Security Guarantees for Automated Software Testing

Danushka Liyanage
Monash University
Australia

ABSTRACT

Before making important decisions about an ongoing fuzzing campaign, we believe an engineer may want to know: (i) the achieved level of confidence about the program's correctness (***residual risk***), (ii) the expected *increase* in confidence about the program's correctness if we invest more time for the current campaign (***cost-benefit trade-off***), and (iii) the total number of bugs that the fuzzer can find in the limit (***effectiveness***). The ability to accurately estimate the above quantities through observed data of the fuzzing campaign allows engineers to make required decisions with quantifiable accuracy. Currently, there are popular data-driven approaches to provide such quantitative guidance on decision making for *white-* and *blackbox* fuzzing campaigns. However, none of these prevailing techniques can guarantee unbiased estimation of residual risk, cost-benefit trade-off, or effectiveness for *greybox fuzzing* – the most popular automated software vulnerability discovery technique to date. Greybox fuzzers introduce an *adaptive bias* to existing estimators that needs to be corrected during the quantitative analysis to make accurate decisions about the campaign.

In this thesis, our primary objective is to develop a rich statistical framework that supports quantitative decision-making for greybox fuzzing campaigns. We leverage this framework to introduce appropriate bias correction strategies to existing estimators and propose novel estimators that account for adaptive bias in greybox fuzzing.

CCS CONCEPTS

• **Security and privacy** → **Software and application security**; • **Software and its engineering** → Software testing and debugging.

KEYWORDS

software testing, fuzzing, estimation, probability, statistics

ACM Reference Format:
Danushka Liyanage. 2021. Security Guarantees for Automated Software Testing. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3468264.3473097

1 INTRODUCTION

In contrast to software verification, testing makes no guarantee that the program under test (PUT) is error-free for all possible inputs [11]. Then, what makes testing widely adopted in practice over verification? Each new test in a symbolic execution-based *whitebox fuzzer* (an automated software verifier) exercises a different path of the program. That is what makes verification *highly effective* compared to testing techniques like *blackbox fuzzing* that do not involve any program analysis [8]. However, in practice, blackbox fuzzers generate significantly more inputs within a specific time period to achieve a certain path coverage (*higher efficiency*) while the most effective techniques fail to deliver the same coverage within the same time budget [7, 17]. In addition to higher efficiency, factors such as less human dependency also contribute to the increasing utility of software testing for assuring program correctness.

Fuzzing is undoubtedly the most popular automated software testing technique to date. Depending on the degree of program analysis involved, we can distinguish three main fuzzing approaches. *Blackbox fuzzers* repeatedly execute target programs through a fully-randomized series of inputs without leveraging *any* program feedback [6, 13]. Having full access to the source code of the program under test, *whitebox fuzzers* perform program analysis to systematically increase code coverage [17, 20]. The third type *"greybox fuzzing"*, can be positioned somewhere in the middle of these two ends based on how it leverages runtime information. Unlike whitebox fuzzers, it does not perform rigorous program analysis. Yet, greybox fuzzers use program feedback (such as coverage information) to guide the subsequent fuzzing executions. [6, 22]

Greybox fuzzing has gained enormous attention as it outperforms all the existing software vulnerability discovery techniques in terms of efficiency. Major players including Google [19], Microsoft [2], Amazon [1] adopt greybox fuzzers to improve the security of their software products while significantly contributing to continuous improvement of the technique. Many researchers and industrial entities conduct their work towards making greybox fuzzers more powerful in terms of efficiently discovering as many potential software bugs as possible.

A fundamental challenge. Software testing can only be used to show the presence of bugs in a program, but never show their absence [10]. No matter how long the fuzzer has run, there is always some non-zero probability that an undiscovered vulnerability exists [14]. Then, how to make sure that the PUT is sufficiently tested before winding up the fuzzing campaign? In order to accurately decide when to terminate a testing process, we require solid techniques to extrapolate from already observed program behaviors. This is a fundamental challenge in software testing. While there is enormous attention of the research community towards improving the efficiency of fuzzing, we realize the essential requirement of developing specific techniques to overcome the above challenge. Understanding its value, researchers thoroughly demanded rich statistical tools for estimating and predicting the testing progress to deliver program correctness guarantees [15]. There have been

ESEC/FSE '21, August 23–28, 2021, Athens, Greece
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8562-6/21/08…$15.00
https://doi.org/10.1145/3468264.3473097

discussions further on incorporating estimation functions of cost/-effectiveness to address "how" and "how much" of testing should perform under given resources [3].

