Principled Sampling for Anomaly Detection

Brendan Juba
Washington University in St. Louis
bjuba@seas.wustl.edu

Christopher Musco, Fan Long,
Stelios Sidiroglou-Douskos and Martin Rinard
Massachusetts Institute of Technology
{fanl,cpmusco,stelios,rinard}@csail.mit.edu

***Abstract*—Anomaly detection plays an important role in protecting computer systems from unforeseen attack by automatically recognizing and filter atypical inputs. However, it can be difficult to balance the sensitivity of a detector – an aggressive system can filter too many benign inputs while a conservative system can fail to catch anomalies. Accordingly, it is important to rigorously test anomaly detectors to evaluate potential error rates before deployment. However, principled systems for doing so have not been studied – testing is typically ad hoc, making it difficult to reproduce results or formally compare detectors.**

To address this issue we present a technique and implemented system, Fortuna, for obtaining probabilistic bounds on false positive rates for anomaly detectors that process Internet data. Using a probability distribution based on PageRank and an efficient algorithm to draw samples from the distribution, Fortuna computes an estimated false positive rate and a probabilistic bound on the estimate's accuracy. By drawing test samples from a well defined distribution that correlates well with data seen in practice, Fortuna improves on ad hoc methods for estimating false positive rate, giving bounds that are reproducible, comparable across different anomaly detectors, and theoretically sound.

Experimental evaluations of three anomaly detectors (SIFT, SOAP, and JSAND) show that Fortuna is efficient enough to use in practice — it can sample enough inputs to obtain tight false positive rate bounds in less than 10 hours for all three detectors. These results indicate that Fortuna can, in practice, help place anomaly detection on a stronger theoretical foundation and help practitioners better understand the behavior and consequences of the anomaly detectors that they deploy.

As part of our work, we obtain a theoretical result that may be of independent interest: We give a simple analysis of the convergence rate of the random surfer process defining PageRank that guarantees the same rate as the standard, second-eigenvalue analysis, but does not rely on any assumptions about the link structure of the web.

I. Introduction

Anomaly detection systems are critical components of many security systems. By recognizing, then discarding, sanitizing, or otherwise nullifying outlier inputs that might otherwise exploit security vulnerabilities, anomaly detectors often play a central role in many computer security systems.

In general, however, anomaly detectors are not perfect. Specifically, anomaly detectors typically navigate a trade off between two kinds of errors:

False Positives - Type I error A Type I error occurs when an anomaly detector (incorrectly) rejects a benign input. A high false positive rate can significantly impair the utility of an anomaly detector — each false positive denies some part of the functionality of the system to the user.

False Negatives - Type II error A Type II error occurs when an anomaly detector (incorrectly) accepts a malicious input, leaving the system open to attack.

In general, making an anomaly detector more sensitive increases the false positive rate and decreases the false negative rate (and vice-versa). Appropriately balancing these two rates is therefore essential in obtaining an effective anomaly detector. Practitioners typically tune their anomaly detectors by running the detector on a (representative) set of inputs to develop an intuitive understanding of how the anomaly detector will operate in practice. Current techniques are ad-hoc, not guided by any theoretically well-founded framework or analysis, and therefore provide no guarantees on the effectiveness of the anomaly detector when deployed in production and no guidance on how to effectively test the anomaly detector to determine if it will work well in practice.

We present a technique and implemented system, Fortuna, that, for the first time, provides bounds on the number of false positives that a given anomaly detector will incur in practice. Fortuna focuses specifically on anomaly detection systems that handle data from browsing the Internet. This class of systems is very well studied and includes, for example, anomaly detectors that filter potentially anomalous video, images, and JavaScript files [18], [20], [28], [36], [38] or seek to filter entire malicious webpages [44], [47], [48]. Nevertheless, our approach is broad and we provide a general framework for building analysis systems for other classes of anomaly detectors.

A. False Positive Bounds

Specifically, Fortuna's sampling algorithm collects a sequence of randomly chosen *benign* inputs that enables Fortuna to derive probabilistic bounds of the following form:

- **One-Sided Bound:** For anomaly detectors with no false positives in the sequence of test inputs that the sampling algorithm generates, Fortuna provides a bound of the form $Pr(err^{(1)} < \epsilon) > 1 - \delta$, where $err^{(1)}$ is the actual Type I error (false positive rate) that the anomaly detector will

B. Juba was affiliated with Harvard University when this work was performed.

NDSS '15, 8-11 February 2015, San Diego, CA, USA
 ISBN 1-891562-38-X
http://dx.doi.org/10.14722/ndss.2015.23268

incur in practice, ϵ is an upper bound on this false positive rate, and $1-\delta$ is the certainty with which Fortuna is able to provide this bound.
For example, with a sequence of 46,052 sampled inputs and no false positives, Fortuna can show that the probability that the false positive rate is less than 0.01% is at least 99%.

- **Two-Sided Bound:** For anomaly detectors with some false positives in the sequence of test inputs that the sampling algorithm generates, Fortuna provides a bound of the form $Pr(|err^{(1)} - \widetilde{err}^{(1)}| < \epsilon) > 1-\delta$, where $err^{(1)}$ is the actual Type I error (false positive rate) that the anomaly detector will incur in practice, $\widetilde{err}^{(1)}$ is the (empirical) Type I error for the sequence of sampled inputs, ϵ is a bound on the difference between $err^{(1)}$ and $\widetilde{err}^{(1)}$, and $1-\delta$ is the certainty with which Fortuna is able to provide this bound.
For example, with a sequence of 26,492 sampled inputs, Fortuna produces an estimate of the false positive rate that is within 1% of the true false positive rate with 99% probability.

These bounds are based on standard statistical inequalities and are a function of the number of inputs that Fortuna's sampling algorithm is directed to obtain, with the bounds becoming tighter as the number of inputs increases. Specifically, for the one-sided bound, the number of required samples is proportional to $(1/\epsilon) \cdot \log(1/\delta)$; for the two-sided bound, the number of required samples is proportional to $(1/\epsilon^2) \cdot \log(1/\delta)$. Fortuna therefore enables practioners to determine, ahead of time, how many test inputs are required to obtain a desired bound. Our experimental results show that, in practice, Fortuna is easily able to obtain enough samples to provide tight bounds (see Section VI).

B. Accurate Sampling

To provide accurate false positive bounds, Fortuna's sampling algorithm delivers inputs from a probability distribution for benign inputs that accurately models the distribution the anomaly detector will encounter in practice.[1] We note that the standard approach of collecting a large set of inputs from an available source such as the Internet, then computing the false positive rate over this input set, does not provide an accurate estimate of the false positive rate — in general, the anomaly detector is more likely to encounter some inputs than others, with significant dependence on the ad hoc collection method selected. To obtain accurate bounds, Fortuna must select inputs from a concrete probability distribution over benign inputs that has two properties:

- **Reflects Production Use:** The probability distribution reflects the distribution of inputs that the anomaly detector will encounter in production use.
- **Efficient Sampling Algorithm:** The probability distribution supports an efficient sampling algorithm that Fortuna can use to generate its random sequence of inputs.

[1]The false positive rate does not depend on the distribution of malicious inputs — each malicious input is either a true negative or a false negative. Each benign input, in contrast, is either a true positive (when the anomaly detector accepts the input) or a false positive (when the anomaly detector rejects the input).

Fortuna is specifically designed to work with systems that process inputs that a typical user would see when browsing the Internet. Thus, it uses a probability distribution based on PageRank (first defined by Page et al. [45] for *ranking* web pages). The original motivation for PageRank was to weight web pages according to user desirability in order to provide relevant search results to a user (cf. [32, p.4]). At the same time, PageRank was also designed to be computationally tractable. A complete justification for our selection of PageRank can be found in Section II.

We are not interested in computing PageRank scores (probabilities) for all pages, but rather in *sampling* according to the PageRank distribution. It turns out that generating samples is even easier than computing the scores, which enables Fortuna to generate large enough sequences to provide tight false positive bounds. Specifically, a characterization of PageRank distributions that first appeared in the work of Andersen et al. [11] can be interpreted as a simple sampling algorithm. We develop a system based on this algorithm for rapidly sampling image files from a PageRank distribution. The system only requires access to freely available data, the web, and simple scripts.

