Risk Estimation in Differential Fuzzing via Extreme Value Theory

Rafael Baez
University of Texas at El Paso
rbaez2@miners.utep.edu

Alejandro Olivas
University of Texas at El Paso
aolivas23@miners.utep.edu

Nathan K Diamond
University of Texas at El Paso
nkdiamond@miners.utep.edu

Marcelo Frias
University of Texas at El Paso
mfrias4@utep.edu

Yannic Noller
Ruhr University Bochum
yannic.noller@acm.org

Saeid Tizpaz-Niari
University of Illinois Chicago
saeid@uic.edu

***Abstract*—Differential testing is a highly effective technique for automatically detecting software bugs and vulnerabilities when the specifications involve an analysis over multiple executions simultaneously. Differential fuzzing, in particular, operates as a guided randomized search, aiming to find (similar) inputs that lead to a maximum difference in software outputs or their behaviors. However, fuzzing, as a dynamic analysis, lacks any guarantees on the absence of bugs: from a differential fuzzing campaign that has observed no bugs (or a minimal difference), what is the risk of observing a bug (or a larger difference) if we run the fuzzer for one or more steps?**

This paper investigates the application of Extreme Value Theory (EVT) to address the risk of missing or underestimating bugs in differential fuzzing. The key observation is that differential fuzzing as a random process resembles the maximum distribution of observed differences. Hence, EVT, a branch of statistics dealing with extreme values, is an ideal framework to analyze the tail of the differential fuzzing campaign to contain the risk. We perform experiments on a set of real-world Java libraries and use differential fuzzing to find information leaks via side channels in these libraries. We first explore the feasibility of EVT for this task and the optimal hyperparameters for EVT distributions. We then compare EVT-based extrapolation against baseline statistical methods like Markov's as well as Chebyshev's inequalities, and the Bayes factor. EVT-based extrapolations outperform the baseline techniques in 14.3% of cases and tie with the baseline in 64.2% of cases. Finally, we evaluate the accuracy and performance gains of EVT-enabled differential fuzzing in real-world Java libraries, where we reported an average saving of tens of millions of bytecode executions by an early stop.

I. Introduction

Modern software systems are notoriously prone to failures, which in 2020 alone caused an estimated $1.56 trillion in economic losses in the U.S. [1]. To mitigate such risks, developers rely on debugging, testing, and verification as critical processes for identifying and preventing bugs. A major research and engineering challenge is to design automated techniques that make these processes scalable and effective. Among existing approaches, fuzzing has emerged as one of the most powerful techniques for automatically detecting bugs and vulnerabilities [2].

A widely used variant of fuzzing is graybox fuzzing [3]. It employs evolutionary algorithms to search the input space, guided by program internals such as whether a test input explores a new path in the control-flow graph. Despite its effectiveness, fuzzing can only demonstrate the absence of bugs for the specific inputs it generates, leaving uncertainty about untested inputs. To address this uncertainty, there have been significant efforts within the software engineering community to provide statistical guarantees on the fuzzing campaign [4], [2], [5], [6], [7], [8]. In particular, they aim to answer questions like: "What is the likelihood of identifying a new vulnerability when running the fuzzing campaign for many more hours?"

Most existing work provides statistical guarantees of fuzzing, focusing on finding single inputs that witness a crash. However, there are significant classes of software requirements that cannot be tested via single input/output generation. Those include software correctness requirements that face the oracle problem [9], [10], [11], [12], [13], [14] (i.e., the ground truth is not available), hyperproperties [15], [16], [17], [18], [19], [20] (i.e., two or more traces together deem correctness or witness a bug), and differential testing [21], [22], [23] (i.e., finding bugs by comparing execution results of similar implementations).

Due to significant applications and challenges in the domain of differential analysis, for example, with differential fuzzing, we study the feasibility of statistical guarantees for this class of software testing methods. Differential testing represents a random process that draws random variables, each taking a value of outputs or behavior differences between two or more similar inputs when executed on the target software. Similar to the statistical guarantees in conventional fuzzing, we should be able to derive statistical claims about the maximum of such random variables in a given period of time even if the underlying distribution is unknown. Therefore, we pose a central question at the heart of differential fuzzing:

> From a differential fuzzing campaign that has witnessed a difference $\delta \geq 0$ after t iterations, what is the risk of observing larger differences if we run the fuzzer for one or more steps longer?

We posit that the worst-case divergence of differential fuzzing represents the maximum of random variables in the fuzzing campaign. Therefore, the statistics of extremes [24]

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00036

that model the tail distribution of random variables is a natural choice. Crucially, extreme value theory (EVT) [25] can reason about the likelihood as well as the amounts of extreme values (return levels) in a given period of time (return period). Hence, we pose the following research question:

> Given an observed (max) cost difference $\delta \geq 0$ between two (similar) inputs at iteration m of differential fuzzing, can EVT estimate the maximum differences in the next n iterations, up to an upper bound $n \leq N$?

If so, we could minimize the runtime of differential fuzzing campaigns by an early termination while still maintaining a high-quality in finding significant differences. In this paper, we focus on the class of differential fuzzing for side-channel analysis. Specifically, we consider DIFFUZZ [15], a state-of-the-art technique that searches for two secret inputs under the same public input, leading to a maximum cost difference in their executions on the target software. Identifying such a high-cost difference between secrets indicates a vulnerability that can allow attackers to exploit timing side channels to infer information about the secrets. Note that small cost differences may not be manifested in the execution times; hence we require to find strong differences to deem a vulnerability. Therefore, DIFFUZZ is an ideal case study for our framework.

However, there are multiple challenges in adapting EVT to provide statistical guarantees in differential fuzzing. One challenge is the notion of statistical testing on the tail samples during the fuzzing to stop differential fuzzing. While such notions are well studied for the statistics of regular distributions (e.g., normal distributions), an appropriate notion of statistical testing of tail distributions is challenging due to the scarcity. While the generalized extreme value (GEV) [26] distributions (as a result of EVT theory) can reason about the tail, they require setting different parameters that can significantly influence the outcome. One such parameter is a threshold parameter that deems any sample that exceeds this threshold a tail sample. The proper choice of threshold is critical. An underestimation of the (ground truth) threshold leads to mixture distributions that violate the asymptotic basis of GEV distributions. Similarly, an overestimation can only include fewer tail samples, which can lead to low confidence in the model due to high variance. There are also different types of GEV distributions, such as Exponential, Poisson Process, and Pareto Distributions, that have different statistical properties. So, the first question is:

RQ1. *How to infer an ideal statistical testing of tail distribution and hyperparameters of GEV distributions as well as its type to accurately estimate the worst-case differential fuzzing?*

Once we infer suitable configurations of tail extrapolations, the next challenge is to establish their efficacy when compared to the baseline statistical techniques. One class of such techniques is based on concentration inequalities, such as Markov's and Chebyshev's inequalities. However, these techniques focus on the tails of regular distributions, rather than modeling the tail distribution directly. Another class of algorithms is various derivations of the Bayes factor that rely on statistical hypothesis testing based on the sequence of extreme events.

RQ2. *How does the extrapolation based on EVT compare to the baseline statistical methods?*

Finally, to establish the usefulness of extreme value theory, it is crucial to study the accuracy of extreme value theory on a set of larger real-world Java web applications. We leverage a set of known side-channel vulnerabilities in critical libraries such as Jetty, Spring Security, and Apache ftp server. In addition, we study how much the prediction of EVT reduces the performance overhead of longer fuzzing campaigns in terms of the number of bytecode executed.

RQ3. *What are the accuracy and performance gains of EVT for extrapolating the worst-case differential fuzzing on the set of larger Java libraries?*

We find that a combination of the Poisson Process distribution with bootstrapping leads to the best configurations for EVT-based extrapolations. Our experiments show that our method outperforms the baseline in 14.3% cases and achieves competitive results in the remaining 64.2%. In addition, our approach provides tight upper bounds in 57.1% of cases, while the competitive baselines are prone to false negatives (underestimation). We also find that EVT-based extrapolations do not underestimate the worst-case differences in the fuzzing. The overestimation ranges from 32.8% to 206.8%. We show that an EVT-enabled DIFFUZZ saved the execution of at least a hundred thousand and up to 1 billion bytecode instructions.

Contributions. The key contributions of this paper are:

- We formalize the connection between differential fuzzing and extreme value theory to extrapolate the maximum differences of an unseen fuzzing campaign;
- We infer the parameters and hyperparameters of extreme value theory for Java programs, such as the early stopping criterion for differential fuzzing;
- We compare the EVT-based extrapolation to three baseline statistical methods; and
- We show the usefulness and performance advantages of EVT-enabled differential fuzzing over larger Java libraries like Spring Security and Apache ftpserver.

II. BACKGROUND AND RELATED WORK

We provide a concise background on the extreme value theory, differential fuzzing, and fuzzing guarantees.

Extreme value theory. EVT [25] is a branch of statistics that deals with the analysis of extreme events in a random process. Given a set of independent and identically distributed random variables $\{a_1, \ldots, a_n\}$, the extreme value theory is concerned with the min/max statistics of a random process as for instance $M_n = \max(\{a_1, \ldots, a_n\})$ as $n \to \infty$.

Under some mild assumptions about the smoothness of the distribution via normalizing constants a_n and b_n, it has been proved (e.g., see Leadbetter et al. [26]) that $Pr[(M_n - b_n)/a_n < a] \to G(a)$ as $n \to \infty$ and G belongs to a family of distributions called the *generalized extreme value (GEV)*

family. Each such distribution has the cumulative distribution function (CDF) of the form

$$G(a) = \exp\left\{-\left[1+\xi\left(\frac{a-\mu}{\sigma}\right)\right]^{-1/\xi}\right\},$$

defined over $\{a : 1+\xi(a-\mu)/\sigma > 0\}$. The model has three parameters: a location parameter $-\infty < \mu < +\infty$, a scale parameter $\sigma > 0$, and a shape parameter $-\infty < \xi < +\infty$. Depending on the shape parameter ξ, the GEV distribution can be classified into three types. Type I, known as the Gumbel family, defines a subset of GEV distribution when $\xi \to 0$. The tail behavior of type I, a_+, has infinite support, but the density of GEV decays exponentially (extrapolations are feasible up to a bounded time horizon). For type II, $\xi > 0$ and a_+ have infinite support (heavy tail), decaying polynomially, and hence no guarantee may be feasible. Finally, for type III, $\xi < 0$ and a_+; it has bounded set of values (light tail). In this case, the statistical guarantees for the worst-case outcomes are feasible.

Generalized Pareto Distribution. There are two basic approaches to infer the parameters of GEV distributions; block maximum and threshold approach. The block maximum approach divides samples into blocks of the same size and uses the maximum of each block as the extreme value. Since such an approach is more appropriate for seasonal data, we use the threshold approach, where extreme events exceed some high threshold u. In other words, $\{a_i : a_i > u\}$, are extreme values. Labeling these exceedances by $\{d_{(1)}, \ldots, d_{(k)}\}$, we define the threshold excesses by $d_j = a_j - u$ for $1 \leq j \leq k$. It follows that if $Pr[M_n < a] \to G(a)$, then for large enough u, the distribution function is approximately:

$$H(t) = 1 - \left(1 + \frac{\xi t}{\hat{\sigma}}\right)^{-1/\xi}$$

where $t > 0$ and $\hat{\sigma} = \sigma + \xi(u-\mu)$ [25]. This distribution is known as *generalized Pareto* distribution. The implication of the shape parameter ξ is the same as $G(a)$, as a special case of GEV distribution. Our EVT approach follows this distribution.