Currently, there are approaches to adequately perform quantitative reasoning for whitebox [12] and blackbox [4] fuzzing campaigns. Even though the black- and greybox fuzzing are algorithmically quite similar except for the use of program feedback, the stated techniques for blackbox fuzzing produce biased quantification for greybox fuzzing. Therefore, practitioners are unable to make accurate decisions about greybox campaigns with quantifiable accuracy using existing techniques available for other fuzzing varients.

Indicators of fuzzing progress. We draw our attention to three quantities that we think would be essentially useful to make decisions about any ongoing greybox fuzzing campaign. Assume that we run an error-less greybox fuzzing campaign $\mathcal{F}$ over some time t to discover bugs in program $\mathcal{P}$. Suppose the campaign $\mathcal{F}$ has generated n test inputs during this time t. We define,

(1) ***Residual risk*** - Given $\mathcal{F}$ has generated n inputs, what is the probability that the next input (i.e. $n + 1^{\text{th}}$ input) discovers a new bug in $\mathcal{P}$?
(2) ***Cost-benefit trade-off*** - What is the expected reduction in residual risk if more time is invested in generating another m inputs?
(3) ***Effectiveness*** - Asymptotically, how many bugs can the fuzzer expose about program $\mathcal{P}$?

Through this research, we expect to develop a rich statistical framework including techniques to estimate these three quantities that account for adaptive bias in greybox fuzzing campaigns.

2 PROBLEM STATEMENT

Estimation for greybox fuzzing is difficult due to *adaptive bias*. Unlike in blackbox fuzzing, the probability to generate a test input that discovers a particular bug ***increases*** throughout a greybox campaign because new inputs are added to the seed corpus. This violates an important assumption during the estimation which is detrimental to estimator performance. The objective of this thesis is to evaluate and account for adaptive bias during the estimation.

2.1 Adaptive Bias in Greybox Fuzzing

Any black- or greybox fuzzing campaign starts with an initial *seed corpus* which contains one or more valid inputs for the PUT $\mathcal{P}$. In contrast to blackbox fuzzing, the initial seed corpus of a greybox fuzzing campaign is subject to change as it discovers new program behaviors of $\mathcal{P}$. Figure 1 shows the difference between black- and greybox fuzzing at a high level. Unlike blackbox fuzzers, greybox fuzzers add generated test inputs that discover previously unknown program behaviors to the seed corpus. The growing seed corpus leads the greybox fuzzer to discover unseen program behaviors much quicker than a blackbox fuzzer.

STADS framework. Considering software testing as discovery of species–the STADS [4] statistical framework estimates and extrapolates from already discovered program behaviors for fuzzing campaigns. STADS borrows a range of statistical methodologies

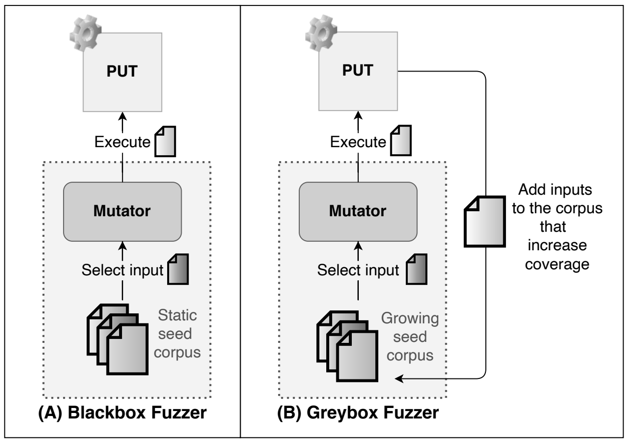

Figure 1: The difference between (A) black- and (B) greybox fuzzing, which causes the adaptive bias.

from over 30 years of research in bio-statistics to tackle the fundamental challenge of developing a "statistically well-grounded extrapolation from program behaviors observed during testing" [4]. The STADS framework models non-deterministic fuzzing as a process of sampling unknown program behaviors based on a smaller number of realistic assumptions about the fuzzing campaign. The framework also includes several bio-statistical estimators to successfully guide quantitative decision making for fuzzing.