Note that there is a crucial difference between the probability distributions of malicious and benign inputs. Because malicious inputs are specifically designed to target (in many cases not publicly known) program vulnerabilities, and because malicious inputs evolve rapidly in response to countermeasures (such as deployed anomaly detection systems), it is not typically possible to obtain reasonable probability distributions for malicious inputs. For this reason, Fortuna does not attempt to provide bounds about the false negative rate that the anomaly detector will encounter in production use. Our focus is in line with previous literature on anomaly detection – in light of limited access to malicious inputs, systems are typically designed to handle a fixed set of malicious examples and, during evaluation, false positive rate is used as the main quality metric.

C. Experimental Results

We evaluate Fortuna on three anomaly detectors: the SIFT [38], SOAP [37], and JSAND [18] anomaly detectors:

- **SIFT:** SIFT uses a conservative static program analysis to obtain a set of constraints on the values of fields in JPG and PNG image files processed by png2swf [8], jpg2swf [8], and Dillo [5].
If an input satisfies the constraints, SIFT guarantees that the input will not trigger target integer overflow errors [38]. Because the SIFT analysis is conservative, it may reject inputs that would not trigger the error. We consider an input file to be a false positive if it 1) violates the constraints but 2) does not trigger the integer overflow error. The results show that the SIFT anomaly detector can effectively guarantee the absence of integer overflow errors at critical memory allocation and block copy sites in our benchmark applications, including at least six potentially exploitable errors.
- **SOAP:** SOAP learns characteristics of fields in JPG and PNG image files that are processed by applications such as ImageMagick [7] and Dillo [5]. Specifically,

SOAP processes input files that the applications handle successfully to infer upper bound constraints of integer fields, sign constraints of integer fields, and upper bound constraints of data field lengths [37].
We consider an input file to be a false positive if it 1) violates the learned constraints but 2) the application can process the input file successfully. SOAP also has the capability to *rectify* such files (i.e., modify the file so that it satisfies the learned constraints [37], then pass the rectified file along to the application). The results show that the SOAP anomaly detector/input rectifier effectively nullifies six potentially exploitable errors in the sample applications [37].
We trained the SOAP anomaly detector on several thousand input files (5130 PNG and 3386 JPG) drawn from Fortuna's PageRank probability distribution. These training files are disjoint from the set of files used to evaluate the anomaly detector and compute the false positive rate bounds.

- **JSAND:** JSAND is a popular, publicly available anomaly detector for JavaScript. JSAND uses a variety of machine learning techniques to detect "Drive-by download" attacks, a common form of JavaScript malware that attempts to automatically download malicious software onto a users computer.
  JSAND provides a widely used Internet interface that allows a user to submit a JavaScript program to the JSAND anomaly detector [9]. The JSAND anomaly detector classifies the submitted program as either normal, suspicious, or malicious. We analyze two cases: the first considers suspicious files as false positives (aggressive filtering), the other considers them benign (conservative filtering).

We report results for the three anomaly detectors on sample inputs drawn from the Internet according to the PageRank probability distribution (see Section V). Over the course of less than 10 hours, Fortuna was able to sample over 40,000 JPG input files, over 60,000 PNG input files, and over 8,000 JavaScript programs from the Internet according to the PageRank distribution. On the resulting JPG and PNG input files, SIFT encountered no false positives. SOAP encountered fewer than 1,000 false positives on these same files. JSAND encountered 12 suspicious and 0 malicious JavaScript programs. The resulting false positive rate bounds are quite tight. Specifically, on all programs evaluated, Fortuna guarantees that SIFT has a false positive rate below 0.011%, with 99% confidence. The false positive rate for SOAP is $1.99\% \pm 0.83\%$ for JPEG files and $0.29\% \pm 0.67\%$ for PNG files, both two-sided bounds with confidence 99%. For JSAND, the false positive rate is less than 0.52% with confidence 99% if conservative filtering is performed (i.e. suspicious inputs are not rejected). If aggressive filtering is used (i.e. suspicious inputs are considered malicious and rejected), it is necessary to use the somewhat weaker two-sided bound, which guarantees the false positive rate is $0.14\% \pm 1.73\%$ with confidence 99%. These results show that it is very feasible to use Fortuna in practice to obtain tight bounds on false positive rates for anomaly detectors.

D. Contributions

This paper makes the following contributions:

- **Technique:** We present a technique for obtaining tight probabilistic bounds on the false positive rates that anomaly detectors will encounter when deployed in production use. This technique is, to the best of our knowledge, the first technique to provide any bound whatsoever on anomaly detector false positive rates.
- **Distribution and Sampling Algorithm:** One of the keys to obtaining tight bounds is obtaining a probability distribution that reflects the distribution of inputs that the anomaly detector will encounter in practice. At the same time, this probability distribution must also support a sampling algorithm that is efficient enough to use in practice. This paper presents a combination of probability distribution and sampling algorithm that satisfies both of these constraints. The distribution models inputs encountered by a typical user browsing the Internet (currently one of the most common input sources for anomaly detectors).
- **Analysis:** We present a formal analysis of the sampling algorithm and resulting false positive guarantees. This analysis includes a review of well known statistical tail inequalities, which characterize how many sample inputs are required to obtain a desired false positive bound (whether one-sided or two-sided).
- **Implementation:** We present Fortuna, a system that implements the PageRank sampling algorithm using freely available data and computes resulting probabilistic false positive rate bounds.
- **Experimental Results:** We present false positive rate bounds for three different anomaly detectors: the SOAP and SIFT anomaly detectors for JPEG and PNG input files and the JSAND anomaly detector for JavaScript programs. These results indicate that Fortuna's sampling algorithm is efficient enough to obtain enough inputs for providing tight bounds for practical anomaly detectors.

Anomaly detectors are a critical component of modern computer security systems. However, despite the central role they often play in such systems, there has been little to no formal analysis that enables practitioners to better understand the accuracy and effectiveness of their anomaly detectors. By providing guaranteed probabilistic bounds on the false positive rates that anomaly detectors will incur in practice, Fortuna can help practitioners better understand the consequences of deploying these powerful security tools.

II. The Case for PageRank

Fortuna computes bounds on Type 1 error by testing an anomaly detector on inputs drawn from a chosen distribution. The resulting bounds are thus meaningful *with respect to data drawn from that distribution*. So to obtain bounds that are useful in practice, it is important to choose a distribution that is as consistent as possible with inputs an anomaly detector will analyze in practice. At the same time, we need to rapidly obtain many samples from the distribution – obtaining tight bounds will require tens of thousands of sample inputs.

The tradeoff here is clear: a more sophisticated model or involved data collection process may provide more accuracy, but at the cost of increased sampling difficultly. For our intended application – anomaly detectors that process web data – we argue that PageRank balances this tradeoff and

thus provides an ideal distribution for testing. Furthermore, its relative simplicity and ease of use suggests that PageRank could become a benchmark for web-data anomaly detectors, replacing ad hoc analysis methods and increasing consistency and reproducibility across results.

Nevertheless, we stress that PageRank is just one possible viable distributions for testing anomaly detectors. When deploying an anomaly detector for use beyond web data, choosing an alternative distribution would be essential. Additionally, even within our chosen application domain, depending on intended application and available computational and data resources, alternative distributions could be more appropriate for testing. Our goal is to provide a framework for computing bounds given *any* chosen distribution and we offer PageRank as a simple, easily implemented, yet powerful example.

A. A Brief Introduction to PageRank

A more complete treatment of the PageRank distribution, including mathematical definitions and a theoretical discussion of our sampling algorithm, is included in Section IV. We present an abbreviated introduction here.

The PageRank distribution was first presented by Page et al. [45] and is the backbone of Google's search result ranking algorithm. Seeking to weight pages by importance, PageRank has been extensively studied in theory and practice thanks to its relative simplicity, accuracy, and (it turns out) utility for other purposes such as local graph partitioning and robust eigenvector approximations, for example [13], [32], [11], [39]. Prior to our work, PageRank distributions had not, however, previously been used as a test distribution for evaluating ADS systems.

