



# When Should We Stop Testing?

## - Fuzzing from the perspective of statistics

Seongmin Lee and Marcel Böhme

### FUZZING: SOTA SOFTWARE TESTING METHOD

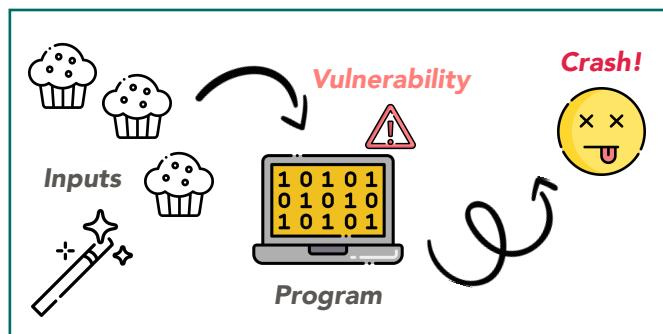


Fig. 1 Basic scheme of fuzzing

Given a program that might contain a vulnerability, **fuzzing**

1. generates lots of different inputs,
2. runs the program with those inputs,
3. and checks if a crash occurs based on the vulnerability.

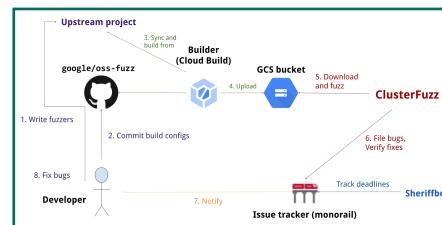


Fig. 2 OSS-Fuzz: open-source fuzzing infrastructure

### [Fuzzing is highly practical]

OSS-Fuzz found 10,000 vulnerabilities and 36,000 bugs across 1,000 open-source projects. (~August 2023)

### Fuzzing workshop 2024



Co-located with ISSTA'24 @ Vienna, Austria

EDFuzz: A Web API Fuzzer for Excessive Data Exposures  
API often transmits far more data to client applications than they need, and in the context of web applications, often do so over public channels. This issue, termed Excessive Data Exposure (EDE), was DNSP's third most significant API vulnerability in 2023. In investigating the problem, it is explicit that the vulnerability does not reveal remote such issues. This is interesting as the problem lacks an exploit to trigger the vulnerability and no exploit can be generated. In this work, we develop a metaprogrammatic relation to isolate that challenge and build the first fuzzing tool—that call EDFuzz—to systematically detect EDEs. EDFuzz can significantly reduce false negatives that occur during manual inspection and ad-hoc testing.

We tested EDFuzz against the only-nine application targets from the Alexa Top-100 and found 21,345 potential lines—highlighting our tool's broad applicability and scalability. In a more tightly controlled experiment of eight popular websites in Austria, EDFuzz achieves a high true positive rate of 98.6%, with minimal configuration, guaranteeing our tool's accuracy and efficiency.

Fig. 3 Academic Interests in fuzzing

### [Fuzzing is actively investigated]

Typical research topics in fuzzing:

- How can we make fuzzers **smarter** at generating inputs?  
E.g., Magic value, Symbolic analysis, etc.
- What/Where else can we apply the fuzzing?  
E.g., Smart contracts, Web applications, etc.

### However, when should we stop the fuzzing campaign?

Software testing is always a trade-off between time/resource spent vs. how secure the software is.  
To answer the question, we need to **extrapolate how the fuzzing will proceed**.

### KEY IDEA: FUZZING IN TERMS OF STATISTICS

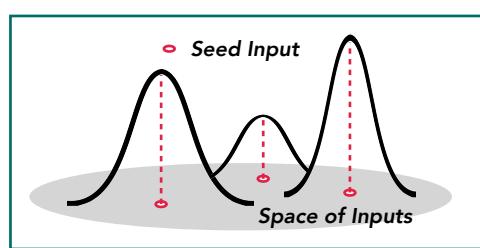


Fig. 4 Blackbox fuzzing as a stochastic process

"If the fuzzing is a **random sampling process of a fixed distribution**, we can **statistically extrapolate** the future of fuzzing."

- Coverage rate  $U(t)$ : the expected number of newly tested elements in  $(t+1)$ -th data point.
- Extrapolator of coverage rate  $U(t+k)$ :

$$\hat{U}(t+k) = \hat{\Phi}_0 \left[ 1 - \left( 1 - \frac{\Phi_1}{t \hat{\Phi}_0 + \Phi_1} \right)^{k+1} \right], \text{ where } \hat{\Phi}_0 = \frac{t-1}{t} \frac{\Phi_1^2}{2\Phi_2}$$