STADS mainly assumes that the test inputs are independently sampled from an *invariant* species distribution throughout the fuzzing campaign [9]. Blackbox fuzzing campaigns sufficiently satisfy this basic assumption as each input is generated by applying a finite set of mutation operators to the same set of seeds in the corpus. However, this basic assumption does not hold for greybox fuzzers that leverage program feedback during the campaign. The distribution of which the inputs are sampled does not remain constant for the entire fuzzing campaign. Each time a new seed is added to the corpus, the greybox fuzzer generates inputs from a different distribution than before. Consequently, the probability to discover a specific program vulnerability *increases* over time; hence introduces a significant *"adaptive bias"* to statistical estimators that are presented in STADS framework.

2.2 Significance of Estimating Greybox Fuzzing Progress

We believe that an engineer can have a much clearer picture of the current state of the greybox campaign by knowing accurate estimates for the above three quantities described in the introduction. Respectively, residual risk, cost-benefit trade-off, and effectiveness provide a full view about the current state of the fuzzing campaign based on (1) the progress we have achieved towards discovering all the bugs in the program, (2) the number of additional bugs we can discover by allocating extra time budget, and (3) the total number of bugs that can be discovered in the limit.

When performing any software testing process, we aim to achieve a higher program correctness level with the optimal use of available resources such as time and processing power. In other words, an engineer needs to terminate the testing campaign after the required assurance level is achieved for $\mathcal{P}$. It allows utilizing scares computing power and time for other testing endeavors while being confident about security/functionality of $\mathcal{P}$ in a production environment. Without accurately knowing the above quantities, it might be difficult for an engineer to quantitatively determine at any point of an ongoing greybox campaign, whether the anticipated correctness level is achieved or not. For instance, one cannot confirm whether the risk of observing an unknown vulnerability is below the maximum allowable threshold without an accurate estimate of *residual risk*.

2.3 Thesis Objectives

In this thesis, we intend to propose a rich statistical framework that **introduces estimators of *residual risk, cost-benefit trade-off*, and *effectiveness* that account for adaptive bias in greybox fuzzing**. Leveraging this new framework, engineers and security researchers should be able to make accurate decisions about ongoing greybox fuzzing campaigns while achieving quantifiable assurances about the program's correctness.

2.4 Research Questions

In order to achieve our objectives, we expect to answer the following research questions in our thesis.

- **RQ1.** How do the existing estimators of *residual risk*, *cost-benefit trade-off*, and *effectiveness* perform in the presence of adaptive bias in greybox fuzzing?
- **RQ2.** How can we systematically adjust existing estimators[1] for blackbox fuzzing to account for adaptive bias?
- **RQ3.** How can we fundamentally adjust the statistical or probabilistic foundation to account for adaptive bias during the estimation of residual risk, cost-benefit trade-off, and effectiveness?
- **RQ4.** To improve the efficiency of greybox fuzzing, how can we use the elements of the proposed statistical framework? (Optional)

3 METHODOLOGY

3.1 Quantifying Adaptive Bias of Existing Estimators

As the first step, we expect to quantify the adaptive bias of existing estimators of fuzzer performance through actual fuzzing experiments and simulation studies. Given that we have determined methods to compute the actual value of the respective quantity (i.e. ground truth) during a fuzzing campaign, the respective estimate can simply be compared to measure the adaptive bias. Computed adaptive bias measures for each existing estimator through designed experimental and simulation studies can be analyzed to answer RQ1.

Simulation studies. It is feasible to design simulation frameworks to mimic the basic functionality of black- and greybox fuzzing to explore the inherent behaviors related to adaptive bias. Such simulation frameworks allow us to easily quantify the adaptive bias of the existing estimators of residual risk, cost-benefit trade-off, and effectiveness. Some results we obtained through our current simulation studies are explained in Section 4.1.