To understand the PageRank distribution, consider the following "random surfer" process for randomly accessing pages on the World Wide Web:

(a) Begin by picking a starting webpage uniformly at random from all possible pages.
(b) If the current page has no outgoing links, jump to a page selected uniformly at random from all possible pages. Otherwise, with probability α choose a link from the page you are currently on uniformly at random, and follow that link; otherwise (with probability $(1-\alpha)$), jump to another page selected uniformly at random from all possible pages. Typically α is set to .85.

This Markov process loosely captures the behavior of a typical Internet user. The PageRank of a URL is then taken to be the fraction of visits to that page during the random surfer process as the number of steps of the process goes to infinity, i.e., the long-run fraction of time spent on that URL. Intuitively, webpages with more incoming links tend to have higher PageRank because the random surfer has more possible ways of getting to the page. It turns out that moreover, PageRank also captures a notion of the quality of these links. Specifically, links from high PageRank sources are more important because they are more likely to be taken than links from a page that is not visited often. This "quality by association" is the important insight behind PageRank, which has empirically outperformed other notions of importance on the web.

B. Merits of PageRank

We claim that computing Type I errors with respect to a PageRank distribution is reasonable for two reasons. First, we argue that PageRank has been successful, in practice, at capturing the relative importance of webpages (cf. Langville and Meyer [32, pp.4,25]). It is therefore reasonable to use PageRank to weight the *importance* of Type I (false positive) errors on various webpages: false positives on more important pages are more serious, as they are more likely to be visited. Second, we point out that the random surfer process is a reasonable synthetic model for the behavior of an average user. The corresponding PageRank distribution, by definition, captures *precisely* the long-run distribution over pages visited by the random surfer, and therefore is a reasonable synthetic model for the long-run distribution over pages visited by an average user.

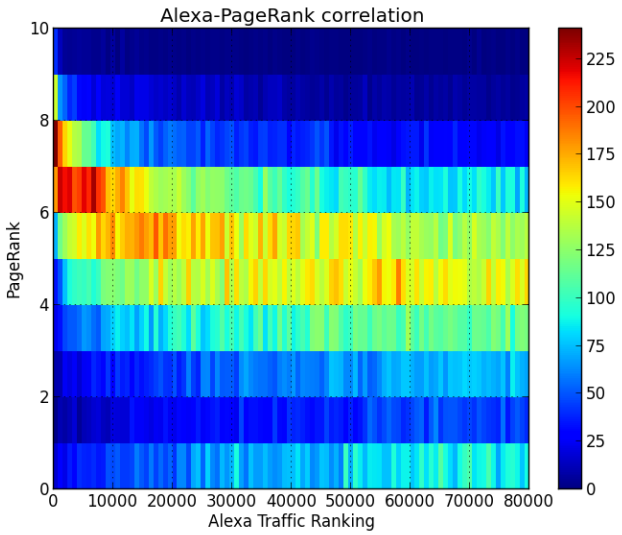

Fig. 1: PageRank Score (logarithmic scale) vs. Alexa Ranking up to $\sim 10^5$ pages, out of $\sim 10^9$–10^{10} total. Warmer (higher) colors indicate high density and thus reveal the trend curve.

In support of this second claim, we collected data on approximately $100,000$ of the Web's most visited webpages [2], comparing their PageRank score (provided by Google [6]) and their global traffic ranking (provided by Alexa Internet [1]). Figure 1 presents this data in a 2-dimensional histogram, which indicates that PageRank decreases in correlation with reduced traffic rank (a traffic rank of 1 is best). Noise in the plot can be attributed to several factors – for example, traffic numbers tend to be quite volatile (significant changes daily) while PageRank captures a more long term measure of importance. In addition, PageRank numbers as provided by Google are given on a logarithmic scale – a PageRank score of $x+1$ is twice is good as a PageRank score of x. This property limits the precision of our y-axis. Finally, Alexa traffic rankings should only be considered representative of true traffic numbers in a coarse grained sense [19, pp.38].

Nevertheless, we are interested in the overall trend – PageRank correlates positively with traffic ranking and thus

roughly captures visit frequencies while providing a long term measure of page importance.

Of course, this point would be moot if it were as expensive to sample from the PageRank distribution as it is to conduct, for example, an large-scale (statistically significant) user study that captures actual visit frequencies for various webpages. As we explain below, potentially more representative distributions are (prohibitively) expensive to sample from, especially in comparison to the efficient method presented in Section IV-B for PageRank sampling.

C. Alternatives to PageRank

There are several alternatives to our choice of the PageRank distribution for web data. PageRank is a computed based on link structure between webpages so it is *not* generally identical to the relative rates of traffic over various webpages, and as we review next, other techniques produce data that may more closely match actual traffic numbers. While we argue that sampling from other distributions is generally prohibitively expensive, the bounds we present in Section III could in theory be applied to any of these distributions. In particular:

1) Large-scale User Studies: An ideal choice for obtaining test data for anomaly detectors for the web would be to obtain data directly from real users. Data from traffic ranking sites, such as Alexa, unfortunately is not fine grained enough for our purposes since domains are tracked instead of individual URLs. Furthermore, such data is typically only available for the Internet's most popular sites (e.g. top 1 million domains out of > 950 million active domains in the case of Alexa [10]).

In order to obtain a representative sample of data, a user traffic study would have to cover a large population over a long period of time. Such a study was actually conducted by Meiss et al. [41] over about seven months during 2006–07 in their work on the quality of the random surfer as a user model. Their methodology involved collecting *all* of the traffic across the gateway for Indiana University's Bloomington campus; needless to say, it would have been difficult to obtain individualized informed-consent for the monitored population. Nevertheless, they note that the users' IP addresses have been deleted and the study received approval from the university's Institutional Review Board.

As effective as their technique is for obtaining a large, relatively high-quality sample of data, two subsequent developments cast doubt on whether or not such studies would still receive IRB approval in the absence of user consent. The first is that work on deanonymization, as developed by Narayanan and Shmatikov [12], [42], [43], demonstrates that merely deleting trivial unique identifiers (such as IP addresses) is not an adequate measure for protecting user privacy. The second is that shortly after the publication of the Meiss et al. study, whistleblowers at Indiana University alleged numerous cases of noncompliance by the Bloomington Office of Human Research Protections [34]. In particular, it was alleged that protocols were misreviewed, personal data had been released without subjects' consent, and subjects' protections were "at the bottom of the list of [the directors'] concerns." While the Meiss et al. study is never mentioned in any of the released portions of the allegations, the affair clearly does raise the question of whether a properly functioning IRB would have approved the study's protocol. As such, it highlights the forbidding challenges inherent in such a large user study.

Furthermore, even with informed consent, several other issues complicate user study design. For example, it is likely that users' browsing habits will change substantially if they explicitly consent to monitoring (a human Heisenberg effect). It is also difficult to access an unbiased population of users willing to participate in a traffic study. Such concerns have arisen in criticism of Alexa and other traffic ranking services who seek to provide much coarser data [19]. Although user studies still seem likely to provide higher quality data than modeling techniques like PageRank, even they are only approximations to the "true" traffic distribution we seek to address.

We briefly note that these concerns do not arise in situations where users have weaker than usual expectations of privacy and necessarily consent to outside monitoring of their surfing, such as in a government or corporate setting. In such settings, it would be possible to simply monitor the browsing behavior of the entire population to collect the data for training and evaluation of an anomaly detector. Again, the bounds we use still apply in this case, with the actual user data substituted for the data we sample from PageRank.

Otherwise, the disadvantages of direct data collection via user studies suggest that it might be desirable to use a representative model to generate synthetic traffic data. Although PageRank is one of the most popular, alternative models exist, and we discuss one potentially more accurate option next.

2) The ABC Model: In a follow-up to the work on comparing the random surfer model to real user data, Meiss et al. [40] proposed a more sophisticated user model, the "ABC model." This model generates traces in a similar way to PageRank's random surfer model, but it simulates features such as back buttons, user bookmarks, and decaying "user interest." Meiss et al. [40] demonstrate that it produces trace data that, in several ways, agree with real user traces better than PageRank. This includes prediction of aggregate page traffic: they find that while the distribution of webpage traffic produced by the ABC model, PageRank, and the empirical data all roughly follow power-law distributions, the ABC distribution more closely matches with the empirical power law distribution.

Thus, if *data quality* is the most important consideration, the ABC model may be more suited to testing web-data based anomaly detectors than our PageRank model. The ABC model would be a good choice in situations where, in addition to the quality of data being the primary concern, large-scale user studies are simply infeasible. As mentioned, it could be used with the general framework we present for obtaining quantitative error bounds from a set of samples.

