A Qualitative Analysis of Fuzzer Usability and Challenges

Yunze Zhao, Wentao Guo, Harrison Goldstein, [2]Daniel Votipka, [3]Kelsey Fulton, Michelle Mazurek

University of Maryland, [2] Tufts University, [3]Colorado School of Mines

Security & People At Mines

SECURITY, PRIVACY, PEOPLE

Introduction / Motivation

Fuzzers discover bugs by generating random inputs to trigger failures.

Some studies have examined fuzzer usability, yet they often focus on constrained or synthetic environments

Fuzzers usability in practice remains underexplored

By interviewing experienced fuzzing tool users, we aim to inform the design of fuzzers that are more usable in practice.

Research Questions / Method

RQ1: What specific challenges do users face across the lifecycle of a fuzzing campaign?

RQ2: What strategies do practitioners use to address or work around these challenges?

RQ3: What improvements to fuzzing tools would better support practical adoption and real-world workflows?

We conducted **18 semi-structured interviews** with experienced practitioners who had hands-on fuzzing experience in both academia and industry.

Recruitment

We recruited participants from top security conference papers, GitHub issues, fuzzing communities on social media (e.g., Discord), snowball sampling, and personal connections.

Challenge

Workaround

Suggestion

Mental Models

Prepare Target

Monitor Campaign

Inspect Crashes

Highlighted Findings

Most participants treat fuzzers as **black boxes**

Most participants learned fuzzing through fragmented, self-guided learning.

Despite this, Participants recognize fuzzing as a powerful technique for uncovering deep and unexpected bugs

Setup is fragile and ad hoc. Harness creation and target selection are hard.

Trial-and-error setup, reuse of test files, scripts to simplify harnessing, use of containers.

Auto-generate harness scaffolding, recommend fuzzing targets, validate configs before execution.

Unclear feedback during fuzzing—no way to tell if it's making progress or stuck

Monitor logs manually, build custom scripts, restart if no crashes for a while.

Visualize coverage growth, highlight mutation lineage, add alerts for stalled campaigns.

Too many duplicate or low-value crashes, unclear root causes, poor prioritization.

Manually cluster crashes, grep through logs, run inputs through external analyzers or scripts.

Smarter crash grouping, root-cause hints, customizable triage views, integrated reproduction tools.

This study was approved by UMD Institutional Review Board

U.S. Department of Defense