Empirical studies. We can design a range of experiments using popular real-world greybox fuzzers such as AFL [23] and LibFuzzer [18] to investigate estimator performance. Most of these fuzzers are open-source and therefore, making required adjustments to calculate ground truth and extract the required data for obtaining estimates are quite possible. For instance, we recently incorporated a ground truth calculation technique for *discovery probability*[2] in LibFuzzer to quantify the adaptive bias of estimators such as *Good-Turing* and *Laplace*. It is also important to carefully choose a range of *test subjects* for experimentation after attentively considering the aspects such as *relevance* and *scale/size*.

The quantified adaptive bias for various estimators can be used to build strong intuitions for introducing bias-corrected versions of these estimators in our thesis.

3.2 Introduce Estimators that Account for Adaptive Bias

We expect to leverage following techniques as we answer RQ2 and RQ3 in our problem statement. The resulting statistical framework will contain both bias-corrected versions of existing estimators and novel estimators that account for adaptive bias in greybox fuzzing.

Statistical approach. We discussed that the cause for adaptive bias in greybox fuzzing is the use of program feedback. Through the outcomes of the above investigations (i.e. answering RQ1), we need to carefully grasp observable factors that vary with adaptive bias (in terms of both magnitude and sign). These factors are the *"sufficient statistics"*[3] in the probabilistic model we develop for greybox fuzzing. Then, our intention should be towards determining the respective functions of these statistics that systematically correct for adaptive bias in existing estimators. There are statistical bias correction techniques in other disciplines like ecology that significantly improve estimation accuracy for specific adaptive sampling techniques [21]. Using the same set of intuitions and probabilistic model above, we seek opportunities to introduce novel estimators of greybox fuzzer performance that account for adaptive bias.

Engineering approach. The explored dynamics in Section 3.1 can also be utilized to engineer existing estimators so that those estimators devote in making decisions fairly accurately for greybox fuzzing. These approaches are entirely not statistically justifiable, but are based on strong intuitions: hence, pragmatically extremely useful. Refer to Section 4.2 for two classes of estimators we engineered for discovery probability.

Evaluation of estimators. We plan to use two basic measures, namely *bias* and *variance* to evaluate the performance and applicability of the proposed estimators. Bias is simply the difference between an estimate and its true value. For any estimation endeavor including fuzzing, we always prefer estimators with *smaller* bias. The variance is simply an indication of how estimates are dispersed around their mean. A good estimator carries a *smaller* variance.

[1] When mentioning "estimators", it refers to all the estimators of residual risk, cost-benefit trade-off, and effectiveness.

[2] Given the fuzzer has generated n inputs, the probability that the next input discovers a new program behavior is defined as the *discovery probability*

[3] In statistics, a *sufficient statistic* carries the same amount of information as sample data about the population parameter of interest.

4 CURRENT STATE OF WORK

Our progress to date includes assessing the behavior of the existing estimators of fuzzer performance (i.e. RQ1) and adjusting existing estimators (i.e. RQ2) of residual risk so that each of those accounts for adaptive bias in greybox fuzzing. Based on the fundamentals of the STADS framework, we performed the first probabilistic analysis for adaptive bias.

4.1 Probabilistic Analysis

Our proposed probabilistic analysis contains several explanations about residual risk and adaptive bias in greybox fuzzing. Under realistic assumptions, we showed that the adaptive bias reduces over time as many fuzzing inputs are generated. When proving this result, we used KL-divergence [16], an information-theoretic concept which measures the distance between two discrete distributions.

We define the *residual risk* of a fuzzing campaign of length n as the probability to discover a vulnerability from the next generated (i.e. $n + 1$-th) test input. In our analysis, we propose to estimate *discovery probability* as the *upper bound* for residual risk for greybox fuzzing campaigns. For a campaign of the same length n, the discovery probability ($\Delta(n)$) is the probability to discover a new species from $n + 1$-th input.

Simulation studies. We designed several simulation studies imitating black- and greybox fuzzing to support our analysis outcomes and to explain the obtained empirical results. Figure 2 shows one of our simulation results that the probability to discover a certain bug *increases* as the greybox campaign progresses. Our simulations and results are made available at https://www.kaggle.com/adaptivebias/simulation.