Threshold Selection. A proper choice of threshold value u is critical to analyze the behavior of extreme value distributions. Low values of the threshold u might include non-tail samples and lead to mixture distributions that violate the asymptotic basis of the model. On the other hand, high values of the threshold u might include only a few tail samples and lead to low confidence in the model. Hence, it is critical to be confident on the threshold to infer tail distributions.

Return Levels. The inverse of the probability density function of GEV at probability p, is the *return level* a_p, associated with the *return period* $1/p$. The level a_p is expected to be exceeded on average once every $1/p$ period of time. A *return level* is represented with (m, a_m), where m is the time period (E.g. the number of iterations in a fuzzing campaign) and the level a_m is the expected extreme value during the m period (e.g., expected worst-case differences in the next m interactions).

Differential Fuzzing. Differential fuzzing for side-channel vulnerabilities with DIFFUZZ [15] deploy the concept of self-composition [27] to identify violations of the non-interference principle. Given a side channel such as execution time, DIFFUZZ attempts to find two secret inputs z_1 and z_2 and one public input x so that the executions of the program under test $\mathcal{P}$ lead to different observations along the side channel c: $c(\mathcal{P}[x, z_1]) \neq c(\mathcal{P}[x, z_2])$. Since the only difference between the two executions is the secret input, an observed difference in the execution behavior along a particular side channel must originate from a secret-dependent path. This does not guarantee that a vulnerability can be exploited, but indicates a vulnerability that should be further investigated. To find inputs that expose this behavior, DIFFUZZ uses a custom graybox fuzzing algorithm that is guided to maximize the cost difference $\delta = |c(\mathcal{P}[x, z_1]) - c(\mathcal{P}[x, z_2])|$ between executions.

Statistical Guarantees for Fuzzing. While there are significant efforts (e.g., ReFuzz [28] to improve the efficacy of graybox fuzzing, we focus on fuzzing techniques with statistical guarantees on stopping criteria. In the context of black-box fuzzing, Woo et al. [29] model the fuzzing process as a weighted coupon collectors problem [30]. A fuzzer discovering a new path or revisiting the same path is akin to collecting seen or unseen coupons. The STADS framework [31] draws similarities between the ecological question of estimating the number of rare species in an area to the number of rare program paths that may contain unseen bugs. Hence, a practitioner can use the STADS to estimate rare paths based on discovered ones during a graybox fuzzing campaign. Residual risk estimation [32] extends the STADS framework to quantify the risk of stopping the fuzzing campaign early and the probability of discovering the next unseen path. Similarly, estimation saturation in fuzzing (Reachable Coverage [33]) provides a statistical approach to the problem of estimating reachable paths as a stopping criterion. Green Fuzzing [34] uses the coverage of potentially vulnerable code fragments rather than the entire code basis as a saturation-based stopping criterion. Rare path guided fuzzing [8], [35] uses a combination of probabilistic analysis and symbolic methods to identify rare paths to guide graybox fuzzing. While these works estimate the maximum trials to cover all reachable paths in the control-flow graphs, our work aims to find the maximum differences between the execution of two similar inputs (i.e., the number of trials to observe the maximum cost differences between two executions of similar inputs. Furthermore, information-flow graphs are critical to our stopping criteria of differential fuzzing, but none of these prior works integrate information flow to estimate when to stop fuzzing. An example is the following program: `void apply_and_wait(bit[32] secret, bit[32] guess): return sleep(secret & guess)` where there is only one control-flow graph path, but there are $65,536$ different waits (costs). The maximum cost difference between two executions is 2^{16}, and it happens when one run has been executed with secret=0...0 and guess=1...1, while another has been executed with secret=1...1 and guess=1...1. Our approach uses EVT to extrapolate the maximum cost differences from observations during differential fuzzing.

stringEquals (WSS4J)

```
boolean stringEquals(String s1, Object s2)
{
  if (s1 == s2) {
     return true;
  }
  if (s2 instanceof String) {
     String s2Str = (String)s2;
     int n = s1.length();
     if (n == s2Str.length()) {
          char v1[] = s1.toCharArray();
          char v2[] = s2Str.toCharArray();
          int i = 0;
          while (n-- != 0) {
               if (v1[i] != v2[i])
                    return false;
               i++;
          }
          return true;
     }
  }
  return false;
}
```

Fig. 1: String equality in Apache WSS4J (`s1` secret, `s2` public). The code snippet is the implementation for the secret comparison with a given public guess.

EVT has been widely used to provide probabilistic guarantees on worst-case execution times in real-time and embedded systems [36], [37], [38], [39]. Prior work has explored EVT's application to timing analysis with random caches, studied the impact of relaxing i.i.d. assumptions, and developed techniques for handling measurement challenges on modern hardware. EVT has been used to detect rare bugs in circuit design[40] and to estimate the worst-case delay of VLSI circuits [41]. While these studies demonstrated EVT's effectiveness for timing analysis in embedded systems, they focused primarily on hardware-level timing behaviors and single worst-case execution, rather than software testing applications.

III. Overview

Let us first consider a code snippet taken from a vulnerable password matching program in Apache WSS4J [42]. Figure 1 shows the code where the secret string s is compared to a given public guess string g that has the same length (32 characters are used for this example). We also abstract the costs with the number of bytecode instructions executed.

In this paper, we leverage DifFuzz for our analysis. We run the fuzzer for 1 hour and collect 23,226 samples. DifFuzz satisfies exponentiality testing at the iteration 3,226. We ran fuzzing further for 20,000 more iterations to collect the ground truth. In other words, we used the first 3,226 samples as training samples, and the rest of fuzzing as testing data samples. The maximum cost observed over the training samples is 69. The mean and standard deviations over the training samples is 2.9 and 5.6, respectively. We use the training samples to extract GEV distributions. In doing so, the first step is to find a suitable threshold value such that any values that exceed this threshold belong to the tail distribution of differential fuzzing. We used a bootstrapping technique (see details in Approach section V) that takes the training samples and infers a suitable threshold. For the example of Apache WSS4J, the threshold sets to 32, meaning that any bytecode differences above 32 are deemed extreme values. Figure 2 (a) shows the cost differences in training set, the threshold for extreme values, and samples that belong to tail distributions (13 samples).

TABLE I: Return Levels of max. costs of graybox fuzzing (WSS4J password matching).

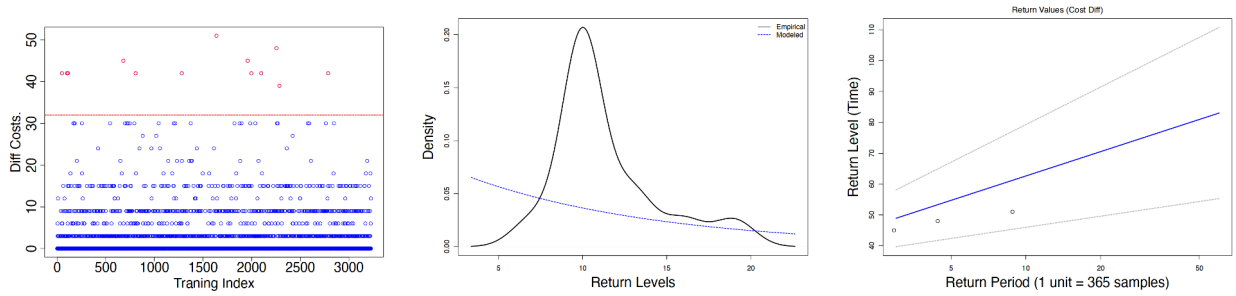
Level	Return Level (**observed**)	Return Level (**Exp.**)	Error of Exp. (%)
1,000	84	43.3 [36.3, 50.3]	-40.1
2,000	84	50.3 [38.9, 61.6]	-26.6
5,000	96	59.4 [42.4, 76.4]	-20.4
10,000	96	66.3 [45.1, 87.6]	-0.09
20,000	96	73.3 [47.7, 98.9]	+0.03

Once the threshold value is inferred, we choose a type of extreme value distribution and infer its parameters. For this benchmark, we use Exponential and Poisson Process (PP) types and infer their parameters via Bayesian optimization. Figure 2 (b) shows the empirical tail distribution with exponential distribution. The location and shape of distributions are 36.4 (+/- 2.4) and 11.4 (+/- 3.2), respectively. One crucial aspect of GEV analysis is the concept of "return-level." It shows the expected maximum value for a given time period in the future. We use the concept of return levels to extrapolate the expected maximum cost differences in the next m-iterations of DifFuzz. Figure 2 (c) shows the return levels where one unit is 365—prevalent EVT models use the number of days in a year—fuzzing iterations. Table I shows the results of GEV-based extrapolations (**Prediction**) at iteration 3,226 of fuzzing, when the exponentiality testing passed. In the next 20,000 fuzzing iterations, we observe the maximum cost difference of 96 bytecodes, whereas the GEV prediction shows a maximum cost difference of 73.3 [47.7, 98.9], which is an overestimation of 0. 03%. Overall, as the number of fuzzing iterations increases, the prediction provides better estimates.

Figure 3 shows the temporal progress of maximum cost differences vs. predictions with EVT with exponential (Figure 3a) and PP (Figure 3b) bases for the first 10,000 fuzzing iterations. Each point in the x-axis shows the number of iterations used to infer EVT distributions, and the corresponding value in the y-axis shows the 95% confidence intervals of the prediction (green series with blue intervals) vs. the ground truth max. differences (red series) up to next 1,000 iterations. The plots show that the PP is more sensitive to changes in the distribution of cost differences and can predict the ground truth more accurately. On the other hand, exponential is more conservative and may provide a sound upper-bound.

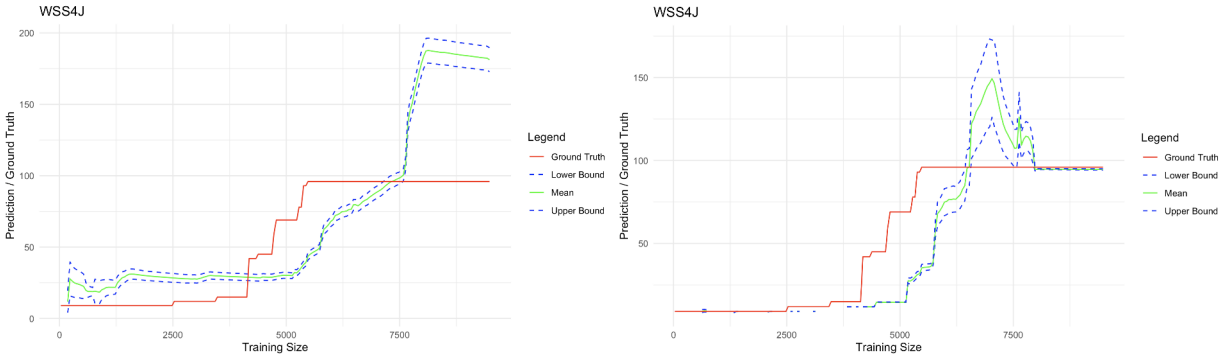
IV. Problem Statement

We consider differential testing methods that take a program $\mathcal{P}$ and their input domain variables $\mathcal{A}$ and search to find two inputs that share a common property ϕ, but their executions on the program $\mathcal{P}$ lead to a maximum difference. For example, a side-channel fuzzer like `DifFuzz` [15] takes a program $\mathcal{P}$ with two set of input variables, a secret set of variables Z (e.g., stored secret password) and a public set of variables X (e.g., a



(a) Training samples and the threshold. (b) Tail Distribution with Exponential Basis. (c) Return Levels of Max. Cost Differences.

