# Toward Facilitating Root Cause Localization in Fuzzing

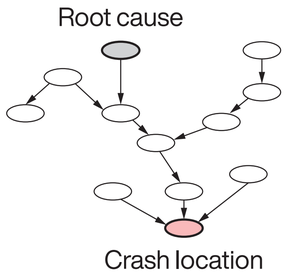
Katsunori Aoki
The University of Tokyo
aoki@os.ecc.u-tokyo.ac.jp

Takahiro Shinagawa
The University of Tokyo
shina@ecc.u-tokyo.ac.jp

Abstract:

## 1 Background

- Fuzzing is a method to find vulnerabilities
  - It finds a lot crashes automatically by generating inputs
- Root cause analysis is difficult
  - Fuzzers only reports crash location
  - Root cause is often far from crash location




## 2 Previous Root Cause Analysis

- Symbolic execution [1]
  - Collect conditions to trigger crash during execution
  - 👎 Path explosion problem 😡
- Statistical Crash Analysis [2]
  - Compare behavior of predicates using similar inputs
  - 👎 Binary-level information and high false positive rate 😡

## 3 Human friendly data flow graph (DFG)

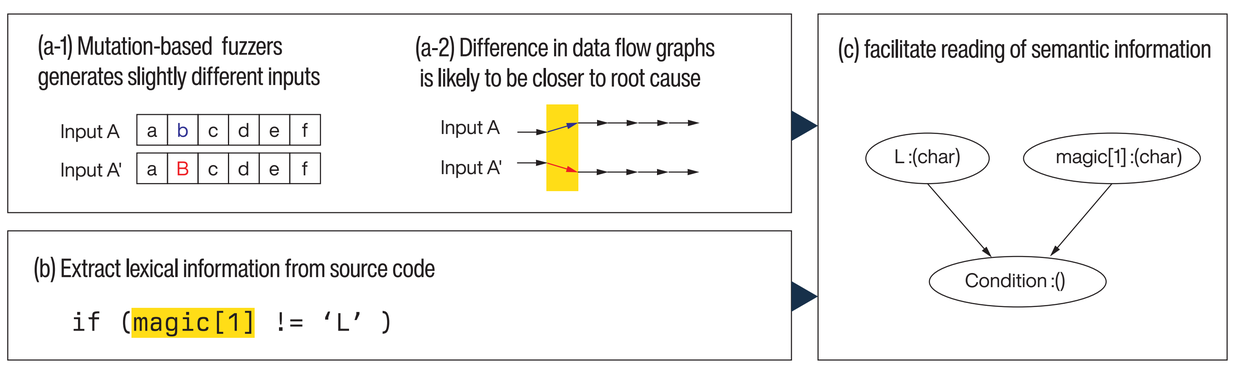
**Our goal**:

Quickly and easily understand root cause

**Key ideas:**

(a) Difference of DFG between crashed and normal execution

(b) Adding lexical information at the source code level




## 4 Implementation

- **Lexical information extraction:** source code instrumentation using LLVM
- **Data flow tracing:** created ourself from scratch