We further extend our simulations to investigate the fuzzing dynamics related to species accumulation (which contributes to RQ1) and the asymptotic number of species in the proposed probabilistic framework for greybox fuzzing.

4.2 Experimental Results

Behavior of existing estimators. To make empirical investigations about residual risk, we designed several fuzzing experiments using the state-of-the-art greybox fuzzer, **LibFuzzer** [18]. It was evident that the proposed residual risk estimators in the STADS framework are substantially underestimating the true residual risk of greybox fuzzers. Leveraging such a classical residual risk estimator is undesirable as it falsely provides higher confidence about program correctness. Instead, a security researcher who uses a *positively-biased* residual risk estimator for his/her decision making is unlikely to terminate the fuzzer before reaching the desired correctness guarantee about the program.

Extrapolating discovery probability. Additionally, we found an interesting power-law relationship between the discovery probability and the generated number of inputs in log-log scale. This *scale invariance* property enables us to extrapolate the discovery probability for multiple orders of magnitude of test inputs with data for a much shorter campaign length (i.e. with a few thousands of test inputs). We fitted simple linear models for $\log(\Delta(n))$ against $\log(n)$, using just 10^4 out of 10^9 test inputs for each campaign. These linear regression models happen to explain more than $R^2 = 97\%$ variance of actual $\log(\Delta(n))$ for most of the subjects.

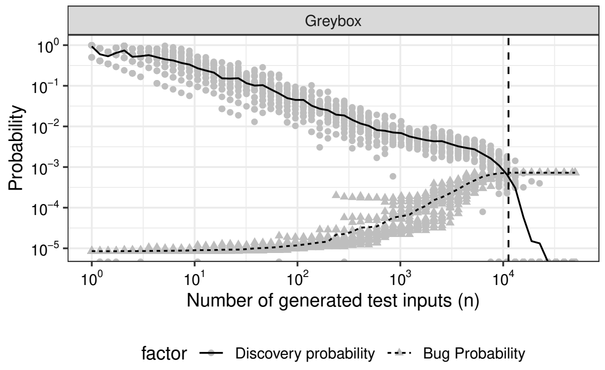

Figure 2: In greybox fuzzing, the probability to discover a bug *increases* as new seeds are added to the corpus . The vertical dashed line shows the average time-to-error.

Proposed estimators of discovery probability. We proposed two new classes of estimators (as part of answering RQ2) for residual risk in greybox campaigns namely *reset estimators* $\hat{\Delta}_R(n, a)$ and *mean local estimators* $\hat{\Delta}_{ml}(n)$. Based on the presented evaluation matrices in Section 3.2; mean local estimators showed the best performance. In fact, the *mean local Good-Turing* $\hat{\Delta}_{ml.GT}(n)$ estimator almost never under-estimated the true discovery probability and had lowest variance.

4.3 Ongoing and Planned Work

All the estimators and approaches presented above are derived through various intuitions of how adaptive bias emerges in greybox fuzzing campaigns. Our completed work [5] is accepted to be published at the ESEC/FSE'21 conference. Going beyond these "engineering approaches", we intend to enrich our probabilistic framework with a suite of statistical estimators that account for adaptive bias in greybox fuzzing. Successful completion of the above-mentioned suite of estimators contributes to RQ3. Later, we plan to introduce more estimators of cost-benefit trade-off and effectiveness addressing RQ2 and RQ3 to systematically support accurate decision making for practical greybox fuzzing processes.

5 CONTRIBUTIONS AND CONCLUSION

In this thesis, we draw our primary attention towards *correcting the problem of adaptive bias* in search-based software testing techniques such as greybox fuzzing. One of the primary objectives of the research is to introduce *dynamic bias correction* strategies to existing estimators for residual risk, cost-benefit trade-off, and effectiveness. We expect to develop a solid statistical/probabilistic framework by incorporating a range of estimation techniques to answer primary research questions in this thesis.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisers Marcel Böhme, Chakkrit Tantithamthavorn, and Zahra Mirzamomen for their invaluable guidance and continuous support throughout this journey. This research is co-funded by Australian Research Council (ARC) (DE190100046) and Monash Univeristy.