Fig. 2: Overview Example. (a) The cost differences over training samples (the first 3,226 samples in DIFFUZZ). (b) The empirical tail distribution of DIFFUZZ. (c) m-return level plot of cost differences with expected values (and their 95% CI).



(a) Predicting via Exponential Basis. (b) Predicting via Poisson Process.

Fig. 3: Temporal Plot of Prediction. We use the size of training (x-axis) to predict the max difference in the next 1,000 fuzzing iterations (green) as compared to the ground truth (red) with Exponential and PP distributions.

guess for a password) and tries to find two inputs with the common public input values which led to a maximum difference in the side-channel observation (e.g., timing) of program $\mathcal{P}$ due to a difference in the secret input, i.e., $\texttt{DifFuzz}(\mathcal{P}, \text{time}, Z, X) ::= \max_{z_1,z_2,x} .|\mathcal{P}_{time}(x, z_1) - \mathcal{P}_{time}(x, z_2)|$, hence it shows the presence of (strong) leaks of secrets via side channels.

One critical limitation of differential testing and fuzzing techniques, due to their dynamic execution nature, is that they are prone to false negatives and cannot provide guarantees on the worst-case divergence. Let $\Delta = \{\delta_1, \ldots, \delta_m\}$ be the set of random variables following an underlying unknown distribution of differentials, where $\delta_i = |\mathcal{P}(a) - \mathcal{P}(a')|$ shows the difference at step i of fuzzing. Practically, we try fuzzing for at most $m << n$ iterations and are interested in estimating $\texttt{max}(\delta_{m+1}, \ldots, \delta_n)$, assuming that our fuzzing process has become stationary at step m of fuzzing. Following the central limit theorem as $n \to \infty$, the sum of random variables follows a Gaussian distribution, i.e., $\mathcal{N}(\mu, \sigma)$ where $\mu = \texttt{average}(\Delta)$ and $\sigma = \texttt{std}(\Delta)$. However, infamous concentration results (e.g., Markov and Chebyshev Inequalities [43]) provide tail guarantees on the *expected* cost differences; while we are interested in the statistical guarantees on the *maximum* differences.

Statistics of Tail Distributions. Rather than modeling the distributions of expectations, we are interested in the tail distribution, i.e., $M_n = \max(\{\delta_1, \ldots, \delta_n\})$. In fact, the testing and fuzzing campaigns are exploring the tail distribution of differentials without any explicit model of such distributions. In the same way that the central limit theorem relates the sampling distribution of expectation to Gaussian distribution; EVT connects the sampling distribution of maximum of random variables to the GEV family of distributions.

Definition IV.1 (Tail Distributions of Differential Fuzzing)**.** *Given a differential fuzzing technique that takes a program and searches its space to find inputs that characterize the maximum differences between similar inputs after m iterations (e.g., after satisfying a statistical testing or after a time-out), our goal is to model the tail distribution of differential fuzzing and extrapolate "what would be the (expected) worst-case difference δ_n if the fuzzer had run for n more iterations?"*

V. APPROACH

Our key approach is to leverage EVT to model the tail of differential fuzzing processes. The key advantage is that EVT

Algorithm 1: EVT-enabled DIFFUZZ algorithm.

Input: Program $\mathcal{P}$, Fuzzing infrastructure FUZZ, Statistical Test of Tail `STT`, Worst-Case Differential Predictions `WCDiff`, Time-out $\mathcal{T}$.

Δ, $Pred$, i, $t \leftarrow \{\}$, -1, 0, `current`(time)
while *current(time)* $\leq t + \mathcal{T}$ **do**
  $z_1, z_2, x \leftarrow$ FUZZ($\mathcal{P}$, $max_{inp}(\Delta)$)
  $cost_1 \leftarrow$ MEASURE($\mathcal{P}$, z_1, x)
  $cost_2 \leftarrow$ MEASURE($\mathcal{P}$, z_2, x)
  $\delta \leftarrow |cost_1 - cost_2|$
  Δ.`add`(δ, $[x, z_1, z_2]$)
  if *STT*(Δ) **then**
    $Pred \leftarrow$ `WCDiff`(Δ)
    return Δ, $Pred$, i
  $i \leftarrow i$ + 1
return Δ, $Pred$, $\mathcal{T}$

Algorithm 2: `Exponentiality` as a type of `STT`

Input: Differentials Δ, Min. Number of Samples k_{min}, Max. Number of Samples k_{max}.

if *size*(Δ) $< k_{max}$ **then**
  return `False`
$res \leftarrow$ `True`
for $k \leftarrow k_{min}$ *to* k_{max} **do**
  $\Theta \leftarrow$ `Select_Top_k`(Δ, k)
  $\overline{\Theta}$, $\sigma(\Theta) \leftarrow$ `average`(Θ), `std`(Θ)
  $CV_k \leftarrow \frac{\sigma(\Theta)}{\overline{\Theta}}$
  if $CV_k \geq 1.0 + (\frac{1}{4*k})$ **then**
    $res \leftarrow$ `False`
    break
return res

directly models the tail distribution, allowing us to reason about the validity and extrapolate the return levels δ_n of extreme values for a time period of n.

Algorithm 1 shows the overall approach that is an extension to differential fuzzing DIFFUZZ [15] with statistical guarantees. It takes a target program $\mathcal{P}$, a basic fuzzing FUZZ, a statistical testing of tail `STT`, a worst-case predictor `WCDiff`, and the maximum allowed time or number of iterations for the analysis $\mathcal{T}$ as inputs. In the first step, the algorithm begins by initializing Δ as an empty set to store differences and $Pred$ to -1 to extrapolate the worst-case differences.

Then, a `while` loop is started to run as long as the current time t does not exceed the time-out limit $\mathcal{T}$. In each iteration, it uses the differential fuzzing tool DIFFUZZ to generate two similar inputs (x, z_1) and (x, z_2) that only differ in some sensitive features by querying the program $\mathcal{P}$. Then, it measures the cost of each execution (I.e., in terms of executed bytecodes), and stores the results in $cost_1$ and $cost_2$. Then it calculates the absolute difference δ between the two measured costs. DIFFUZZ finally adds the computed difference to the set of differences Δ as the feedback to the fuzzing engine besides other metrics such as the input visited a new path in the control flow graph.

Next, our approach performs a statistical testing (`STT`) during the fuzzing, and if the collected samples satisfy the test, we query the prediction models (`WCDiff`) that model the tail distributions of cost differences to extrapolate the maximum differences in a given number of fuzzing trials, and we terminate the fuzzer. To collect the ground truth differences, in practice, we continue fuzzing to record the differentials up to the time-out. Figure 4 visualizes Algorithm 1 to infer the worst-case difference in a differential fuzzing via EVT.

A. Statistical Testing of Tail (`STT`)

There are multiple statistical tests on the tails of random processes to convince a reliable distribution of tails [44]. We consider two tests: 1) Laplace and 2) Exponentiality.

Laplace. The widely recognized Laplace estimator assigns a small probability to unobserved events by treating each one as if it had been observed exactly once. Pierre-Simon Laplace applied this approach to tackle the sunrise problem; given that the sun has risen for n consecutive days up to today, what is the likelihood it will rise again tomorrow? This line of inquiry led Laplace to formulate the rule of succession, laying the essential groundwork for Bayesian statistics.

Let $\delta_1, \ldots, \delta_n$ be a sequence of cost differences observed during fuzzing. Let i be the index of a random variable that exceeds any previous cost differences, i.e., $\delta_i > \max\{\delta_1, \delta_{i-1}\}$. Following the Laplace estimator, with a probability $p = \frac{1}{j+1}$, we can stop fuzzing if and only if $\max_i\{\delta_{i+1}, \delta_{i+j}\} \leq \delta_i$. One can set j to 100 to bound the max cost difference with at most, 0.05 probability.

Exponentiality. This test utilized the Coefficient of Variation (CV) to determine whether the tail distribution is well-behaved. Algorithm 2 shows the steps in performing exponentiality testing. Specifically, the test goes over the k highest values of the cost differences and calculates the CV value where k ranges from k_{min} to k_{max}. If for all values of $k \in [k_{min}, k_{max}]$, the CV is less than $(1.0 + \frac{1}{4*k})$, then we are statistically confident that we have enough samples from the tail to infer a well-behaved tail distribution [45]. Note that the extra term $(\frac{1}{4*k})$ is to correct the bias in the estimation of CV due to small sample size in the tail [46]. Otherwise, if any values of CV are greater than $(1.0 + \frac{1}{4*k})$, we may not be able to infer a well-behaved distribution in the tail. Hence, we return `False`, it requires further fuzzing iterations.

B. Extrapolations via Tail Distributions (`WCDiff`)

Once a statistical testing of tail is convinced, the next step is to infer the tail distribution and estimate the return levels of worst-case cost differences.

Bayes Factor. Following the standard hypothesis testing, one can come up with two hypotheses where the null hypothesis is a predicate that the cost differences are below a threshold and

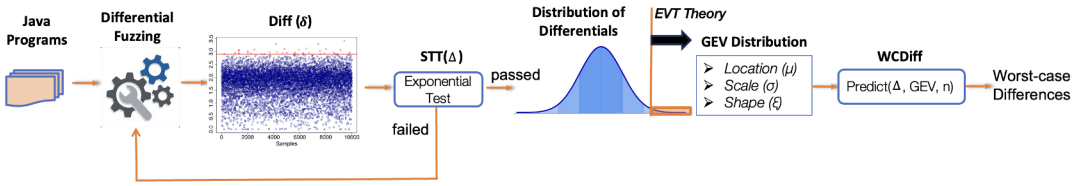


Fig. 4: The Conceptual Diagram of Algorithm 1: Steps to infer the worst-case differences.

Algorithm 3: Prediction of worst-case differentials via a generalized extreme value distribution.

