Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2019-IW-004

2019-IW-004

Energy Distribution Matters in Greybox Fuzzing

Lingyun Situ, Linzhang Wang, Xuandong Li, Le Guan, Wenhui Zhang, Peng Liu

International Conference on Software Engineering: Companion Proceedings 2019

2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

Energy Distribution Matters in Greybox Fuzzing

Lingyun Situ*, Linzhang Wang*, Xuandong Li*, Le Guan†, Wenhui Zhang‡ and Peng Liu‡
*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
†University of Georgia, Athens, GA, USA
‡Pennsylvania State University, State College, PA, USA
situlingyun@seg.nju.edu.cn, {lzwang, lxd}@nju.edu.cn, leguan@cs.uga.edu, wenhui@gwmail.gwu.edu, pliu@ist.psu.edu

***Abstract*—Existing energy distribution strategies of AFL and its variants have two limitations. (1) They focus on increasing coverage but ignore the fact that some code regions are more likely to be vulnerable. (2) They randomly select mutators and deterministically specify the number to mutator, therefore lack insights regarding which granularity of mutators are more helpful at that particular stage. We improve the two limitations of AFL's fuzzing energy distribution in a principled way. We direct the fuzzer to strengthen fuzzing toward regions that have a higher probability to contain vulnerabilities based on static semantic metrics of the target program. Furthermore, granularity-aware scheduling of mutators is proposed, which dynamically assigns ratios to different mutation operators. We implemented these improvements as an extension to AFL. Large-scale experimental evaluations showed the effectiveness of each improvement and performance of integration. The proposed tool has helped us find 12 new bugs and expose three new CVEs.**

***Index Terms*—GreyBox Fuzzing, Directed Fuzzing, Mutator Schedule**

I. Introduction

American Fuzzy Lop (AFL) [1] [5] is one of the most effective fuzzing tools to explore vulnerabilities. A lot of works improve AFL's abilities by maximizing code coverage. This is typically achieved by (1) improving feedback accuracy and granularity [4]; (2) enhancing mutation strategies (i.e. where and what to mutate) [3]; (3) providing high quality and diverse seeds [8]; (4) accelerating execution speed by utilizing hardware features (e.g Intel-PT) [7], and (5) balancing energy distribution (e.g., low-frequency and untouched path deserve more energy [2] [4] [6]).

Although it is shown that strategically distributing fuzzing energy could substantially enhance the effectiveness of fuzzing, such strategies have not yet been systematically studied in the literature. Existing energy distribution of AFL and its variants have two limitations:

- ***Vulnerability Region Unawareness***. They focus on increasing coverage, but lack guidance to direct the fuzzer to strengthen fuzzing code regions that are more likely to be vulnerable.
- ***Mutation Granularity Unawareness***. They randomly select mutators and deterministically specify the number to mutator, therefore lack insights regarding which granularity of mutators are more helpful at that particular stage.

We improve the above two limitations in a principled way. We direct fuzzer to strengthen fuzzing regions that are more likely for a vulnerability to reside based on static semantic metrics of the target program. More specifically, four kinds of promising vulnerable regions (i.e., sensitive, complex, deep and rare-to-reach regions) are given more resources during fuzzing. Furthermore, granularity-aware scheduling for different mutation operators is proposed. The ratio of mutation operators is increased gradually if they have better ability to trigger new paths. All improvements are integrated and implemented into a new open source fuzzing tool named TAFL. Large-scale experimental evaluations are performed showing the effectiveness of each improvement and performance of integration. Furthermore, the proposed tool has helped us find 12 new bugs and identify three new CVEs.

In summary, the contributions of our work are as follows:

- **Improvements.** We improve the AFL's energy distribution by making it vulnerability region aware and mutation granularity aware.
- **Tool.** We implement and integrate our improvements into afl-2.52b and develop a new open source fuzzing tool named TAFL, which is accessible from https://github.com/stuartly/RegionFuzz.
- **Vulnerabilities.** We perform a large-scale evaluation to show TAFL's effectiveness and efficiency. Our approach has helped us to find 12 unknown bugs and identify three new CVEs. (CVE-2018-1000654, CVE-2018-1000667, and CVE-2018-1000886).

II. Approach

We improve two limitations of AFL's fuzzing energy distribution and make it vulnerability region aware and mutation granularity aware.

A. Becoming Vulnerability Region Aware

The core idea of improving AFL's awareness of vulnerable regions is to extract semantic metrics of the target program, and instrument these qualitative metrics' weight into the target program. Then, we assign energy based on the run-time reward of the executed path during fuzzing. In particular, seeds which win more reward will be assigned with more energy. In this way, we direct fuzzer to strengthen fuzzing code regions that are more likely for a vulnerability to reside.