REFERENCES

[1] 2017. How we found a tcpdump vulnerability using cloud fuzzing. https://www.softscheck.com/en/identifying-security-vulnerabilities-with-cloud-fuzzing
[2] 2020. Microsoft announces new Project OneFuzz framework, an open source developer tool to find and fix bugs at scale. https://www.microsoft.com/security/blog/2020/09/15/microsoft-onefuzz-framework-open-source-developer-tool-fix-bugs/
[3] Antonia Bertolino. 2007. Software Testing Research: Achievements, Challenges, Dreams. In *Future of Software Engineering (FOSE '07)*. 85–103. https://doi.org/10.1109/FOSE.2007.25
[4] Marcel Böhme. 2018. STADS: Software Testing as Species Discovery. *ACM Transactions on Software Engineering and Methodology* 27, 2, Article 7 (June 2018), 52 pages. https://doi.org/10.1145/3210309
[5] Marcel Böhme, Danushka Liyanage, and Valentin Wüstholz. 2021. Estimating Residual Risk in Greybox Fuzzing. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), Aug 23–27, 2021, Athens, Greece (ESEC/FSE)*. 11 pages.
[6] Marcel Böhme, Valentin Manès, and Sang Kil Cha. 2020. Boosting Fuzzer Efficiency: An Information Theoretic Perspective. In *Proceedings of the 14th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 1–11.
[7] Marcel Böhme and Soumya Paul. 2014. On the efficiency of automated testing. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 632–642.
[8] Marcel Böhme, Van-Thuan Pham, and Abhik Roychoudhury. 2017. Coverage-based greybox fuzzing as markov chain. *IEEE Transactions on Software Engineering* 45, 5 (2017), 489–506.
[9] Marcel Böhme. 2019. Assurances in software testing: A roadmap. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 5–8.
[10] Edsger Wybe Dijkstra. 1970. Notes on Structured Programming. (April 1970). EWD249.
[11] Vijay D'silva, Daniel Kroening, and Georg Weissenbacher. 2008. A survey of automated techniques for formal software verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 7 (2008), 1165–1178.
[12] Jaco Geldenhuys, Matthew B. Dwyer, and Willem Visser. 2012. Probabilistic Symbolic Execution. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis (ISSTA 2012)*. 166–176.
[13] Patrice Godefroid. 2007. Random Testing for Security: Blackbox vs. Whitebox Fuzzing. In *Proceedings of the 2nd International Workshop on Random Testing: Co-Located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)* (Atlanta, Georgia) *(RT '07)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/1292414.1292416
[14] D. Hamlet and R. Taylor. 1990. Partition testing does not inspire confidence (program testing). *IEEE Transactions on Software Engineering* 16, 12 (1990), 1402–1411.
[15] Mary Jean Harrold. 2000. Testing: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (Limerick, Ireland) *(ICSE '00)*. Association for Computing Machinery, New York, NY, USA, 61–72. https://doi.org/10.1145/336512.336532
[16] Solomon Kullback and Richard A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (March 1951), 79–86.
[17] Jun Li, Bodong Zhao, and Chao Zhang. 2018. Fuzzing: a survey. *Cybersecurity* 1, 1 (2018), 6.
[18] LibFuzzer. 2019. LibFuzzer: A library for coverage-guided fuzz testing. http://llvm.org/docs/LibFuzzer.html. Accessed: 2020-08-26.
[19] Kostya Serebryany. 2017. OSS-Fuzz-Google's continuous fuzzing service for open source software. (2017).
[20] Michael Sutton, Adam Greene, and Pedram Amini. 2007. *Fuzzing: brute force vulnerability discovery*. Pearson Education.
[21] Steven K Thompson. 1990. Adaptive cluster sampling. *J. Amer. Statist. Assoc.* 85, 412 (1990), 1050–1059.
[22] Valentin Wüstholz and Maria Christakis. 2018. Learning inputs in greybox fuzzing. *arXiv preprint arXiv:1807.07875* (2018).
[23] Michal Zalewski. 2015. American fuzzy lop.(2015). *URL http://lcamtuf. coredump. cx/afl* (2015).