Input: Differentials Δ, Threshold Finding Method T, Type of EVT Distribution $\mathcal{D}$

Threshold $\leftarrow$ 0.0
if T == *'BootStrap'* **then**
  $t \leftarrow$ `quantile`(Δ, seq(0.99, 0.75, by = -0.01))
  $\Delta_{bs} \leftarrow \lambda.\ t$ `sample`($\Delta[\Delta > t]$, R=`True`, 1000)
  $GPD \leftarrow \lambda.\ \Delta_{bs}$ `fit`('gpd', Δ_{bs})
  Threshold $\leftarrow \lambda.\ t, \Delta_{bs}$ $\min_t(GPD$.parameters.CI(Δ_{bs}))
$GEV \leftarrow$ `fevd`($dist$ = $\mathcal{D}$, pot = Threshold)
location, scale, shape $\leftarrow$ `distill`(GEV)
if *shape is valid* **then**
  Prediction $\leftarrow \lambda$.period `ReturnLevel`(location, scale, shape, period)
else
  Prediction $\leftarrow \max(\Delta)$
return Prediction

the alternative hypothesis is the negation of such predicate. The null and alternative hypotheses are

$$\mathcal{H}_0 : \mathbb{P}(\delta) \leq \tau, \quad \mathcal{H}_1 : \mathbb{P}(\delta) > \tau$$

where $\mathbb{P}(\delta)$ is the probability that a cost difference δ stays below a threshold τ, $\mathcal{H}_0$ is the null hypothesis, and $\mathcal{H}_1$ is the alternative. We say a Bayes factor has passed if we witness enough samples that are below the threshold $\leq$ to accept $\mathcal{H}_0$ as opposed to $\mathcal{H}_1$. There are multiple ways to conduct such statistical testing. The sequential probability ratio test and a Bayes factor are examples. We follow Jeffreys test [47], [48], a variant of Bayes factor, with a uniform prior to find a lower-bound on the number of successive samples K that sufficient for us to convince $\mathcal{H}_0$:

$$K \geq \lceil (-\log_2 B)/(\log_2 \theta) \rceil$$

where B is Bayes factor and can be set to 100 for very strong evidence. For instance, to achieve $\theta = 0.95$, we need to set $K \geq 90$ to accept $\mathcal{H}_0$.

Concentration Inequalities. One natural idea is to leverage concentration inequalities such as Markov's Inequality, Chebyshev's Inequality, and Höffding Bound [30]. Markov's Inequality states that for a non-negative random variable δ, the probability that it exceeds the expectation μ by a factor of k times are less than $\frac{1}{k}$, i.e., $Pr[\delta > k.\mu] \leq \frac{1}{k}$.

Similarly, Chebyshev's Inequality states that for a non-negative random variable δ with an expectation μ and variance σ^2 and for any real number $k > 0$, we have $\Pr(|\delta - \mu| \geq k\sigma) \leq \frac{1}{k^2}$. One critical limitation of these inequalities in our setting is the focus on the expectation and how much a random variable can exceed the expectation. While we are interested in modeling and reasoning about the tail of random variables; these concentration results do not explicitly model the tail distribution.

Rather than analyzing the concentration of differentials, we propose to explore the tail distribution. Algorithm 3 aims to predict worst-case differentials using EVT. The algorithm processes a dataset of observed differentials Δ to model the extreme values and estimate the maximum expected differential over specified periods. The algorithm also takes the method for selecting threshold T and the type of tail distribution $\mathcal{D}$ as inputs. If the threshold inferring method is "BootStrap", it proceeds with the bootstrap method to determine an optimal threshold value (any values above the threshold are extreme). The "BootStrap" generates a sequence of quantile thresholds t ranging from the 99th percentile to the 75th percentile of Δ, decreasing by 1% increments. This creates multiple candidate thresholds for EVT modeling. For each threshold t, it performs bootstrap sampling on the exceedances (data points where $\Delta[\Delta > t]$) and samples 1000 data points with replacement (bootstrap sampling) from the exceedances. Then, for each set of exceedances (based on the threshold t), we fit Generalized Pareto Distribution (GPD) over the samples and infer the parameters of distributions and their confidence intervals. Following the BootStrap method, if the parameter estimations for GPD are tight (a narrow confidence interval for a valid distribution), then it considers the corresponding threshold t as the optimal threshold that provides the best statistical fit. Finally, the algorithm fits a generalized extreme value (GEV) distributions to the data using a type of distribution ($\mathcal{D}$=Poisson Process, exponential, etc.) with the optimal threshold t as the peaks over threshold value. The algorithm extracts the parameters of GEV and validates its behavior based on its shape. In particular, if the shape is zero or negative ($\xi <= 0$), then the GEV belongs to the type I (exponential) or type III (light), and an extrapolation is feasible. Finally, the algorithm calculates the return levels of extreme values based

on the parameters of GEV distributions for a given return period (i.e., the number iterations in fuzzing). The outcome is the prediction for the worst-case cost differences.

VI. Experiments

A. Research Questions.

In this paper, we study the following research questions:

RQ1 What are the best statistical testing to stop fuzzing to predict the worst-case cost differences and infer a tail distribution of fuzzing process with ideal configurations of extreme value distributions?

RQ2 Do GEV distributions predict the worst-case cost differences better than the baseline such as Markov's Inequality [49] and Chebyshev's Inequality [50]?

RQ3 How accurate is the EVT in predicting the maximum differences on larger (real-world) Java libraries, and what are the efficiency characteristics?

B. Subjects

For RQ1, we use a set of micro-benchmark to infer ideal statistical techniques and their hyperparameters. In particular, we compare two statistical testing methods as stopping criterion during fuzzing, two threshold finding methods, and two types of EVT distributions over *Leak Set* programs that are leaking the number of set bits. In *Leak Set*, the size of secrets are from 12 to 28 bits for *Leak Set 1* to *Leak Set 5*, respectively. In RQ2, we compare our EVT-based extrapolations to three primary statistical methods. We used benchmarks from Blazer [51], Themis [52], and DifFuzz [15] that include vulnerabilities in *Eclipse Jetty* and *Apache WSS4J*[1]. Finally, RQ3 includes larger-scale Java libraries such as *Spring Security* and *Apache Ftpserver* where we used the best statistical techniques and their configurations to investigate the accuracy of our approach in predicting the worst-case differential fuzzing and evaluate the performance gain due to early stopping of fuzzing.

C. Technical Details

Experiments were ran on an Amazon AWS EC2 m5.large instance with Ubuntu $18.04.1\,LTS$ featuring $2x$ Intel(R) Xeon(R) CPU $X5365$ @ $3.00GHz$ with $8GB$ of memory, OpenJDK 1.8.0_422 and GCC 9.4.0. Following the setup for DifFuzz [15], we run each benchmark five times and report the mean and standard deviations of the results. We also run each of the experiments for 30 minutes. We apply Mann–Whitney U test [53] to establish that an error of approach 1 is statistically less than another and vice-versa.

D. RQ1 – Inferring ideal configurations for EVT prediction

Our goal is to infer which statistical testing during fuzzing provides an ideal sample set to infer the parameters of extreme value distributions. We modify DifFuzz to implement two methods of early stop: 1) Laplace and 2) Exponentiality. Our evaluation quantifies the error in the estimation of worst-case differential costs as compared to the ground truth. We note that while we set the training sample size to the time when the statistical testing for early stopping is satisfied, we continue fuzzing to record the ground-truth for our evaluations. Then, we use two techniques to infer thresholds of GEV distributions: 1) bootstrapping and 2) 0.95-Quantile. Finally, we consider two types of GEV distributions: 1) Exponential (Exp.) that represent the tail when shape is zero (i.e., it assumes a infinite tail, but decaying), and 2) Poisson Process (PP) that is valid for non-positive values of shape (i.e., the shape is finite). Therefore, we exclude invalid results that do not satisfy these invariants.

Table II shows the performance of different methods. We report the total number of test cases generated in 30 mins fuzzing campaign, the training (when the statistical testing passed during fuzzing) vs. testing (when we continue fuzzing after meeting the early stop criterion to collect the ground truth), the ground truth maximum cost differences, and the EVT-based prediction. We also report the most accurate predictions (over the repeated benchmarks) as well as the least error. We divide the table into different parts and use the least error to identify superior techniques. In doing so, we highlight any predictions that are within 5% of ground truth. Since we prefer over-approximation of ground-truth over the under-approximation, we also highlight any results that over-approximate the ground truth by at most 10%. For example, in *Leak Set (1)*, with Exponentiality testing, bootstrapping threshold, and PP distribution type; the best EVT extrapolation achieved 0.1% error, compared to the ground truth.

The results in Table II show that Exponentiality testing as a stopping criterion meets our conditions for accurate predictions in 17 cases whereas Laplace led to an accurate prediction of the ground truth in 13 cases (both out of 20 cases). Within the Exponentiality testing, we study the performance of bootstrapping vs. Quantile methods to pick threshold of GEV accurately. We observe that bootstrapping achieves better results than the Quantile method in 3 cases. Finally, we compare two types of GEV distributions, i.e., exponential vs. PP distributions. Our results show that both techniques have similar performance w.r.t. the best prediction. While the lowest error across all the different configurations that used Exponentiality testing, bootstrapping, and exponential distribution of GEV is +0.26%; the lowest error of benchmarks that used Exponentiality testing, bootstrapping, and Poisson Process distribution is +0.1%.

Answer RQ1: First, we find that Exponentiality testing, as a stopping criterion during differential fuzzing, leads to more accurate EVT-based extrapolations. Second, our experiments show that bootstrapping (as a method for inferring the threshold of extreme values) with Poisson Process (as a type of GEV distribution) slightly outperforms Quantile and exponential distribution.

E. RQ2 – Comparing the EVT to the baseline

The previous experiments convinced us that the Exponentiality stopping criterion, bootstrapping, and PP distribution is

[1] https://issues.apache.org/jira/browse/WSS-677

TABLE II: Comparing statistical methods for early stopping of fuzzing (Exponentiality vs. Laplace), selecting the threshold of GEV distributions (Bootstrap vs. Quantile), and the type of GEV distributions (Exponential vs. Poisson Process Distributions).