Let weight of a basic block BB for a specific metric be $weight_M(BB)$. Then, reward of a path that executed with input t could be denoted as:

$$Reward(t) = \frac{\sum_{i=0}^{n} weight_M(BB_i)}{n}, BB_i \in path(t) \quad (1)$$

2574-1934/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-Companion.2019.00109

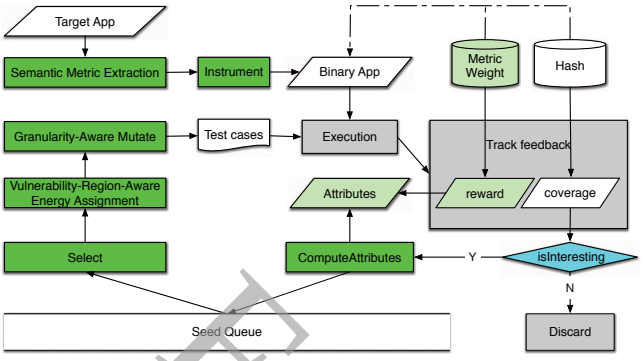

Fig. 1: TAFL Workflow

After an execution is finished, the maximum, minimum reward and average reward are updated. Furthermore, $Factor$ used for energy assignment is computed based on a seed's reward and average reward value.

$$Factor(t) = \frac{Reward(t)}{AvgReward} \tag{2}$$

Let $P_{afl}(t)$ be the energy assigned by AFL for input t, and energy $P(t)$ for test case t assigned after improvement could be donated as following:

$$P(t) = P_{afl}(t) * 2^{10*Factor(t)} \tag{3}$$

Furthermore, four kinds of semantic metrics are designed and used to drive fuzzing based on institutions that the sensitive, complex, deep and rarely reachable regions have more chances to be vulnerable.

B. Becoming Mutation Granularity Aware

We performed an empirical evaluation of AFL's mutators and the results indicate that (1) coarse-grained mutators are better than fine-grained mutators on helping path growth; (2) the effect of combining multiple mutators is better than the effect of using a single kind of mutator.

Granularity-Aware scheduling of mutators is proposed based on the above observations. The proportion of mutation operators, which has better ability to trigger new paths (e.g., extra mutators) will be increased gradually. And the number of mutation operators will be increased over time.

III. Implementation and Evaluation

We incorporate our improvements into afl-2.52b and develop a new fuzzer named TAFL. The workflow of TAFL is demonstrated in Fig. 1. We perform large scale evaluation to show its effectiveness and efficiency on well-known benchmarks (e.g. LAVA-M) and some real-world open source projects.

A. Benefits of Improved Vulnerability Region Awareness

Four kinds of region guided fuzzing (i.e., sensitive, complex, deep and rare-to-reach regions) are evaluated and compared with original AFL. In order to reduce the randoms of fuzzing, we run each project ten times, each time lasts for 24 hours, and get the average value. The Results show the effectiveness of improved vulnerability region awareness. All the four regions guided fuzzers perform better than AFL, especially in terms of the first crash (e.g. max improvement is 41.67% and 21.14% on average) and total crashes (e.g. max promotion is 24.15%, 20.05% on average).

B. Benefits of Improved Mutation Granularity Awareness

We compare AFL, AFLFast, and their extensions with our mutator scheduling improvement on selected benchmarks. The results show that scheduling of mutators is helpful to improve code coverage in most cases. For AFL and AFLast, the average promotion percentages are 7.03% and 4.82% respectively. In a specific case such as libxml2-2.9.2, the promotion could be as high as 15.8% and 14.7%.

C. Evaluation of Integrated Performance

We evaluated the performance of TAFL compared with AFL [5], AFLFast [2] and FairFuzz [6] in overall. Our results show that TAFL performs better than existing AFL based greybox fuzzers in all the tested measurement metrics. Specifically, TAFL obtains 23.40% promotion on triggering the first crash, 27.35% on total crashes, and 18.57% on total paths than the original AFL.

IV. Conclusion

We improve two limitations of existing AFL's fuzzing energy distribution to make it vulnerability region aware and mutation granularity aware. We direct fuzzer to strengthen fuzzing regions that are more likely for a vulnerability to reside. Granularity-Aware scheduling for mutators is proposed. All improvements are integrated into a new open source fuzzing tool named TAFL. The large-scale experiment showed the effectiveness of each improvement and the performance of integration. Furthermore, TAFL has helped us find 12 new bugs and identify three new CVEs.

V. Acknowledgement

The paper was supported by the Nanjing University Innovation and Creative Program for PhD candidate (No.2016014).

References

[1] M. Böhme, V.-T. Pham, M.-D. Nguyen, and A. Roychoudhury, "Directed greybox fuzzing," in *CCS*. ACM, 2017, pp. 2329–2344.
[2] M. Böhme, V.-T. Pham, and A. Roychoudhury, "Coverage-based greybox fuzzing as markov chain," *TSE*, 2017.
[3] P. Chen, H. Chen, Y. Zhang, J. Dai, X. Zhang, S. Huang, Z. Yang, M. Yang, H. Chen, W. Han *et al.*, "Angora: efficient fuzzing by principled search," in *S&P*, vol. 14. Springer-Verlag New York, 2013, pp. 117–149.
[4] S. Gan, C. Zhang, X. Qin, X. Tu, K. Li, Z. Pei, and Z. Chen, "Collafl: Path sensitive fuzzing," in *S&P*. IEEE, 2018, pp. 679–696.
[5] http://lcamtuf.coredump.cx/afl/, "Afl."
[6] C. Lemieux and K. Sen, "Fairfuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage," in *ASE*. ACM, 2018, pp. 475–485.
[7] S. Schumilo, C. Aschermann, R. Gawlik, S. Schinzel, and T. Holz, "kafl: Hardware-assisted feedback fuzzing for os kernels," in *Usenix Security*, 2017.
[8] J. Wang, B. Chen, L. Wei, and Y. Liu, "Skyfire: Data-driven seed generation for fuzzing," in *S&P*. IEEE, 2017, pp. 579–594.