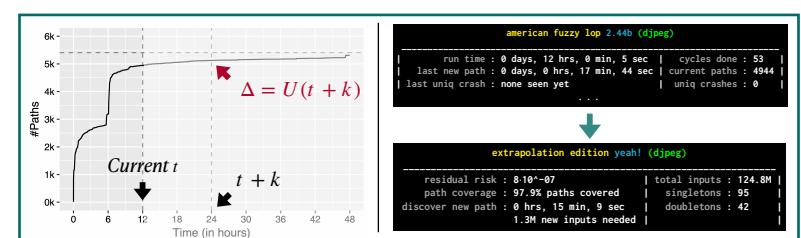


Fig. 5 Coverage rate extrapolation and its expected impact

### ADVANCED: GREYBOX FUZZING

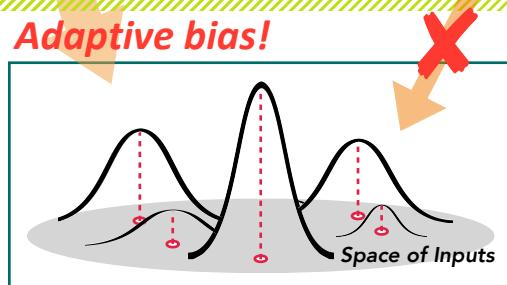


Fig. 6 Greybox fuzzing as a stochastic process

The sampling distribution changes as time goes on in a Greybox fuzzing. Adaptive bias

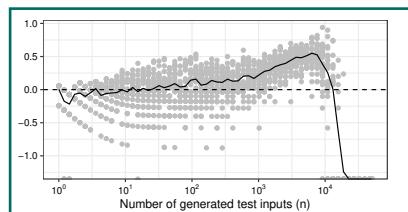


Fig. 7 Effect of adaptive bias on testing efficiency in greybox fuzzing

### TWO INSIGHTS TO HANDLE ADAPTIVE BIAS

#### Microscopic view

1. While the entire greybox fuzzing campaign is dominated by the **adaptive bias**.
  2. The local region of the campaign has a **less adaptive bias**.
  3. The near future is **predictable**!
- 

#### Macroscopic view

**"The adaptive bias of the greybox fuzzing is predictable!"**

Its changes are not random but have a pattern:

1. The change occurs when a new input that increases the coverage is found.
  2. The input is added to the seed corpus.
  3. The inputs around that new input are sampled.
  4. (Cont'd)
- 

### MODEL SUMMARY

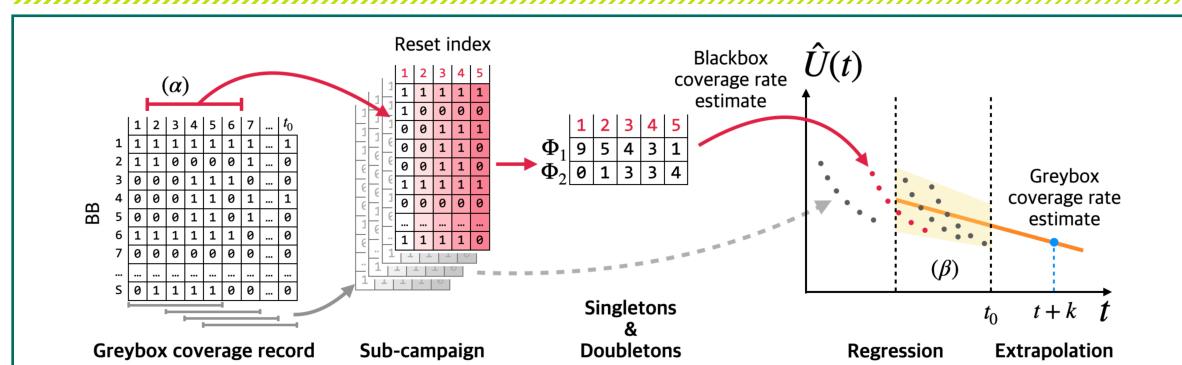


Fig. 8 Greybox coverage rate estimation model considering the adaptive bias

### TAKEAWAY

1. Statistical modeling can predict the future of the testing process and, more generally, any sampling-based optimization/searching process.
2. We bring two key insights to handle the adaptive bias, i.e., time-wise change of the distribution.
3. The regression model is more accurate in predicting the future of the greybox fuzzing campaign.

### EVALUATION

Q. How accurate is the **regression model considering the adaptive bias** compared to the **no-adaptive extrapolation model**?

- Subject program: five open-source C libraries
- Procedure: 1) run the greybox fuzzer until it gets  $t$  executions, 2) apply each extrapolator to extrapolate  $\hat{U}(t+k)$ , 3) run the greybox fuzzer for  $k$  more executions to get  $U(t+k)$ .

#### [Result]

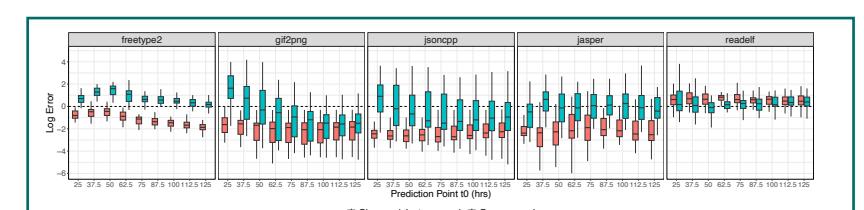


Fig. 9 Difference between  $\log(U(t+k))$  vs.  $\log(\hat{U}(t+k))$

- For 4 / 5 subjects,  $|Bias(\text{regress})| < |Bias(\text{no-adaptive})|$ , **at least one order of magnitude difference**.
- The average ratio  $U(t+k)/\hat{U}(t+k)$ : **No-adaptive model: 1.6 - 800 vs. Regression model: 1.17 - 7**