				GEV (Exponentiality Testing)						GEV (Laplace Testing)						
Benchmark	Num. Inputs	Threshold	Type	Training	Testing	Ground Truth	Prediction	Best(Prediction)	Best(Error%)	Threshold	Type	Training	Testing	Ground Truth	Prediction	Best(Prediction)	Best(Error%)
Leak Set (1)	13,186 (+/- 369)	Bootstrap	**Exp.**	**7.7k (+/-5.2k)**	**5.8k (+/-4.2k)**	**1.1k (+/-21.1)**	**1.8k (+/-0.7k)**	**1.1k**	**2.54**	Bootstrap	Exp.	107.3 (+/-23.0)	14.3k (+/-0.4k)	1.1k (+/-23.14)	3.4k (+/-2.1k)	629.0	41.1
			PP	**7.7k (+/-5.3k)**	**5.9k (+/-4.2k)**	**1.1k (+/-21.4)**	**1.1k (+/-0.3k)**	**1.1k**	**0.1**		PP	110 (+/- 26.5)	14.9k (+/-4.7k)	1.1k (+/-25.5)	1.3k (+/-1.2k)	841.6	21.2
		Quantile	**Exp.**	**2.0k (+/-0.6k)**	**1.2k (+/-2.7k)**	**1.1k (+/-21.1)**	**2.3k (+/-1.1k)**	**1.1k**	**3.69**	Quantile	Exp.	0.1k (+/-15.2)	13.5k (+/-3.0k)	1.1k (+/-17.6)	3.9k (+/-2.2k)	1.3k	20.0
			PP	**2.0k (+/-0.6k)**	**11.5k (+/-2.8k)**	**1.1 (+/-20.9)**	**1.0k (+/-0.3k)**	**1.1k**	**1.86**		**PP**	**100 (+/-0)**	**12.9k (+/-0.4k)**	**1.1k (+/-19.0)**	**1.0k (+/-0.5k)**	**1.1k**	**3.1**
Leak Set (2)	13,371 (+/- 168)	Bootstrap	**Exp.**	**6.6k (+/-5.9k)**	**7.0k (+/-5.0k)**	**1.4k (+/-19.91)**	**2.1k (+/-0.8)**	**1.5k**	**1.72**	Bootstrap	**Exp.**	**100(+/-0)**	**12.9k (+/-0.3k)**	**1.5k (+/-13.8)**	**2.6k (+/-1.3k)**	**1.5k**	**4.92**
			PP	**6.6k (+/-5.9k)**	**7.0k (+/-5.0k)**	**1.5k (+/-19.91)**	**1.4k (+/-0.4k)**	**1.5k**	**0.32**		**PP**	**100 (+/-0)**	**12.9k (+/-0.3k)**	**1.5k (+/-13.8)**	**9.9k (+/-25k)**	**1.5k**	**2.65**
		Quantile	Exp.	1.4k (+/-0.3k)	12.2k (+/-2.4k)	1.5k (+/-19.91)	3.7k (+/-0.9k)	1.3k	13.02	Quantile	**Exp.**	**108.5 (+/-33.4)**	**13.5k (+/-2.7k)**	**1.5k (+/- 22.9)**	**4.4k (+/-1.6k)**	**1.5k**	**1.58**
			PP	**1.4k (+/-0.3k)**	**12.2k (+/-2.5k)**	**1.5k (+/-19.91)**	**1.1k (+/-0.4k)**	**1.5k**	**8.37**		PP	112 (+/- 39.1)	13.7k (+/- 3.2k)	1.5k (+/-25.4)	42.k (+/-155.2k)	1157	18.75
Leak Set (3)	13,434 (+/- 307)	Bootstrap	**Exp.**	**9.7k (+/-13.5k)**	**5.8k (+/-5.0k)**	**1.7k (+/-82.6)**	**2.1k (+/-0.8k)**	**1.6k**	**1.06**	Bootstrap	**Exp.**	**104.2 (+/-13.4)**	**13.2k (+/-0.3k)**	**1.7k (+/-80.9)**	**4.5k (+/-2.0k)**	**1.9k**	**9.03**
			PP	**10.0k (+/-13.6k)**	**5.6k (+/-4.9k)**	**1.7k (+/-83.9)**	**1.5k (+/-0.3k)**	**1.6k**	**2.99**		**PP**	**105.6 (+/-15.3)**	**13.2k (+/-0.3k)**	**1.7k (+/-84.3)**	**462.2k (+/- 1.0m)**	**1.7k**	**2.47**
		Quantile	Exp.	1.3k (+/-0.3k)	14.3k (+/-11.7k)	1.7k (+/-84.2)	3.1k (+/-847.86)	1.6k	19.71	Quantile	Exp.	103.6 (+/- 11.7)	13.2k (+/-0.2k)	1.7k (+/-84.8)	4.4k (+/-2.0k)	1.5k	11.4
			PP	**1.3k (+/- 0.3k)**	**14.3k (+/-11.9k)**	**1.7k (+/-83.9)**	**1.3k (+/-0.3k)**	**1.8k**	**3.4**		**PP**	**104.1 (+/- 12.3)**	**13.2k (+/-0.2k)**	**1.7k (+/-85.5)**	**21.0k (+/-0.7k)**	**1.7k**	**2.15**
Leak Set (4)	13,157 (+/- 442)	Bootstrap	**Exp.**	**6.9k (+/-5.2k)**	**6.1k (+/-5.2k)**	**2.1k (+/-0.1k)**	**2.3k (+/-0.8k)**	**2.2k**	**1.38**	Bootstrap	**Exp.**	**0.1k (+/- 26.8)**	**13.0k (+/-0.4k)**	**2.1k (+/-87.8)**	**4.6k (+/-1.8k)**	**2.2k**	**5.56**
			PP	**6.8k (+/-5.3k)**	**6.3k (+/-5.2k)**	**2.1k (+/-0.1k)**	**2.0k (+/-0.9k)**	**2.2k**	**1.06**		PP	0.1k (+/-30.5)	13.0k (+/-0.4k)	2.1k (+/-63.0)	9.3k (+/-12.5k)	3.0k	37.07
		Quantile	**Exp.**	**1.1k (+/-0.2k)**	**12.0k (+/-0.4k)**	**2.1k (+/-0.1k)**	**3.0k (+/-0.8k)**	**2.1k**	**0.53**	Quantile	**Exp.**	**0.1k (+/- 0)**	**13.1k (+/- 0.4k)**	**2.1k (+/- 0.1k)**	**6.1k (+/-1.7k)**	**2.2k**	**1.78**
			PP	1.1k (+/-0.2k)	12.0k (+/-0.4k)	2.1k (+/-0.1k)	1.7k (+/-0.7k)	1.9k	13.9		**PP**	**100 (+/- 0)**	**13.1k (+/-0.4k)**	**2.1k (+/-0.1k)**	**6.1k (+/-1.7k)**	**2.2k**	**1.78**
Leak Set (5)	13,205 (+/- 456)	Bootstrap	**Exp.**	**6.4k (+/-5.3k)**	**6.6k (+/-5.5k)**	**2.4k (+/-0.1k)**	**2.7k(+/-0.7k)**	**2.4k**	**0.26**	Bootstrap	**Exp.**	**0.1k (+/- 13.6)**	**12.9k (+/- 0.4k)**	**2.4k (+/- 0.1k)**	**5.3k (+/- 3.4k)**	**2.4k**	**3.89**
			PP	**6.4k (+/-5.3k)**	**6.6k (+/-5.5k)**	**2.4k (+/-0.1k)**	**2.7k (+/-0.7k)**	**2.4k**	**2.64**		PP	0.1k (+/-16.5)	12.9k (+/- 0.5k)	2.4k (+/- 0.1k)	5.3k (+/- 3.4k)	2.4k	24.93
		Quantile	**Exp.**	**0.9k (+/-0.2k)**	**12.1k (+/-0.4k)**	**2.4k (+/-0.1k)**	**2.8k (+/-0.4k)**	**2.6k**	**4.74**	Quantile	**Exp.**	**0.1k (+/- 0)**	**12.9k (+/- 0.4k)**	**2.4k (+/- 0.1k)**	**5.5k (+/- 2.5k)**	**2.3k**	**5.08**
			PP	**0.9k (+/-0.2k)**	**12.1k (+/- 0.4k)**	**2.4k (+/-0.1k)**	**1.9k (+/-0.7k)**	**2.4k**	**0.42**		**PP**	**100 (+/- 0)**	**12.9k (+/- 0.4k)**	**2.4k (+/-0.1k)**	**8.4k (+/-20.4k)**	**2.2k**	**7.62**

TABLE III: Comparison of EVT-based extrapolations with the baseline methods in predicting the worst-case cost differences. The highlighted values are winners based on the Mann–Whitney U-test.

