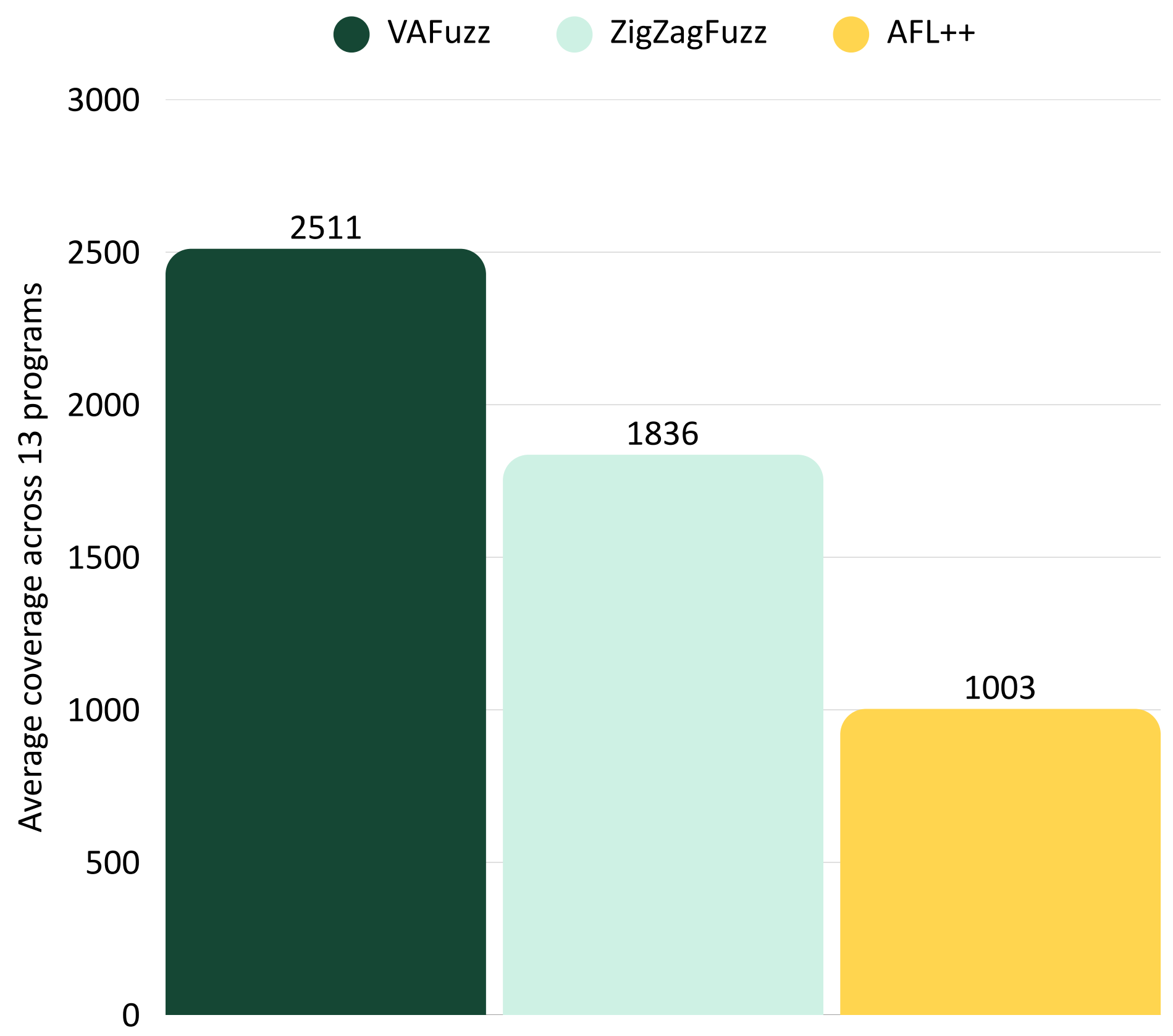
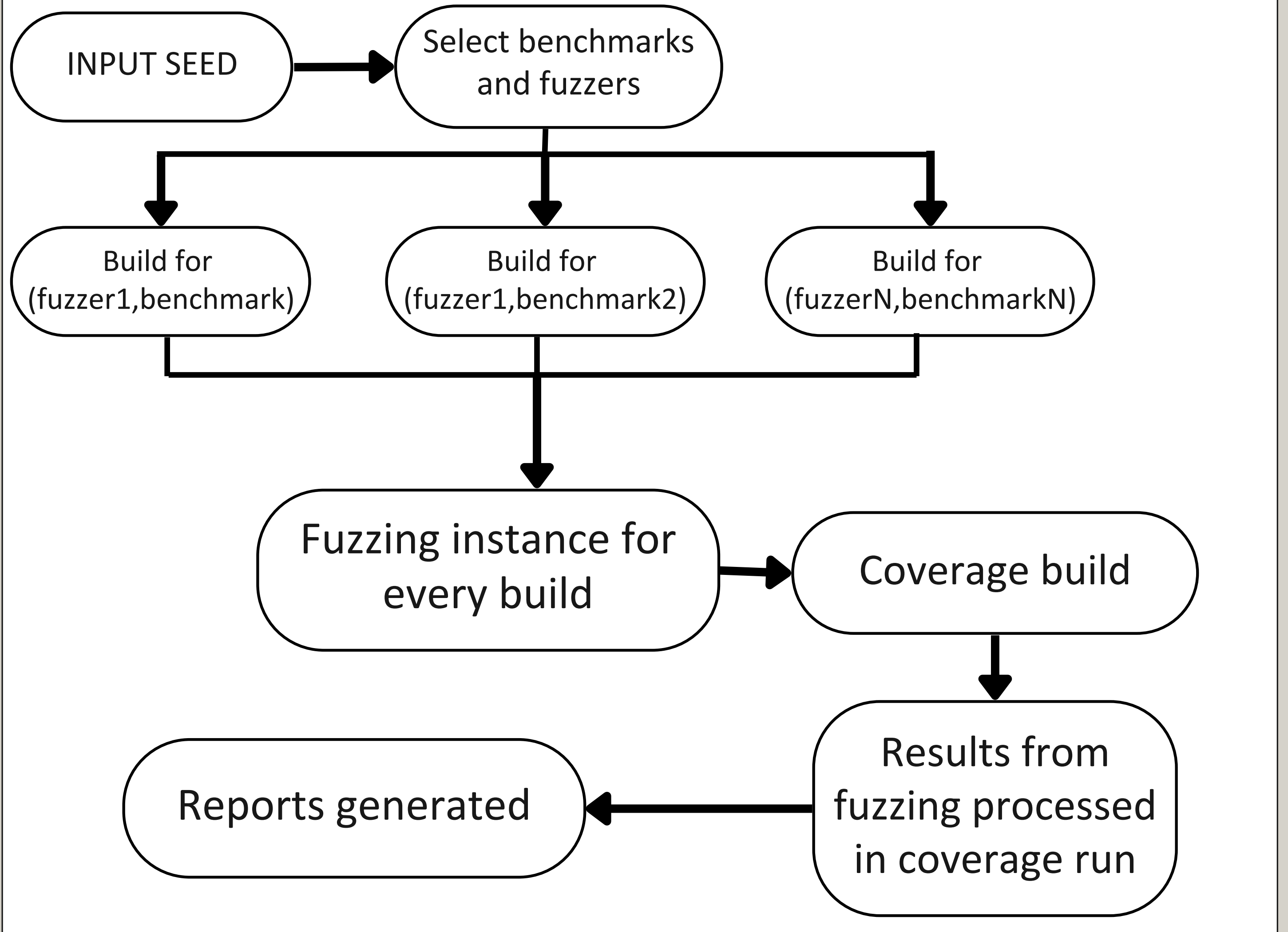


Fig. 3 Average coverage of the 3 tested fuzzers

Conclusion
VAFuzz and ZigZagFuzz (configuration-aware fuzzers) outperforms AFL++ (traditional non-configuration-aware fuzzer)

INTEGRATION OF CONFIGURATION AWARENESS INTO THE FRAMEWORK



INPUT SEED → Select benchmarks and fuzzers

Build for (fuzzer1,benchmark)
Build for (fuzzer1,benchmark2)
Build for (fuzzerN,benchmarkN)

Fuzzing instance for every build → Coverage build

Results from fuzzing processed in coverage run → Reports generated

Fig. 4 Detailed design of the evaluation framework

Key Changes

- Benchmarks with targets which changed from LibFuzzer harness and Clang to AFL++ compiler to allow compatibility with configuration aware fuzzers

Future work

Completing the implementation with full functionality including multiple configuration aware fuzzers and the option to add a new configuration fuzzers to be tested on a large benchmark set with the choice of compile time strategy

Contact

Arnab Dev
arnab.dev@utdallas.edu