Unfortunately, improved accuracy is achieved at a significant cost per sample. Unlike PageRank, the ABC model is inherently non-Markovian and thus the usual convergence analyses do not apply. In fact, it is not clear from work on the model that the long-run distribution over pages *ever* converges. Even if the ABC model converges, it is not empirically known how long one should simulate the ABC model for before obtaining samples of page visits. Naturally, a "burn in" period would be required to ensure that samples are sufficiently independent from the chosen start page. PageRank is considered

to mix very quickly – nevertheless, if it is sampled by simply simulating the described random surfer model and taking a sample after initial burn in, over 120 steps (page downloads) would be necessary for selecting each unbiased test URL. In contrast, the optimization we obtain from the reformulation of Andersen et al. [11], which is unique to PageRank's definition, requires about 7 page downloads on average. It would be an interesting direction for future theoretical work to see if the long-run distribution of ABC model (or another similarly close approximation to the real traffic distribution) could also be sampled so efficiently.

III. Statistical Background

Before giving an in-depth theoretical treatment of PageRank and introducing our proposed sampling algorithm, we review standard probability techniques that Fortuna will use to compute its false positive rate bounds. In particular, we explain how to determine the required number of test inputs for obtaining a bound on the false positive rate (one-sided or two-sided) to within any desired accuracy.

All of the error bounds described have been known for quite some time in statistics and machine learning. Our contribution is applying them to anomaly detection for the first time and, more importantly, providing a distribution and accompanying sampling algorithm that is efficient enough to effectively apply the bounds presented.

A. Bounds on Sample Size

Let S be the space of all possible inputs to a program. We will model the benign inputs as having been drawn (independently) from a distribution $\mathcal{D}$, with support $X \subseteq S$; that is, X is the space of benign inputs one could possibly encounter in practice. Although in practice inputs are *not* independent, we are only interested in averages over a long-run distribution over inputs, and not in the order in which the inputs were encountered. Equivalently, we could consider defining $\mathcal{D}$ to be an average over time of the (correlated) inputs. We will see later that it is legitimate to treat the examples as being independently distributed from a fixed distribution for the actual distribution we sample from—that is, that we can sample directly from the "long-run" inputs of a synthetic input distribution.

Let $p : S \to [0,1]$ be the probability density function of $\mathcal{D}$, and let $T : S \to \{0,1\}$ be the indicator function for harmful inputs, i.e., labeling each $s \in S$ as either benign (denoted by 0) or harmful (denoted by 1). Then, since X is the subset of S (with nonzero density under $\mathcal{D}$) containing benign inputs, $T(x) \equiv 0$ for all $x \in X$. Now, suppose we have access to an anomaly detection function $F : S \to \{0,1\}$. F is an attempt to approximate T.

To capture differences between F and T, note that $|F(x) - T(x)| = 0$ if the functions agree for an input x and $|F(x) - T(x)| = 1$ if they disagree. While T is uniformly 0 on X, the same is not necessarily true for F. We define:

Definition 1 (Type I Error over $\mathcal{D}$):

$$err_{\mathcal{D}}^{(1)}(F) = \mathop{\mathbb{E}}_{\mathcal{D}}[F|X] = \sum_{x \in X} \frac{p(x)}{p(X)} \cdot F(x)$$

where $p(X)$ denotes $\sum_{x \in X} p(x)$.

Unless X is small, it is infeasible to compute $err_{\mathcal{D}}^{(1)}(F)$ exactly. However, given access to n independent draws from $\mathcal{D}$, $\{x_1, \ldots, x_n\}$, we can estimate $err_{\mathcal{D}}^{(1)}(F)$ by

Definition 2 (Empirical Type I Error over $\mathcal{D}$):

$$\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) = \sum_{i=1}^{n} \frac{1}{n} F(x_i)$$

$\frac{1}{n}$ is the empirical frequency of x_i from our sample, and thus replaces $p(x)$ in the expectation. Observe that, for a random variable X' denoting the conditional distribution of $\mathcal{D}$ given the input is benign (i.e., lies in X) $F(X')$ is just a Bernoulli random variable with parameter $err_{\mathcal{D}}^{(1)}(F)$ so, we can rewrite

$$\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) = \frac{1}{n} \sum_{i=1}^{n} F(x_i) = \frac{1}{n} \sum_{i=1}^{n} y_i$$

where each $y_i \sim \mathcal{B}(err_{\mathcal{D}}^{(1)}(F))$.

When our data is generated from $\mathcal{D}$, $\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) \to err_{\mathcal{D}}^{(1)}(F, n)$ as $n \to \infty$. Accordingly, we can obtain a better estimate of F's false positive rate by selecting a larger set of test inputs. The exact size of the sample required is given in the following:

Theorem 3 (Hoeffding's bound [27]): For any $\epsilon, \delta \in (0,1]$, if $n \geq \frac{\ln(2/\delta)}{2\epsilon^2}$ and $x_1, \ldots, x_n$ are drawn independently from $\mathcal{D}$, then with probability $1 - \delta$ over $x_1, \ldots, x_n$,

$$\|\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) - err_{\mathcal{D}}^{(1)}(F)\| \leq \epsilon$$

If we set $\delta = .05$ and $\epsilon = .01$, Theorem 3 tells us how to set n so that we can make the following statement with 95% confidence: "F's true false positive rate on data generated from $\mathcal{D}$ is within an additive 1% of our estimated false positive rate".

This bound is "two-sided": it guarantees that, with high probability, $\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) > err_{\mathcal{D}}^{(1)}(F) - \delta$ and that $\widetilde{err}_{\mathcal{D}}^{(1)}(F, n) < err_{\mathcal{D}}^{(1)}(F) + \delta$. Unfortunately, obtaining tight two-sided bounds is expensive. The number of examples required by Theorem 3 scales as $\Omega(1/\epsilon^2)$, which grows rapidly. Since ADS systems today aim to provide very low false positive rates of $.01\%$ or less, it is often infeasible to collect a sample of the required size in practice. For example, for 99% confidence and an error bound of $.01\%$, we would need more than 264 million examples. Theorem 3 is still very helpful for estimating the quality of systems with higher false positive rates. For 99% confidence and an error bound of 1%, a sample of size $n = 39,120$ suffices. We will apply it to such a system ([37]) in Section VI.

To certify the quality of systems with very low false-positive rates, we switch to a "one-sided" bound, following a standard calculation from learning theory (e.g., appearing in Theorem 2.2 of Kearns and Vazirani [29, p.36]):

Theorem 4: For any $\epsilon, \delta \in (0,1]$, if we draw $n \geq (1/\epsilon)\ln(1/\delta)$ examples from $\mathcal{D}$ and

$$err_{\mathcal{D}}^{(1)}(F) > \epsilon$$

then the probability that we encounter *no* false positives is at most δ.

Proof: Suppose $err^{(1)}_{\mathcal{D}}(F) > \epsilon$. Then for each example x_i, the probability that x_i is *not* a false positive is at most $(1-\epsilon)$. Since the examples are independently drawn from $\mathcal{D}$, the probability that none of them are false positives is at most $(1-\epsilon)^n$. Now, using the fact that $(1-1/x)^x \leq e^{-x}$ for $x \geq 1$, we find that for our choice of n, the probability that we do not encounter a false positive is at most

$$(1-\epsilon)^n = (1-\epsilon)^{(1/\epsilon)\ln(1/\delta)} \leq e^{-\ln(1/\delta)} = \delta$$

■

Note that, while our dependence on δ has remained the same, Theorem 4 requires significantly fewer examples for a given ϵ since it depends on $1/\epsilon$ instead of $1/\epsilon^2$. For example, we need just $46,052$ samples to claim that $err^{(1)}_{\mathcal{D}}(F) < .01\%$ with with 99% confidence. The difference is quite stark for small ϵ, as shown in Figure 2.