Benchmark	Num. Inputs	Extrapolation	Training	Max. Training	Testing	Max. Testing	Prediction	Error %
Leak Set (1)	13186.0 (+/- 369.2)	Markov	1,200 (+/- 0.0)	744.2 (+/- 33.41)	11985.0 (+/- 369.2)	1089.6 (+/- 19.7)	7266.0 (+/- 4561.2)	565.2 (+/- 410.32)
		Chebyshev	1,200 (+/- 0.0)	744.2 (+/- 33.4)	11985.0 (+/- 369.2)	1089.6 (+/- 19.7)	1660.9 (+/- 171.3)	52.5 (+/- 16.0)
		Bayes Factor	**6114.0 (+/- 1864.4)**	**1039.0 (+/- 41.4)**	**19774.2 (+/- 5365.2)**	**1089.6 (+/- 19.7)**	**1039.0 (+/- 41.4)**	**-4.58 (+/- 5.08)**
		EVT	8999.2 (+/- 3155.4)	1075.2 (+/- 16.1)	4211.2 (+/- 3159.0)	1089.6 (+/- 19.7)	1240.5 (+/- 507.3)	13.5 (+/- 45.2)
Leak Set (2)	13371.2 (+/- 167.7)	Markov	1,200 (+/- 0.0)	975.2 (+/- 123.4)	12170.2 (+/- 167.7)	1462.4 (+/- 21.5)	12673.7 (+/- 1461.7)	766.66 (+/- 98.73)
		Chebyshev	**1,200 (+/- 0.0)**	**975.2 (+/- 123.4)**	**12170.2 (+/- 167.7)**	**1462.4 (+/- 21.5)**	**1457.6 (+/- 280.9)**	**-0.31 (+/- 19.15)**
		Bayes Factor	**5706.0 (+/- 980.7)**	**1299.4 (+/- 148.9)**	**18910.0 (+/- 3348.2)**	**1462.4 (+/- 21.5)**	**1299.4 (+/- 148.9)**	**-11.11 (+/- 10.39)**
		EVT	**8201 (+/- 1795.4)**	**1414.8 (+/- 68.6)**	**5168.5 (+/- 1963.4)**	**1460 (+/- 24)**	**1505.3 (+/- 121.2)**	**3.2 (+/- 9.2)**
Leak Set (3)	13433.6 (+/- 307.3)	Markov	1,200 (+/- 0.0)	1214.4 (+/- 77.0)	12232.6 (+/- 307.3)	1717.6 (+/- 108.6)	37161.3 (+/- 1477.3)	2068.79 (+/- 129.7)
		Chebyshev	**1,200 (+/- 0.0)**	**1214.4 (+/- 77.0)**	**12232.6 (+/- 307.3)**	**1717.6 (+/- 108.6)**	**1606.9 (+/- 342.6)**	**-5.99 (+/- 22.24)**
		Bayes Factor	**7440.0 (+/- 1834.7)**	**1425.0 (+/- 129.2)**	**24387.4 (+/- 6529.6)**	**1717.6 (+/- 108.6)**	**1425.0 (+/- 129.2)**	**-16.86 (+/- 7.9)**
		EVT	**5766 (+/- 2522.6)**	**1525.8 (+/- 151.2)**	**7691.4 (+/- 2735.2)**	**1717.6 (+/- 108.6)**	**1508.4 (+/- 218.1)**	**-12.45 (+/- 8.9)**
Leak Set (4)	13156.8 (+/- 442.1)	Markov	1,200 (+/- 0.0)	1380.0 (+/- 243.4)	11955.8 (+/- 442.1)	2189.6 (+/- 41.1)	55868.0 (+/- 1719.0)	2452.42 (+/- 97.72)
		Chebyshev	**1,200 (+/- 0.0)**	**1380.0 (+/- 243.4)**	**11955.8 (+/- 442.1)**	**2189.6 (+/- 41.1)**	**1964.6 (+/- 608.4)**	**-10.31 (+/- 27.38)**
		Bayes Factor	**5808.0 (+/- 524.2)**	**1720.0 (+/- 268.0)**	**18720.0 (+/- 2103.9)**	**2189.6 (+/- 41.1)**	**1720.0 (+/- 268.0)**	**-21.54 (+/- 11.41)**
		EVT	**8791.4 (+/- 2171.1)**	**2047.0 (+/- 39.8)**	**4392.6 (+/- 1822.9)**	**2189.6 (+/- 41.1)**	**2129.4 (+/- 559.6)**	**-2.83 (+/- 24.96)**
Leak Set (5)	13205.0 (+/- 456.1)	Markov	1,200 (+/- 0.0)	1711.2 (+/- 167.1)	12004.0 (+/- 456.1)	2412.0 (+/- 121.2)	79846.01 (+/- 1538.7)	3218.5 (+/- 210.06)
		Chebyshev	**1,200 (+/- 0.0)**	**1711.2 (+/- 167.1)**	**12004.0 (+/- 456.1)**	**2412.0 (+/- 121.2)**	**2527.8 (+/- 280.5)**	**4.75 (+/- 9.47)**
		Bayes Factor	6252.0 (+/- 1015.5)	1748.0 (+/- 130.1)	20149.8 (+/- 2527.6)	2412.0 (+/- 121.2)	1748.0 (+/- 130.1)	-27.22 (+/- 8.73)
		EVT	9741.2 (+/- 2058.2)	2208 (+/- 195.2)	3496.4 (+/- 2350.5)	2412 (+/- 121.2)	2014.8 (+/- 172.0)	-16.48 (+/- 5.43)
Array Unsafe	10567.8 (+/- 313.8)	Markov	1,200 (+/- 0.0)	192.0 (+/- 0.0)	9366.8 (+/- 313.8)	192.0 (+/- 0.0)	4489.1 (+/- 821.2)	2238.05 (+/- 427.71)
		Chebyshev	1,200 (+/- 0.0)	192.0 (+/- 0.0)	9366.8 (+/- 313.8)	192.0 (+/- 0.0)	853.4 (+/- 56.7)	344.47 (+/- 29.52)
		Bayes Factor	**4410.0 (+/- 458.4)**	**192.0 (+/- 0.0)**	**10833.4 (+/- 1387.3)**	**192.0 (+/- 0.0)**	**192.0 (+/- 0.0)**	**0.0 (+/- 0.0)**
		EVT	**6955 (+/- 1004.1)**	**192.0 (+/- 0.0)**	**3797 (+/- 783.5)**	**192 (+/- 0.0)**	**192 (+/- 0.0)**	**0.0 (+/- 0.0)**
gpt14 Unsafe	13156.0 (+/- 226.9)	Markov	1,200 (+/- 0.0)	4692.4 (+/- 1229.2)	11955.0 (+/- 226.9)	6389.6 (+/- 455.6)	181668.2 (+/- 2512.1)	2755.44 (+/- 219.03)
		Chebyshev	**1,200 (+/- 0.0)**	**4692.4 (+/- 1229.2)**	**11955.0 (+/- 226.9)**	**6389.6 (+/- 455.6)**	**7284.5 (+/- 783.0)**	**14.21 (+/- 11.53)**
		Bayes Factor	**5898.0 (+/- 920.4)**	**5470.2 (+/- 548.0)**	**18927.8 (+/- 2430.7)**	**6389.6 (+/- 455.6)**	**5470.2 (+/- 548.0)**	**-14.42 (+/- 5.28)**
		EVT	**9266.6 (+/- 3785.6)**	**5795.6 (+/- 469.8)**	**4210.8 (+/- 3855.4)**	**6389.6 (+/- 455.6)**	**5257.2 (+/- 682.3)**	**-17.12 (+/- 14)**
k96 Unsafe	13596.8 (+/- 954.8)	Markov	1,200 (+/- 0.0)	3867.6 (+/- 743.7)	12395.8 (+/- 954.8)	5291.8 (+/- 577.1)	83100.2 (+/- 2444.8)	1485.42 (+/- 180.95)
		Chebyshev	**1,200 (+/- 0.0)**	**3867.6 (+/- 743.7)**	**12395.8 (+/- 954.8)**	**5291.8 (+/- 577.1)**	**3680.6 (+/- 459.2)**	**-29.92 (+/- 10.42)**
		Bayes Factor	**5220.0 (+/- 1014.5)**	**3939.6 (+/- 717.9)**	**17812.0 (+/- 3640.0)**	**5291.8 (+/- 577.1)**	**3939.6 (+/- 717.9)**	**-25.85 (+/- 9.19)**
		EVT	**10177.4 (+/- 1455.9)**	**4321.2 (+/- 275.9)**	**3667.4 (+/- 1732.1)**	**5291.8 (+/- 577.1)**	**6723.9 (+/- 6050.7)**	**25.29 (+/- 105.26)**
login Unsafe	12804.6 (+/- 389.4)	Markov	1,200 (+/- 0.0)	6.4 (+/- 6.1)	11603.6 (+/- 389.4)	62.0 (+/- 0.0)	93.9 (+/- 86.8)	51.38 (+/- 140.07)
		Chebyshev	1,200 (+/- 0.0)	6.4 (+/- 6.1)	11603.6 (+/- 389.4)	62.0 (+/- 0.0)	17.0 (+/- 13.6)	-72.56 (+/- 21.9)
		Bayes Factor	6660.0 (+/- 901.0)	12.0 (+/- 8.0)	20684.8 (+/- 3214.7)	62.0 (+/- 0.0)	12.0 (+/- 8.0)	-80.65 (+/- 12.9))
		EVT	**6670.8 (+/- 987.3)**	**61.2 (+/- 1.8)**	**6135.8 (+/- 1047.4)**	**62 (+/- 0)**	**84.1 (+/- 18.8)**	**35.64 (+/- 30.31)**
modPow1 Unsafe	12392.8 (+/- 352.2)	Markov	1,200 (+/- 0.0)	1624.0 (+/- 246.1)	11191.8 (+/- 352.2)	2543.2 (+/- 328.6)	38153.3 (+/- 1728.0)	1412.79 (+/- 122.83)
		Chebyshev	**1,200 (+/- 0.0)**	**1624.0 (+/- 246.07)**	**11191.8 (+/- 352.2)**	**2543.2 (+/- 328.6)**	**2359.0 (+/- 320.9)**	**-6.66 (+/- 12.68)**
		Bayes Factor	5946.0 (+/- 795.5)	2002.4 (+/- 157.0)	17510.0 (+/- 1902.8)	2543.2 (+/- 328.6)	2002.4 (+/- 157.0)	-20.03 (+/- 13.53)
		EVT	8234.8 (+/- 1489.4)	2214.8 (+/- 107.0)	4235.4 (+/- 1149.7)	2543.2 (+/- 328.6)	2034.6 (+/- 162.1)	-19.37 (+/- 8.18)
modPow2 Unsafe	14782.0 (+/- 2535.4)	Markov	1,200 (+/- 0.0)	36.8 (+/- 46.4)	13581.0 (+/- 2535.4)	143.8 (+/- 11.3)	46.5 (+/- 63.1)	-68.24 (+/- 42.65)
		Chebyshev	1,200 (+/- 0.0)	36.8 (+/- 46.4)	13581.0 (+/- 2535.4)	143.8 (+/- 11.3)	38.3 (+/- 49.8)	-73.79 (+/- 33.63)
		Bayes Factor	7014.0 (+/- 2442.9)	81.2 (+/- 49.3)	27784.8 (+/- 16389.9)	143.8 (+/- 11.3)	81.2 (+/- 49.3)	-44.70 (+/- 31.05)
		EVT	**6652.6 (+/- 3457.4)**	**121.6 (+/- 17.6)**	**8143 (+/- 5511.8)**	**143.8 (+/- 11.3)**	**135.2 (+/- 57.2)**	**-6.87 (+/- 36.41)**
passwordEq Unsafe	12719.6 (+/- 172.0)	Markov	1,200 (+/- 0.0)	56.0 (+/- 15.6)	11518.6 (+/- 172.0)	63.0 (+/- 0.0)	2175.5 (+/- 670.6)	3353.18 (+/- 1064.47)
		Chebyshev	1,200 (+/- 0.0)	56.0 (+/- 15.6)	11518.6 (+/- 172.01)	63.0 (+/- 0.0)	239.1 (+/- 82.6)	279.56 (+/- 131.13)
		Bayes Factor	**5532.0 (+/- 1287.8)**	**60.8 (+/- 4.9)**	**17368.0 (+/- 4021.1)**	**63.0 (+/- 0.0)**	**60.8 (+/- 4.9)**	**-3.5 (+/- 7.81)**
		EVT	**5798 (+/- 258.8)**	**63 (+/- 0)**	**6816 (+/- 188.1)**	**63 (+/- 0)**	**69.6 (+/- 9.4)**	**10.52 (+/- 14.9)**
sanity Unsafe	1464.4 (+/- 234.1)	Markov	1,200 (+/- 0.0)	2867086184.5 (+/- 351155148.48)	329.8 (+/- 209.1)	2774479762.0 (+/- 419417121.7)	80645893388.6 (+/- 17819138629.2)	2606.11 (+/- 971.09)
		Chebyshev	1,200 (+/- 0.0)	2867086184.5 (+/- 351155148.5)	329.8 (+/- 209.1)	2774479762.0 (+/- 419417121.7)	12611863196.6 (+/- 832399251.3)	317.05 (+/- 81.37)
		Bayes Factor	**2234.0 (+/- 333.6)**	**2806659155.4 (+/- 332775574.5)**	**0.0 (+/- 0.0)**	**2806659155.4 (+/- 332775574.5)**	**2806659155.4 (+/- 332775574.5)**	**0.0 (+/- 0.0)**
		EVT	**2234.0 (+/- 333.6)**	**2806659155.4 (+/- 332775574.5)**	**0.0 (+/- 0.0)**	**2806659155.4 (+/- 332775574.5)**	**2806659155.4 (+/- 332775574.5)**	**0.0 (+/- 0.0)**
straightline Unsafe	14207.2 (+/- 96.0)	Markov	1,200 (+/- 0.0)	8.0 (+/- 0.0)	13006.2 (+/- 96.0)	8.0 (+/- 0.0)	152.8 (+/- 65.3)	1810.07 (+/- 816.47)
		Chebyshev	1,200 (+/- 0.0)	8.0 (+/- 0.0)	13006.2 (+/- 96.0)	8.0 (+/- 0.0)	31.7 (+/- 8.2)	296.09 (+/- 103.02)
		Bayes Factor	**4788.0 (+/- 989.0)**	**8.0 (+/- 0.0)**	**17432.8 (+/- 3461.0)**	**8.0 (+/- 0.0)**	**8.0 (+/- 0.0)**	**0.0 (+/- 0.0)**
		EVT	**4788.0 (+/- 989.0)**	**8.0 (+/- 0.0)**	**17432.8 (+/- 3461.0)**	**8.0 (+/- 0.0)**	**8.0 (+/- 0.0)**	**0.0 (+/- 0.0)**

one of the best configurations to predict the worst-case cost differences via EVT. In this section, we compare our extrapolation approach via EVT to three baseline statistical techniques; Markov's Inequality [49], Chebyshev's Inequality [50], and Bayes Factor (following Jeffery's Test [47]). Following the standard requirements to achieve a confidence of $1 - c$ with an error probability below α of estimation for concentration inequalities, we need a training sample size of $O(\frac{ln(1/c)}{\alpha})$. We set the training size to at least $1,200$ to achieve a 0.95 confidence with an error probability 0.05. Following common practices, we used Laplace testing to determine the sample size for Bayes Factor.

Table III show the results of comparing the EVT extrapolations to Markov, Chebyshev, and Bayes factor. The statistically significant results based on Mann-Whitney U-test are highlighted in bold. When there is more than one highlighted row for a benchmark, it means two or more techniques are tied. Overall, EVT-based extrapolations are highlighted in 11 cases out of 14 benchmarks while Bayes factor is highlighted in 10 cases out of 14 benchmarks. The Chebyshev is highlighted in 7 cases out of 14 benchmarks, but Markov did not win in any cases. When comparing EVT to Bayes factor, we observe that EVT wins in 2 cases, losses in 1, and ties in 9 cases. However, the winning case for the Bayes factor (i.e., Leak Set 1) is an underestimation that may still miss bugs. Comparing EVT to Chebyshev, we see that EVT wins, losses, and ties in 5, 2, and

5 cases.