## 5 Preliminary experiments

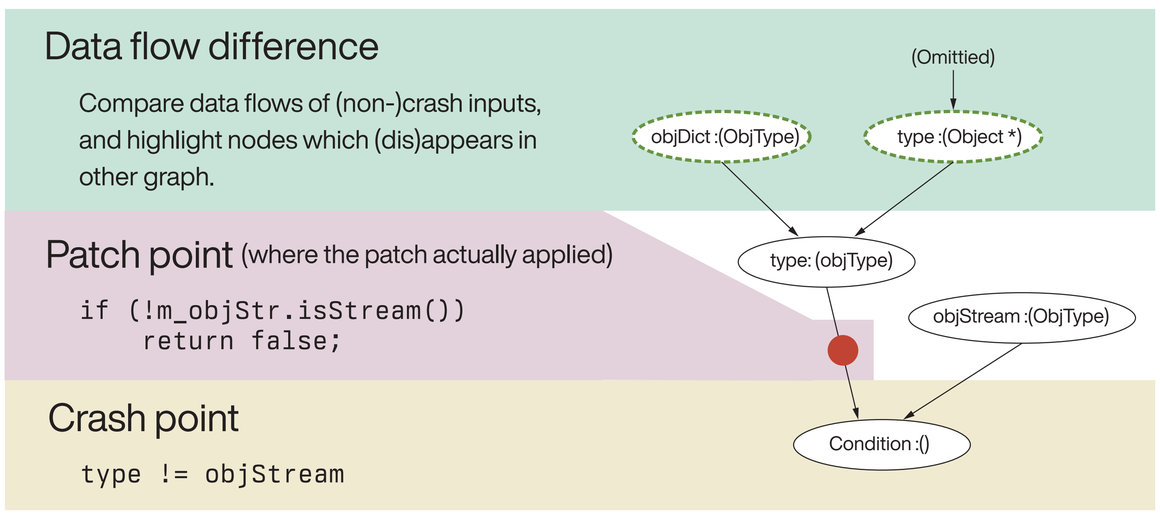
**Evaluation dataset:**

- Magma [3] : Reproduces known software vulnerabilities
  - Provides (non-)crash inputs as artifacts

**Evaluation method:**

- Measure distance between data flow difference and patch point

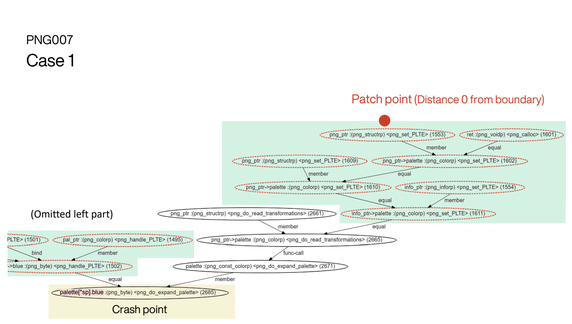
**How to read graph:**




**Result (4 cases):**

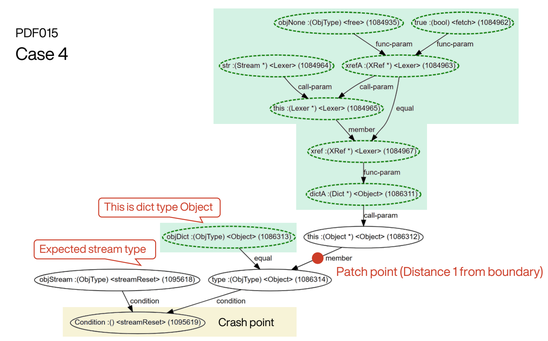
- (Distance 0) Case 1, 2: patch point was control flow governing the boundary of difference
- (Distance 0) Case 3: patch point was tangent to the boundary
- (Distance 1) Case 4: a sanitization patch was applied to a point at distance 1 from the boundary

PNG007
Case 1




Case 1: libpng PNG007

PDF015
Case 4




Case 4: popper PDF015

- In all cases,
  - no need to follow data flow far from the boundary
  - patch point located near the boundary

## 6 Future work

- Evaluate our approach in more crash cases
- Evaluate whether our approach eases root cause analysis compared to manual analysis and previous studies

**References**

[1] C. Yagemann et al. ARCUS: Symbolic Root Cause Analysis of Exploits in Production Systems. In Proc. 30th USENIX Security Symposium, Aug. 2021.

[2] T. Blazytko et al. AURORA: Statistical Crash Analysis for Automated Root Cause Explanation. In Proc. 29th USENIX Security Symposium, Aug. 2020.

[3] A. Hazimeh et al. Magma: A Ground-Truth Fuzzing Benchmark. Proc. ACM Meas. Anal. Comput. Syst., 4(3), Dec. 2020.