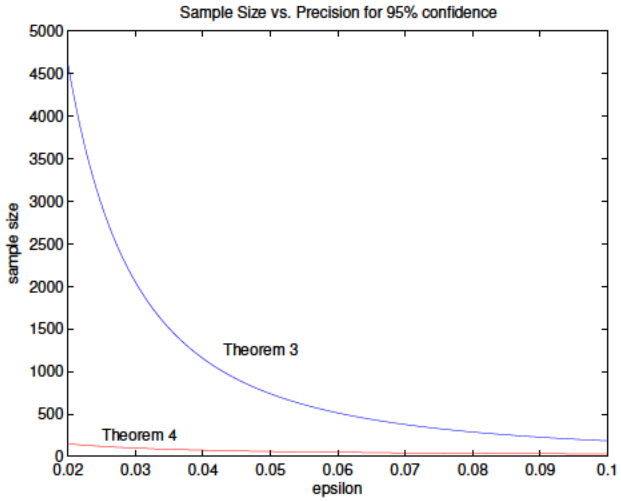

Fig. 2: Required sample size from Theorems 3 and 4. Theorem 3 requires substantially more examples for small ϵ.

If we attempt to use Theorem 4 in the straightforward way without prior knowledge of ϵ, after drawing a large sample we may encounter a false positive and then Theorem 4 makes no guarantee. Instead, we will use the following algorithm: fix a desired ϵ and δ, and sample from $\mathcal{D}$ until we either encounter a false positive or until $n = \frac{1}{\epsilon}\ln\frac{1}{\delta}$. In the latter case we return $\epsilon' = \epsilon$, and otherwise (if the $(n+1)$st example was our first false positive) return $\epsilon' = \frac{1}{n}\ln\frac{1}{\delta}$.

Theorem 5: With probability $1-\delta$,

$$err^{(1)}_{\mathcal{D}}(F) \leq \epsilon'$$

for the ϵ' output by the above algorithm.

Proof: Let $\epsilon^* = err^{(1)}_{\mathcal{D}}(F)$, and let $n^* = \lceil \frac{1}{\epsilon^*}\ln\frac{1}{\delta}\rceil$. Notice, as long as we encounter a false positive within n^* examples, we will output a satisfactory ϵ'. For each n, let B_n be the event that there is a false positive among the first n samples. The event that the algorithm produces a satisfactory ϵ' is therefore $\bigvee_{n=1}^{n^*} B_n$. Note that for $n' > n$, $B_n \subset B_{n'}$, and hence for all n', $\mathbb{P}\left(\bigvee_{n=1}^{n'} B_n\right) = \mathbb{P}(B_{n'})$. Therefore, the probability that we obtain a satisfactory estimate is $\mathbb{P}(B_{n^*}) \geq 1-\delta$ by Theorem 4. ■

B. Using Approximate Distributions

The theorems we have recalled that obtain an actual error bound from an empirical false positive rate are stated for the case where the data is drawn from the same distribution as used to define the error rate. We note that it is sufficient to draw the training data from a distribution that is not identical to the distribution we use to define the error rates, but merely a reasonable approximation to it. Specifically, we recall the *statistical distance* (or *total variation distance*):

Definition 6 (Statistical distance): For two probability distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over a common sample space X, the *statistical distance* between $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ is defined as

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) = \max_{A \subseteq X} |\mathop{\mathbb{P}}_{\mathcal{D}_1}[A] - \mathop{\mathbb{P}}_{\mathcal{D}_2}[A]|.$$

In particular, in our case X is the set of webpages; suppose we have fixed an ADS F, and let A be the set of webpages that are false positives for F. Then $err^{(1)}_{\mathcal{D}_1}(F) = \mathbb{P}_{\mathcal{D}_1}[A]$ and $err^{(1)}_{\mathcal{D}_2}(F) = \mathbb{P}_{\mathcal{D}_2}[A]$. Given training data from $\mathcal{D}_1$, the theorems allow us to bound $err^{(1)}_{\mathcal{D}_1}(F)$ directly. If we furthermore know that $\mathcal{D}_2$ is *close to* $\mathcal{D}_1$ in the sense that the statistical distance $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ is small, say less than ϵ, then we know that

$$\begin{aligned} |err^{(1)}_{\mathcal{D}_1}(F) - err^{(1)}_{\mathcal{D}_2}(F)| &= |\mathop{\mathbb{P}}_{\mathcal{D}_1}[A] - \mathop{\mathbb{P}}_{\mathcal{D}_2}[A]| \\ &\leq \max_{A' \subseteq X} |\mathop{\mathbb{P}}_{\mathcal{D}_1}[A'] - \mathop{\mathbb{P}}_{\mathcal{D}_2}[A']| \\ &= \Delta(\mathcal{D}_1, \mathcal{D}_2) \end{aligned}$$

so in particular, $err^{(1)}_{\mathcal{D}_2}(F) \leq err^{(1)}_{\mathcal{D}_1}(F) + \epsilon$. We may thereby obtain a bound on the error rate with respect to our desired, difficult to sample distribution $\mathcal{D}_2$ using examples from a sufficiently good approximation, $\mathcal{D}_1$.

Although the definition we have given clarifies the significance of the statistical distance for our application, there is a standard alternative definition of the statistical distance that demonstrates that it is also surprisingly easy to reason about:

Theorem 7: $\Delta(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2}\sum_{x \in X} |\mathbb{P}_{\mathcal{D}_1}[x] - \mathbb{P}_{\mathcal{D}_2}[x]|$.

So "statistical distance" really is essentially just the ℓ_1 distance between the density functions of $\mathcal{D}_1$ and $\mathcal{D}_2$, taken as vectors over $[0,1]$.

IV. Analysis and Sampling Algorithm

With our sampling bounds in place, we present a more formal treatment of the PageRank probability distribution. After discussing its theoretical foundation, we describe a sampling algorithm based on work of Andersen et al. [11] that allows us to efficiently obtain *exact* samples from the distribution. The algorithm relies on a basic web crawling procedure that, on average, accesses very few web pages for each sample. It is

efficient enough to enable Fortuna to obtain tight false positive bounds based on the theorems from Section III.

Again, while the error bounds for sampling are well known, the difficulty in making use of them is that we require test data that has been drawn from a distribution that is representative of the real distribution over benign inputs one would encounter. As obtaining such data may be expensive, many authors have had to settle for test data that was selected in some other manner. As noted by Chandola et al. [16], the use of such synthetic data may undermine the quality of the evaluation. As discussed in Section II, we avoid this issue in the case of anomaly detectors that process data from the web. PageRank provides a distribution over web data that is both easy to sample from and representative of the inputs.

A. Formal Properties of PageRank

Building on the overview given in Section II-A, we formalize the random surfer process described by using a "random walk" matrix, $\mathbf{A}$. (We never construct this matrix explicitly. It is only used to analyze the process). If n is the number of pages on the web, $\mathbf{A}$ is an $n \times n$ real-valued matrix. Every entry in $\mathbf{A}$ lies in $[0, 1]$ and specifically, $\mathbf{A}_{i,j}$ is the probability of moving from page j to page i. For pages that j links to, this probability should equal:

$$\frac{\alpha}{degree(j)} + \frac{1-\alpha}{n}$$

If j has links, but the page i is not directly linked to from j, $\mathbf{A}_{i,j}$ simply equals:

$$\frac{1-\alpha}{n}$$

and if $degree(j)$ is 0, $\mathbf{A}_{i,j}$ should simply equal $\frac{1}{n}$ for all i. If we sum up entries in any row of $\mathbf{A}$, the total should be 1. Each row gives a probability distribution on webpages, conditioned on our current position. That is, $\mathbf{A}$ is a *"row stochastic"* matrix.

Actually, the process described above describes *one of many* possible PageRank distributions. In practice, it is neither feasible nor (it turns out) desirable to use precisely this random surfer process. First, the web is constantly growing and may contain pages or subsets of pages not linked to by any other page. And second, it is believed that Google combats attempts at manipulating the PageRank by explicitly modifying the distribution (cf. the case of Google vs. SearchKing [32, p.54]). Thus, in practice one never performs a true random jump to a uniformly selected page from the web. Instead, one typically selects a fixed "teleportation" vector $\mathbf{v}$ that approximates the uniform distribution over all (legitimate) webpages. At each random jump of the surfer process, we move to page i with probability $\mathbf{v}_i$. In such a case, $\mathbf{v}$ contains uniform probabilities over a certain large proportion of webpages known to exist, with zeros everywhere else. We will use such a teleportation distribution in our sampling implementation (see Section V). As long as $\mathbf{v}$ is rigorously specified with a reasonably large support, the resulting distribution over web pages is still representative. Furthermore, we will use a distribution over "seed" webpages that is public and simple to access, allowing for consistency with papers that seek to employ a sampling strategy similar to ours.