Specifically, both EVT-based extrapolations and Bayes factor perform very well for 3 cases of "Array Unsafe", "Sanity Unsafe", and "Straightline Unsafe" (0% error); but Chebyshev has some errors in all benchmarks. The dynamic training sizes used by the Bayes factor (via Laplace testing) and EVT (via Exponentiality testing) seem more effective than the fixed training size for Markov/Chebyshev. Since tight over-approximations are preferred, we observe that in 5 out of 11 remaining benchmarks, EVT-based extrapolations provide a tighter over-approximation than Bayes factor. There are no cases where Bayes factor outperforms EVT-based extrapolations in terms of tight over-approximation.

Answer RQ2: EVT-based extrapolations outperform the most competitive baseline (Bayes Factor via Jeffery's Test [47]). In 57.1% of the cases, EVT-based extrapolations provide a tight over-approximation, while the Bayes Factor under-approximates the ground truth in 78.6% of cases.

F. RQ3 – Measuring the accuracy and performance gain of EVT-enabled Differential Fuzzing in realistic Java libraries

Table IV shows the results of EVT-enabled differential fuzzing in larger Java benchmarks. The positive errors in all cases show that EVT-based extrapolations provide over-approximation, so it does not underestimate the worst-case cost differences. The error ranges from 32.8% (Stateless Authenticated) to 206.8% (Apache FtpServer Stringutils). In cases when maximum of unobserved test data is higher than the maximum of observed training data, the EVT prediction provides very close prediction (e.g., 51 vs. 52 for Jetty, 341 vs 381 for Tourplanner, and 143 vs 175 for Apache Ftpserver).

When the max. cost difference is the same between observed training and unseen testing data, the EVT prediction still results in an over-approximation (positive error). This means that even when the maximum observed cost during training is the true maximum, the EVT model is still projecting a tail that extends beyond this observed maximum. This is inherent to how EVT models tail behavior; it extrapolates beyond the observed data.

Finally, RQ3 includes larger-scale Java libraries such as *Spring Security* and *Apache Ftpserver*. We also calculate the performance gain of EVT-enabled differential fuzzing over baseline methods in terms of bytecode execution saved by early stopping of fuzzing campaigns. We report the performance gain of early return in the last column of Table IV. The results shows that in one case for *Apache Ftpserver Salted*, 1,674,774,946 bytecode executions has been saved. Since we run all the benchmarks for 30 mins, the ratio of $\frac{|Testing|}{|Testing|+|Training|}$ approximately shows the wall clock savings (e.g., 0.7*30 $\approx$ 21 mins saved for Stateless Auth). Finding a trade-off between the accuracy of the prediction (the "Error" column) vs. the performance gain is an interesting direction for future work.

Finally, for those cases with larger errors in the prediction, we also notice that the Scale parameter of the EVT distribution is large. This can be used to guide a search algorithm to potentially find better threshold of extreme values.

Answer RQ3: We find that EVT-based differential fuzzing does not underestimate the worst-case cost differences in any larger Java libraries. The error ranges from 32.8% (Stateless Authenticated) to 206.8% (Apache FtpServer Stringutils). In 4 out of 9 cases, EVT-based differential fuzzing provides a tight upper-bound of the worst-case cost differences. We also report the significant performance gain of fuzzing when early stopping and extrapolation via EVT is applied.

VII. Discussion

The choice of extreme value theory over other potential approaches like machine learning-based extrapolation was deliberate. EVT provides a strong theoretical foundation specifically designed for modeling extreme events and tail behavior. Machine learning approaches, while flexible, typically focus on modeling the average case behavior rather than extremes, and may require significantly more training data to make reliable predictions about rare events. We focused our baseline comparisons on classical statistical methods (Markov, Chebyshev, Bayes factor) as they provide theoretical bounds with clear probabilistic interpretations. While other approaches like linear regression or more sophisticated time series models could be considered, they generally make stronger assumptions about the underlying distribution and may not be well-suited for modeling extreme events.

Limitation. Our EVT model of differential fuzzing is limited to the specific program under test and the ongoing fuzzing campaign. The model's purpose is not to predict a program's absolute worst-case difference, but to predict the worst-case difference likely to be discovered by the current fuzzing.

Differential Metrics. Our framework supports differential metrics that are quantitative, ordinal, and have a sufficiently rich value space in its upper tail. This makes our approach broadly applicable to a range of common differential metrics beyond Java bytecode counts. For instance, it generalizes directly to performance testing (measuring execution time differences in microseconds) and resource consumption analysis (e.g., peak memory usage differences in bytes). In essence, any scenario where the "worst-case" is characterized by the magnitude of a numeric difference falls within the ideal application scope of our method.

The efficacy of our approach degrades as the differential metric becomes more discrete. In the extreme case of a binary metric (e.g., passed vs. failed), the concept of an extreme magnitude collapses. In this scenario, our EVT framework pivots from modeling how large the next difference will be to modeling the waiting time until the next (failed) event. This makes the problem conceptually similar to prior works that use Bernoulli or Poisson statistics to model bug discovery rates. However, a key distinction remains: while prior works bound the average discovery rate, our framework would model the tail distribution of the inter-arrival times between events.

TABLE IV: Evaluation of EVT-enabled DIFFUZZ in larger Java libraries.

Benchmark	Num. Inputs	Training	Max. Training	Testing	Max. Testing	Scale	Prediction	Error %	Performance Gain
Stateless Auth	12685.4 (+/- 703.9)	3702.8 (+/- 1067.5)	101 (+/- 0.0)	8984.6 (+/- 1550.6)	101 (+/- 0.0)	5.9 (+/- 3.7)	134.1 (+/- 18.4)	32.8 (+/- 18.2)	2,300,058
Jetty Safe	13934.4 (+/- 185.0)	532 (+/- 49.1)	27.6 (+/- 2.2)	13407.6 (+/- 229.8)	54 (+/- 7.6)	7.4 (+/- 1.8)	75.1 (+/- 19.9)	39.0 (+/- 43.7)	1,241,133
Jetty	12649.4 (+/- 620.6)	6403.2 (+/- 470.83)	51.4 (+/- 1.3)	6250.2 (+/- 334.9)	52 (+/- 0.0)	10.8 (+/- 8.0)	85.9 (+/- 61.4)	65.2 (+/- 118.0)	80,741
Orientdb	13963 (+/- 556.6)	5605.8 (+/- 1442.3)	47 (+/- 0.0)	8359.2 (+/- 1741.4)	47 (+/- 0.0)	13.63 (+/- 10.7)	137.8 (+/- 73.5)	193.2 (+/- 156.3)	234,058
Picketbox	13527.8 (+/- 771.0)	7479 (+/- 1458.6)	30.8 (+/- 0.45)	6050.4 (+/- 994.6)	31 (+/- 0.0)	7.32 (+/- 2.4)	72.1 (+/- 11.3)	132.7 (+/- 36.4)	193,613
Spring Security	13807.6 (+/- 324.3)	10419 (+/- 2855.2)	149 (+/- 0.0)	3328 (+/- 2472.5)	149 (+/- 0.0)	38.2 (+/- 18.6)	333.7 (+/- 142.2)	124.0 (+/- 95.4)	479,232
Tourplanner	7465.2 (+/- 153.1)	3205 (+/- 1805.1)	341.2 (+/- 37.4)	4290 (+/- 1830.0)	380.8 (+/- 28.0)	79.6 (+/- 83.7)	784.7 (+/- 547.3)	106.1 (+/- 160.8)	1,434,977
Apache Ftpserver Salted	13347.4 (+/- 178.7)	5245 (+/- 4188.7)	143.2 (+/- 33.5)	8125.4 (+/- 4195.3)	175 (+/- 18.6)	25.2 (+/- 10.4)	255.1 (+/- 78.4)	45.793 (+/- 28.1)	1,674,774,946
Apache Ftpserver Stringutils	10295.8 (+/- 1278.7)	504.4 (+/- 103.7)	53 (+/- 0.0)	9793.4 (+/- 1264.5)	53 (+/- 0.0)	15.3 (+/- 2.0)	162.6 (+/- 18.3)	206.8 (+/- 34.5)	571,164

Non-i.i.d. Setting. Another core assumption of classical EVT is that samples are independent and identically distributed (i.i.d.). This assumption is violated in a graybox fuzzing setting. We mitigate this challenge in two ways. First, we hypothesize that after an initial warm-up phase, the fuzzing process enters a relatively stationary state, where the underlying distribution of observed differences becomes more stable. This addresses the "identically distributed" aspect. Second, to handle the lack of independence, we employ a block bootstrapping technique based on recent work for dependent data [54]. This method resamples blocks of consecutive observations rather than individual points, preserving the local dependency structure within the samples. While this is a practical mitigation rather than a perfect theoretical fix, it is a standard approach [55], [56], [57] for applying statistical models to dependent data where true independence cannot be guaranteed.

Overheads. The overhead of our method is primarily related to bootstrapping (Algorithm 3), while the exponentiality test (Algorithm 2) is a lightweight statistical test. The bootstrapping step can take up to 6 minutes to complete per fuzzing campaign. However, this cost is easily justified. The expensive inference is a one-time cost that occurs only once per campaign, right before making a stopping decision. As our results show (Table IV), this one-time cost is far outweighed by the significant savings from early termination.

Threats to Validity. To address *internal validity* concerns and account for the stochastic nature of fuzzing, we adhered to best practices outlined in previous work [58], [59] for rigorous fuzzing evaluation. In particular, we repeated the experiments 30 times, showed the error margins, performed experiments with multiple seed inputs, and considered not only the final results, but also the temporal development. An area of weakness that requires further research is the relatively high variance in the outcomes of EVT. We chose Exponential and Poisson Process as they are direct and principled applications of the Peaks-Over-Threshold methodology in EVT. The occurrences of events exceeding a high threshold can be modeled as a Poisson Process. Similarly, the size of these excesses (how much larger they are than the threshold) is theoretically modeled by the Exponential distribution.

To address *external validity* concerns, we evaluated our approach on a diverse set of benchmarks ranging from small micro-benchmarks to large real-world Java applications. Our subjects included programs from established benchmark suites like BLAZER [51], THEMIS [52], and DIFFUZZ [15], as well as widely-used libraries such as Spring Security and Apache FtpServer. This variety helps demonstrate that our EVT-based predictions generalize across different program sizes and complexity levels. To show the applicability beyond DIFFUZZ, we explore the use of another differential fuzzing framework, called QFuzz [16]. Instead of byte-code cost differences, we leverage our EVT approach to predict the number of clusters inferred by QFuzz. We consider the leak set and password matching programs and run QFuzz on each benchmark for 30 minutes (repeated 5 times with different seeds). The results are presented in Table V. The average error in using the proposed EVT-based stopping criteria for QFuzz is below 1%.

TABLE V: EVT-based prediction of clustering in QFUZZ [16]

Benchmark	Training	Testing	Ground Truth	Prediction	Error%
Leaky (1)	182 (+/-46.23)	1618 (+/-46.23)	13 (+/-0)	12.96 (+/-0)	-0.27 (+/-0.02)
Leaky (2)	893.25 (+/-169.56)	906.75 (+/-169.56)	16.99 (+/-0.01)	17.1 (+/-0.27)	0.65 (+/-1.54)
Leaky (3)	1701 (+/-113.2)	99 (+/-113.2)	20.89 (+/-0.03)	20.9 (+/-0.12)	0.07 (+/-0.52)
Leaky (4)	1756.4 (+/-36.69)	43.6 (+/-36.69)	24.5 (+/-0.1)	24.44 (+/-0.11)	-0.25 (+/-0.06)
Leaky (5)	1749.6 (+/-29.68)	50.4 (+/-29.68)	27.66 (+/-0.38)	27.6 (+/-0.38)	-0.22 (+/-0.02)
Password Matching	1667 (+/-146.14)	133 (+/-146.14)	16.81 (+/-0.22)	16.7 (+/-0.27)	-0.65 (+/-0.32)

VIII. CONCLUSION

In this paper, we explored the application of Extreme Value Theory (EVT) to provide statistical guarantees on the worst-case divergence in differential fuzzing. We adapted EVT to model the maximum cost differences of differential fuzzing and analyzed the tail distribution of these differences. Through extensive experiments on real-world Java libraries and web servers, we demonstrate that EVT can effectively predict the maximum cost differences in side-channel analysis. It also outperformed the baseline statistical methods. Finally, we showed EVT-enabled differential fuzzing can provide significant performance gains through early termination.

There are multiple interesting directions for future work. First, we plan to extend our approach to other differential testing domains, such as machine learning models and libraries. Second, we plan to explore the optimal hyperparameters for EVT distributions and the optimal termination condition for differential fuzzing campaigns.

Acknowledgment. The authors thank the anonymous ASE reviewers for their time and invaluable feedback to improve this work. This project has been supported by NSF under Grant No. CNS-2230060 and CNS-2527657.

References

[1] H. Krasner, "The cost of poor software quality in the us: A 2020 report," *Proc. Consortium Inf. Softw. QualityTM (CISQTM)*, vol. 2, 2021.

[2] M. Böhme, C. Cadar, and A. Roychoudhury, "Fuzzing: Challenges and reflections," *IEEE Software*, vol. 38, no. 3, pp. 79–86, 2020.

[3] G. Klees, A. Ruef, B. Cooper, S. Wei, and M. Hicks, "Evaluating fuzz testing," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2123–2138. [Online]. Available: https://doi.org/10.1145/3243734.3243804

[4] M. Böhme, "Stads: Software testing as species discovery," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 2, pp. 1–52, 2018.

[5] M. Böhme, D. Liyanage, and V. Wüstholz, "Estimating residual risk in greybox fuzzing," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 230–241.

[6] S. Lee and M. Böhme, "Statistical reachability analysis," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2023, 2023, p. 12.

[7] S. Saha, M. Downing, T. Brennan, and T. Bultan, "Preach: a heuristic for probabilistic reachability to identify hard to reach statements," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1706–1717.

[8] S. Saha, L. Sarker, M. Shafiuzzaman, C. Shou, A. Li, G. Sankaran, and T. Bultan, "Rare path guided fuzzing," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 1295–1306.

[9] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, Sept 2016.

[10] T. Y. Chen, S. C. Cheung, and S. Yiu, "Metamorphic Testing: A New Approach for Generating Next Test Cases," Tech. Rep. HKUST-CS98-01, 1998. [Online]. Available: https://arxiv.org/abs/2002.12543

[11] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[12] S. Tizpaz-Niari, V. Monjezi, M. Wagner, S. Darian, K. Reed, and A. Trivedi, "Metamorphic testing and debugging of tax preparation software," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2023, pp. 138–149.

[13] S. Tizpaz-Niari, P. Černỳ, and A. Trivedi, "Detecting and understanding real-world differential performance bugs in machine learning libraries," in *Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis*, 2020, pp. 189–199.

[14] S. Tizpaz-Niari, P. Cerny, B.-Y. E. Chang, and A. Trivedi, "Differential performance debugging with discriminant regression trees," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[15] S. Nilizadeh, Y. Noller, and C. S. Pasareanu, "Diffuzz: Differential fuzzing for side-channel analysis," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 176–187.

[16] Y. Noller and S. Tizpaz-Niari, "Qfuzz: Quantitative fuzzing for side channels," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 257–269.

[17] S. Tizpaz-Niari, P. Černỳ, and A. Trivedi, "Quantitative mitigation of timing side channels," in *International conference on computer aided verification*. Springer, 2019, pp. 140–160.

[18] S. Tizpaz-Niari, P. Černỳ, S. Sankaranarayanan, and A. Trivedi, "Efficient detection and quantification of timing leaks with neural networks," in *International Conference on Runtime Verification*. Springer, 2019, pp. 329–348.

[19] H. Ruan, Y. Noller, S. Tizpaz-Niari, S. Chattopadhyay, and A. Roychoudhury, "Timing side-channel mitigation via automated program repair," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–27, 2024.

[20] V. Monjezi, A. Trivedi, V. Kreinovich, and S. Tizpaz-Niari, "Fairness testing through extreme value theory," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2025, pp. 607–607.

[21] W. M. McKeeman, "Differential testing for software," *Digital Technical Journal*, vol. 10, no. 1, pp. 100–107, 1998.

[22] T. Petsios, A. Tang, S. Stolfo, A. D. Keromytis, and S. Jana, "Nezha: Efficient domain-independent differential testing," in *2017 IEEE Symposium on security and privacy (SP)*. IEEE, 2017, pp. 615–632.

[23] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[24] E. J. Gumbel, *Statistics of extremes*. Columbia university press, 1958.

[25] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An introduction to statistical modeling of extreme values*. Springer, 2001, vol. 208.

[26] M. R. Leadbetter, G. Lindgren, and H. Rootzén, *Extremes and related properties of random sequences and processes*. Springer Science & Business Media, 2012.

[27] G. Barthe, P. R. D'Argenio, and T. Rezk, "Secure information flow by self-composition," *Mathematical Structures in Computer Science*, vol. 21, no. 6, p. 1207–1252, 2011.

[28] Q. Lyu, D. Zhang, R. Da, and H. Zhang, "Refuzz: A remedy for saturation in coverage-guided fuzzing," *Electronics*, vol. 10, no. 16, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/16/1921

[29] M. Woo, S. K. Cha, S. Gottlieb, and D. Brumley, "Scheduling black-box mutational fuzzing," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 511–522. [Online]. Available: https://doi.org/10.1145/2508859.2516736

[30] R. Motwani, *Randomized Algor ithms*. Cambridge University Press, 1995.

[31] M. Böhme, "Stads: Software testing as species discovery," 2018. [Online]. Available: https://arxiv.org/abs/1803.02130

[32] M. Böhme, D. Liyanage, and V. Wüstholz, "Estimating residual risk in greybox fuzzing," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 230–241. [Online]. Available: https://doi.org/10.1145/3468264.3468570

[33] D. Liyanage, M. Böhme, C. Tantithamthavorn, and S. Lipp, "Reachable coverage: Estimating saturation in fuzzing," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 371–383.

[34] S. Lipp, D. Elsner, S. Kacianka, A. Pretschner, M. Böhme, and S. Banescu, "Green fuzzing: A saturation-based stopping criterion using vulnerability prediction," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 127–139. [Online]. Available: https://doi.org/10.1145/3597926.3598043

[35] S. Saha, M. Downing, T. Brennan, and T. Bultan, "Preach: a heuristic for probabilistic reachability to identify hard to reach statements," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1706–1717. [Online]. Available: https://doi.org/10.1145/3510003.3510227

[36] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A new way about using statistical analysis of worst-case execution times," *ACM SIGBED Review*, vol. 8, no. 3, pp. 11–14, 2011.

[37] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *2012 24th euromicro conference on real-time systems*. IEEE, 2012, pp. 91–101.

[38] J. Hansen, S. Hissam, and G. A. Moreno, "Statistical-based wcet estimation and validation," in *9th international workshop on worst-case execution time analysis (WCET'09)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[39] S. Tizpaz-Niari and S. Sankaranarayanan, "Worst-case convergence time of ml algorithms via extreme value theory," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, ser. CAIN '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 211–221. [Online]. Available: https://doi.org/10.1145/3644815.3644989

[40] A. Singhee and R. A. Rutenbar, "Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its

application," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.
[41] C. Antoniadis, D. Garyfallou, N. Evmorfopoulos, and G. Stamoulis, "Evt-based worst case delay estimation under process variation," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1333–1338.
[42] "Comparison in validate class of wss4j core is vulnerable to timing side channels," 2020. [Online]. Available: https://issues.apache.org/jira/browse/WSS-677
[43] R. Motwani and P. Raghavan, "Randomized algorithms," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 33–37, 1996.
[44] K. A. Doksum and B. S. Yandell, "26 tests for exponentiality," *Handbook of statistics*, vol. 4, pp. 579–611, 1984.
[45] J. D. Castillo, J. Daoudi, and R. Lockhart, "Methods to distinguish between polynomial and exponential tails," *Scandinavian Journal of Statistics*, vol. 41, no. 2, pp. 382–393, 2014.
[46] R. Sokal and F. Rohlf, "Biometry: The principles and practice of statistics in biological research 3rd edition wh freeman and co," *New York*, 1995.
[47] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *CMSB*. Springer, 2009, pp. 218–234.
[48] S. Sankaranarayanan, A. Chakarov, and S. Gulwani, "Static analysis for probabilistic programs: inferring whole program properties from finitely many paths," in *PLDI*, 2013, pp. 447–458.
[49] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, pp. 493–507, 1952.
[50] P. L. Chebyshev, "Des valeurs moyennes," *J. Math. Pures Appl*, vol. 12, no. 2, pp. 177–184, 1867.
[51] T. Antonopoulos, P. Gazzillo, M. Hicks, E. Koskinen, T. Terauchi, and S. Wei, "Decomposition Instead of Self-Composition for Proving the Absence of Timing Channels," *SIGPLAN Not.*, vol. 52, no. 6, pp. 362–375, jun 2017.
[52] J. Chen, Y. Feng, and I. Dillig, "Precise Detection of Side-Channel Vulnerabilities Using Quantitative Cartesian Hoare Logic," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 875–890.
[53] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.
[54] M. Hrba, M. Maciak, B. Peštová, and M. Pešta, "Bootstrapping not independent and not identically distributed data," *Mathematics*, vol. 10, no. 24, p. 4671, 2022.
[55] S. N. Lahiri, *Resampling methods for dependent data*. Springer Science & Business Media, 2013.
[56] D. N. Politis and J. P. Romano, "The stationary bootstrap," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1303–1313, 1994. [Online]. Available: http://www.jstor.org/stable/2290993
[57] S. Gilda, B. Heidrich, and F. Kiraly, "tsbootstrap: Enhancing time series analysis with advanced bootstrapping techniques," 2024. [Online]. Available: https://arxiv.org/abs/2404.15227
[58] A. Arcuri and L. Briand, "A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.
[59] G. Klees, A. Ruef, B. Cooper, S. Wei, and M. Hicks, "Evaluating Fuzz Testing," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 2123–2138.