It will be helpful to separate $\mathbf{v}$'s contribution to $\mathbf{A}$ from the effect of actual links. Thus, define $\mathbf{P}$ to be another $n \times n$ matrix with entries in $[0, 1]$. $\mathbf{P}_{i,j} = 0$ if j does not link to i and $1/degree(j)$ is j does link to i.

With $\mathbf{v}$ and $\mathbf{P}$ chosen, we can define our random surfer model as a random walk process. Suppose $\mathbf{e}$ is the all ones vector. Then we can write:

$$\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\mathbf{e}\mathbf{v}^\top \tag{1}$$

Let $\mathbf{s}_t$ be the probability distribution over webpages after step t of our walk. So, $\mathbf{s}_t(i)$ is the probability of being at page i after t steps. Since the walk starts with a standard teleport, $\mathbf{s}_0 = \mathbf{v}$.

Furthermore, if $\mathbf{A}$ holds the conditional probabilities specified, $\mathbf{s}_t = \mathbf{s}_{t-1}^\top \mathbf{A}$. Expanding this recurrence, $\mathbf{s}_t = \mathbf{s}_0^\top \mathbf{A}^t$. The PageRank distribution is the limit of this random walk process. Specifically,

Definition 8 (PageRank): The *PageRank vector* $\mathbf{s}$ for random walk matrix $\mathbf{P}$, teleportation distribution vector $\mathbf{v}$ and $\alpha \in [0, 1)$ is given by

$$\mathbf{s}^\top = \lim_{t\to\infty} \mathbf{s}_0^\top \mathbf{A}^t \tag{2}$$

for $\mathbf{s}_0 = \mathbf{v}$ and $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\mathbf{e}\mathbf{v}^\top$.

The actual existence of the limit $\mathbf{s}^\top$ is a standard property of PageRank and follows from noting that, restricted to the webpages reachable from the none zero elements of $\mathbf{v}$, the PageRank random walk forms an irreducible, aperiodic Markov chain (see for example Haveliwala and Kamvar [26] or Farahat et al. [22]).

One is generally not only interested in whether or not the Markov chain converges to the stationary distribution, but also how quickly, in case one is using the power method to compute the PageRank vector. This is especially important in case one is not computing the entire vector, but instead sampling from the vector by performing a random walk according to the random surfer distribution; in this case, a bound on the convergence rate is essential since one cannot test for numerical convergence (as is usually done when computing PageRank vectors in practice). Unfortunately, the state of the literature in this regard is quite poor. The original work of Page et al. [45] incorrectly references an analysis of an undirected Markov chain; subsequent work on the convergence rate of the power method that corrected this error, e.g. Haveliwala and Kamvar [26], still assumed that the matrix $\mathbf{A}$ is diagonalizable, which in general is not true. (This assumption is expressly noted by Langville and Meyer [32, p.164].) Specifically, there is no guarantee that the transition matrix of a directed Markov chain has a full basis of eigenvectors.

In spite of these analytical snags, the power method works rather well in practice, as well as predicted by these "second eigenvalue" analyses. For completeness, we now offer a direct proof that is similar to the standard arguments (e.g., [13], [14]) but does not rely on diagonalizability (or even the existence of a second eigenvector) and obtains the same convergence rate as a "second-eigenvalue" analysis of the power method. We therefore find that none of these assumptions are necessary to

explain the fast rate of convergence of the power method in practice.

Theorem 9: Suppose $\mathbf{s}^\top = \lim_{t\to\infty} \mathbf{s}_0^\top \mathbf{A}^t$ for $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\mathbf{e}\mathbf{v}^\top$ with $\alpha \in [0,1)$, a probability distribution $\mathbf{v}$, and a random walk matrix $\mathbf{P}$. Then ${\mathbf{s_0}}^\top \mathbf{A}^t$ is within $2\alpha^t$ of $\mathbf{s}^\top$ under the ℓ_1-norm.

Proof: Expanding our recurrence gives:

$$\begin{aligned}\mathbf{s}_t^\top &= \mathbf{s}_{t-1}^\top \mathbf{A} \\ &= \alpha \mathbf{s}_{t-1}^\top \mathbf{P} + (1-\alpha)\mathbf{s}_{t-1}^\top \mathbf{e}\mathbf{v}^\top \\ &= \alpha \mathbf{s}_{t-1}^\top \mathbf{P} + (1-\alpha)\mathbf{v}^\top \end{aligned} \tag{3}$$

The last step follows because $\mathbf{s}_{t-1}^\top \mathbf{e} = \|\mathbf{s}_{t-1}^\top\|_1 = 1$ since $\mathbf{s}_{t-1}^\top$ is a probability distribution. Similarly, since $\mathbf{s}^\top = \mathbf{s}^\top \mathbf{A}$, $\mathbf{s}^\top = \alpha \mathbf{s}^\top \mathbf{P} + (1-\alpha)\mathbf{v}^\top$. Thus

$$(\mathbf{s} - \mathbf{s}_t)^\top = \alpha(\mathbf{s} - \mathbf{s}_{t-1})^\top \mathbf{P} \tag{4}$$

Since t was arbitrary, by induction on t we conclude that

$$(\mathbf{s} - \mathbf{s}_t)^\top = \alpha^t (\mathbf{s} - \mathbf{s}_0)^\top \mathbf{P}^t \tag{5}$$

Now, since they are probability distributions, $\|\mathbf{s}\|_1 = \|\mathbf{s}_0\|_1 = 1$. By the triangle inequality, $\|\mathbf{s} - \mathbf{s}_0\|_1 \leq 2$. Furthermore, since $\mathbf{P}^t$ is a row stochastic transition matrix, $\|\mathbf{P}^t\|_1 = 1$. Combined with the triangle inequality this gives:

$$\begin{aligned}\|(\mathbf{s} - \mathbf{s}_0)^\top \mathbf{P}^t\|_1 &\leq \|\mathbf{s} - \mathbf{s}_0\|_1 \|\mathbf{P}^t\|_1 \\ &= \|\mathbf{s} - \mathbf{s}_0\|_1 \\ &\leq 2\end{aligned}$$

It follows that $\|\alpha^t(\mathbf{s} - \mathbf{s}_0)^\top \mathbf{P}^t\|_1 \leq 2\alpha^t$. Thus, from Equation 5, the ℓ_1 norm of our error from $\mathbf{s}$ is bounded by$\|\mathbf{s} - \mathbf{s}_t\|_1 \leq 2\alpha^t$. ∎

The basic paradigm for using PageRank in answering search queries (described by Page et al. [45]) is to first compute a PageRank vector for the entire web, second filter out the pages that do not contain the requested terms, and then output the remaining pages, ordered by their PageRank. We can view this filtering step as *conditioning* the PageRank distribution on the presence of the requested terms. This is how we will define distributions over specific types of files:

Definition 10 (PageRank for a file type): For a file type τ and PageRank vector $\mathbf{s}$, we define $\mathbf{s}|_\tau$ to be the vector equal to $\mathbf{s}(i)$ if i is of type τ and 0 otherwise. Then the *PageRank distribution over files of type* τ is $\mathbf{s}_\tau = \mathbf{s}|_\tau / \|\mathbf{s}|_\tau\|_1$.

B. Sampling from PageRank

Since it is defined as a limit, it seems reasonable to compute an estimate for $\mathbf{s}$ by simply taking t to be a large number and applying $\mathbf{A}$ to an arbitrary starting vector t times. This process is analogous to the power method for finding the top eigenvector of a diagonalizable matrix. It was originally suggested by Page et al. [45] and has been analyzed and applied throughout the literature. Since the error ($2\alpha^t$) shrinks exponentially with the number of steps, the power method is fast and is the standard means for computing PageRank.

However, it turns out that *sampling* from the PageRank distribution is even simpler. In fact, it is possible to sample from the distribution exactly, with no error. We first note that the PageRank distribution can be reformulated as a random walk of a random (geometrically-distributed) length, a fact essentially observed by Andersen et al. [11]. Recall that a *geometrically distributed random variable of parameter* p is a random variable that counts the number of times a p-biased coin must be tossed before a 'heads' is obtained. Precisely, it takes value i with probability $(1-p)p^i$, for every integer $i \geq 0$.

Given a parameter α, random walk matrix $\mathbf{P}$, and teleportation vector $\mathbf{v}$, consider the distribution over webpages sampled as follows:

1) Sample t from a geometric distribution of parameter α.
2) Sample i from $\mathbf{v}$, and take a t-step random walk started from i according to $\mathbf{P}$.

Theorem 11: The sampling algorithm produces a webpage from the PageRank distribution with parameter α, random walk matrix $\mathbf{P}$ and teleportation vector $\mathbf{v}$. Furthermore, it only requires visiting $1/(1-\alpha)$ pages in expectation.

Proof: Our analysis is based on an observation of Andersen et al. [11]. The distribution of web pages output by the sampling algorithm is given by the vector

$$\mathbf{s'}^\top = \sum_{t=0}^{\infty} (1-\alpha)\alpha^t \mathbf{v}^\top \mathbf{P}^t.$$

We show $\mathbf{s}'$ is the PageRank distribution with parameter α, random walk matrix $\mathbf{P}$ and teleportation vector $\mathbf{v}$. As we have already established existence and uniqueness of the PageRank vector, we only need to verify that the geometric random walk distribution $\mathbf{s}'$ is a fixed point of the map

$$\mathbf{A}(x) = x[\alpha\mathbf{P} + (1-\alpha)\mathbf{e}\mathbf{v}^\top]$$

that takes an additional random step according to the random surfer process. Thus, by linearity:

$$\begin{aligned}\mathbf{A}(\mathbf{s'}^\top) &= \left(\sum_{t=0}^{\infty} (1-\alpha)\alpha^t \mathbf{v}^\top \mathbf{P}^t\right)[\alpha\mathbf{P} + (1-\alpha)\mathbf{e}\mathbf{v}^\top] \\ &= \left(\sum_{t=0}^{\infty} (1-\alpha)\alpha^t \mathbf{v}^\top \mathbf{P}^t(\alpha\mathbf{P})\right) + (1-\alpha)\mathbf{v}^\top \\ &= \left(\sum_{t=1}^{\infty} (1-\alpha)\alpha^t \mathbf{v}^\top \mathbf{P}^t\right) + (1-\alpha)\alpha^0 \mathbf{v}^\top \mathbf{P}^0 \\ &= \mathbf{s'}^\top\end{aligned}$$

The expected number of steps of the random walk is given exactly by

$$\sum_{t=1}^{\infty} \mathbb{P}[\#steps \geq t] = \sum_{t=1}^{\infty} \alpha^t = \frac{\alpha}{1-\alpha}$$

Since the number of pages visited is the number of steps of the random walk plus one initial page from the teleportation distribution, this is $1/(1-\alpha)$ pages in total. ∎

Sampling from the PageRank distribution for a specific file type is easily accomplished as long as the file type is sufficiently common. We can sample from $\mathbf{s}_\tau$ for a file type τ by using the previous algorithm to sample from the PageRank

```
1  Parameters:
2    α: PageRank random jump probability
3    Fmt: File format to extract
4  Return:
5    Downloaded files
6
7  random_walk(α, Fmt) {
8      N ← 0
9      while (rand(0,1) < α):
10         N ← N + 1
11     U ← get_random_url()
12     cnt ← 0
13     while (cnt < N):
14         U ← get_random_outgoing_link(U)
15         if (U is empty) then:
16             U ← get_random_url()
17         cnt ← cnt + 1
18     return fetch_file_from_url(U, Fmt)
19 }
```

Fig. 3: Random walk algorithm

distribution $\mathbf{s}$ repeatedly until we draw a URL for a file of type τ.

Theorem 12: The above sampling algorithm produces files from the PageRank distribution over files of type τ.

Proof: Notice, the probability of this algorithm returning the ith URL is 0 if it is not a file of type τ and otherwise is proportional to $\mathbf{s}(i)$. Since the algorithm induces a distribution over files, the probability that the ith URL is output must be precisely $\mathbf{s}(i)/\|\mathbf{s}|_\tau\|_1$, which we see is the PageRank distribution over files of type τ. ■

C. Stability of PageRank

Our PageRank distribution is partially computed using a random walk on the actual web. The web is not static, however: its link structure is constantly changing over time. The utility of our PageRank distribution as a common benchmark for generating reproducible results would be limited if it were too sensitive to these changes in the web's structure. Fortunately, we note that it is quite insensitive to these changes. Langville and Meyer provide a nice review of the stability of PageRank [32, Chapter 6]—very roughly, the effect of a change to the links on a page is proportional to that page's PageRank, which is small for most pages. Indeed, PageRank can be viewed as implicitly being "regularized" [39] and hence very stable to changes in either the web's link structure or the choice of teleportation vector.

V. Design and Implementation

We next discuss our implementation of Fortuna to estimate Type I errors by sampling from the PageRank distribution using the the efficient algorithm of Theorem 11. Fortuna is implemented in approximately 700 lines of Python and it uses Common Crawl [3], a regularly updated snapshot of the public Internet, as the PageRank sampling source.

A. Sampling from PageRank

Common Crawl is a public repository, hosted on Amazon's Elastic Cloud, that builds and maintains a snapshot of the web. To avoid the cost of building an index from scratch (in the range of $10000 with current Amazon pricing [2]), Fortuna uses an existing copy of the Common Crawl URL Index provided by triv.io [4]. The URL Index is implemented as a prefixed B-tree that provides the ability to efficiently search the index by URL, URL prefix, subdomain or top-level domain. The currently available index is approximately 220 GB in size, representing approximately $2.3x10^9$ URLs.

Fortuna uses the Common Crawl URL index to approximate an uniform distribution of the entire web (i.e., teleportation distribution $\mathbf{v}$ in Section IV-B). As described in Section IV-B, Fortuna samples random pages from the URL index as the starting page for a random walk (i.e, sample random pages from teleportation distribution $\mathbf{v}$ in Section IV-B).

B. Random Walk Algorithm

Figure 3 presents Fortuna's random walk algorithm. The algorithm takes as input the PageRank parameter α, and the desired file format Fmt. It first computes the number of required walk steps, N, based on a geometric distribution with parameter α at lines 8-10 in Figure 3 (see Section IV-B). It simply flips a biased coin (with heads probability $(1-\alpha)$) until heads comes up.

The algorithm extracts a random URL from the URL Index (get_random_url() at line 11) to seed the random walk. Next, the algorithm downloads and parses the URL to extract all html links $links$. To simulate the random walk, a link U is randomly selected from $links$, and the walk count cnt is incremented. If the URL does not contain any link, the algorithm will teleport to another random URL from the URL Index (lines 15-16). The process is repeated using U as the seed URL and continues until the algorithm has performed enough steps (i.e., $cnt < N$). Once enough steps have been performed, the algorithm parses the last URL U and examines the HTML for links pointing to file of format Fmt (line 18). If U contains more than one file of format Fmt, one is randomly selected.

Note that this algorithm can be parallelized easily. To efficiently download files Fortuna uses a distributed crawler infrastructure where workers (on multiple machines) run the random walk algorithm and store results to a centralized database.

VI. Experimental Results

We evaluate Fortuna on three anomaly detectors: SOAP [37], SIFT [38], and JSAND [18] on three input formats: JPEG and PNG (SOAP and SIFT) and JavaScript (JSAND).

1) SOAP [37] is an input rectification system that learns input constraints over a set of benign inputs and then enforces the learned constraints over the incoming inputs. In our experiments for SOAP, we count a false positive as occurring if SOAP rectifies a (benign) collected input.
2) SIFT [38] statically analyzes an application and generates sound input constraints, so that any input that satisfies these constraints will not trigger integer overflow errors at memory allocation and block copy sites of the application.
3) JSAND [18] is a publicly available anomaly detector for JavaScript. It has a web interface [9] where users can submit JavaScript programs. The system will generate a report for each submitted program and classify the program as normal, suspicious, or malicious.

[2] http://aws.amazon.com/s3/pricing

A. Methodology

Collect Sample Inputs: Fortuna uses its sampling algorithm to collect JPEG, PNG, and JavaScript files from the Internet. Using this sampling algorithm, Fortuna collected 42299 JPEG files, 64089 PNG files, and 8853 JavaScript files in less than 10 hours.

Set up Anomaly Detectors: In our experiments, we applied SIFT to dillo [5] and png2swf [8] for PNG files, as well as jpeg2swf [8] for JPEG files. We applied SOAP to dillo [5] for PNG files, as well as ImageMagick [7] for JPEG files. We selected these applications because they were the benchmark applications in the original papers of SOAP [37] and SIFT [38]. For the SIFT experiments, we ran the SIFT static analysis on the source code of the applications to obtain our input constraints. For the SOAP experiments, we randomly selected 5130 PNG files and 3386 JPEG files from the collected files as training examples to generate our input constraints.

Test Anomaly Detectors: We tested the constraints that SOAP and SIFT generated on the collected JPEG and PNG files. We excluded the files we selected for training examples in the SOAP experiments. We also tested JSAND on the collected JavaScript files. We report the false positive rate bounds that Fortuna computed for these anomaly detectors.

Note that the distribution sampled by Fortuna differs in practice from the ideal PageRank distribution in two ways. First, we do not visit some pages indicated by `/robots.txt` files. And second, when a website requires a login (e.g., a social network or a paywall) our "random surfer" cannot follow the link. These are unavoidable, inherent limitations of any automated system for collecting inputs from the web.

B. Results

Table I summarizes our experimental results. There is a row in the table for each combination of anomaly detection system and application (SIFT) or anomaly detection system and input file format (SOAP and JSAND). The first column lists the anomaly detection system we are testing, which is either SIFT, SOAP, or JSAND. The second column indicates the input format, which is either JPEG, PNG, or JavaScript. The third column indicates the application utilizing the inputs (thus, the target of the analysis) in each SOAP or SIFT experiment. The fourth column indicates the number of training example inputs used in each SOAP experiment. The fifth column presents the number of collected inputs used for testing in each experiment. The sixth column presents the number of false positives we encountered in each experiment.

The seventh column presents the corresponding theoretical bounds on the false positive rate for the anomaly detection system in each experiment. Each bound is either of the form "$err^{(1)} < X\%$" (one-sided) or of the form "$X\% \leq err^{(1)} \leq Y\%$" (two-sided). For the systems with zero observed false positives – SIFT and JSAND with conservative filtering (i.e. JavaScript that triggers strong warnings is rejected) – we use Theorem 5, obtaining a bound of the form, "the Type I error is < X%." For the systems for which false positives were observed – SOAP and JSAND with aggressive filtering (i.e. JavaScript that triggers any warning is rejected) – we use Hoeffding's bound (Theorem 3) to obtain a bound of the form, "the Type I error is between X% and Y%." In all cases, we used a confidence bound of 99% ($\delta = 1\%$).

Note that SOAP has a higher false positive rate that SIFT or JSAND. A higher false positive rate is acceptable for SIFT because it does not discard false positives. It instead applies input rectification, which changes the input file but still presents the rectified input file to the application.

C. Comparison to Ad Hoc Methods

To emphasize the importance of using PageRank in place of an ad hoc crawling method, we also collected data via the web crawling method used to originally gather test data for the SOAP anomaly detector [37]. The method initiated a breadth first web crawl from a chosen site, downloading all files in all descendent sites. This sort of method has been widely used for collecting data for testing anomaly detectors in the past. In a separate test from that used for Table I, we trained SOAP using 7000 JPEG files before evaluating on PageRank data and data collected from the ad hoc method. The later approach significantly underestimated false positive rate in comparison to testing with PageRank – for ImageMagick we saw a rate of 0.66% in comparison to a rate of 1.15%. Similarly, we trained SOAP using 3999 PNG files. Again, testing with the ad hoc method significantly underestimated false positive rate – for dillo we saw a rate of 0.14% in comparison to 0.41% for PageRank. Using a standard Pearson's chi-squared test, these experiments were determined to be statistically significant to within a p-value of .01.

We conclude that the naive method originally employed for testing SOAP does not provide an adequate test for false positive rate. Specifically, it seems likely that the method does not collect a wide enough variety of data, thus underestimating false positive rate. PageRank, on the other hand, ensures wide coverage of the web.

VII. Related Work

Intrusion Detection: Errors or vulnerabilities in deployed software are often triggered by atypical inputs that the application code does not properly process or reject. Initially introduced by Denning [21], Anomaly Detection techniques [20], [31], [44], [46], [47], [48], [49], [51] learn over a set of benign inputs to generate a classifier that can detect potentially malicious inputs to the system. For example, Kruegel et al. [31] learn character distributions of benign HTTP traffic to detect potential malicious HTTP requests. Wang et al. [51] propose a similar technique that looks at character distributions in payloads to detect network intrusions. Ntoulas et al. [44] propose to detect malicious web pages by learning features like word and anchor text length. Cova et al. [18] statically analyze benign JavaScript programs to extract features such as string lengths and character distributions to detect malicious JavaScript programs. SOAP [37] is an automatic input rectification system that learns a set of input constraints from training inputs and then enforces these learned constraints by rectifying any input that violates the learned constraints. Chandola et al. [16] summarizes a good overview on these anomaly detection systems.

Anomaly detection systems have also been built that rely on static analysis of application source code [23], [25], [38], [50],

System	File Format	Target Application	# Training Examples	# Test Inputs	# Encountered False Positives	Bound on False Positive Rate (with 99% confidence)
SIFT	JPEG	jpg2swf	N/A	42299	0	$err^{(1)} < 0.0108\%$
SIFT	PNG	dillo	N/A	64089	0	$err^{(1)} < 0.0072\%$
SIFT	PNG	png2swf	N/A	64089	0	$err^{(1)} < 0.0072\%$
SOAP	JPEG	ImageMagick	3386	38913	775	$1.16\% \leq err^{(1)} \leq 2.82\%$
SOAP	PNG	dillo	5130	58959	172	$0\% \leq err^{(1)} \leq 0.96\%$
JSAND w/ Conservative Filtering	JavaScript	N/A	N/A	8853	0	$err^{(1)} < 0.0520\%$
JSAND w/ Aggressive Filtering	JavaScript	N/A	N/A	8853	12	$0\% \leq err^{(1)} \leq 1.87\%$

TABLE I: Experimental Results. We obtain strong bounds on Type I error (False Positive Rate) for anomaly detectors processing several file types. Each bound holds with 99% confidence.

[15]. Specifically, researchers have developed techniques that use pushdown automata (PDA) to model invalid program paths that can only occur if being exploited [23], [25], [50]. Precondition analysis has also been used to directly synthesize input constraints that can eliminate certain classes of errors [38], [15].

In recent years, Anomaly Detection Systems have increasingly been deployed to handle data from the web. Some common applications include detecting malicious websites [44], [48], automatically neutralizing malicious JavaScript code [18], [20], [28], [36], and filtering anomalies in more general data types that, in today's world, are usually acquire from the web [37], [38].

Rigorous ADS Testing: It has been recognized in the anomaly detection literature that drawing test data (and training data) from a well defined distribution over "normal" inputs is desirable. However, as Chandola et al. [16] point out, while statistical techniques for ADS are based on the assumption of an underlying input distribution, it is often difficult or impossible to define, let alone sample, from this distribution. Thus, samples for testing are typically collected with a best effort approach, with the focus placed on detection techniques. As recognized by Gao et al. [24]:

> *It is not our goal here to determine how to acquire adequate training data for a program. Rather, we simply assume we have adequate training data in our study; if this is not true, our techniques might yield false detections, i.e., they may detect anomalies that are not, in fact, intrusions.*

With respect to collecting representative samples from the web, while we are not aware of any work that samples directly from the PageRank distribution, several works on anomaly detection have employed crawl based methods for collecting data [20], [37], [36], [48]. It has been recognized that simply drawing webpages from a known list (i.e. search engine indexes) gives a limited diversity of input samples [18]. Crawling helps ameliorate that issue.

Ntoulas et al. [44] further recognize that crawling biases samples towards "high quality," frequently visited webpages. This is a key motivation in our work, as a distribution weighted by such features more accurately represents the pages a typically user is expected to encounter. While motivated by similar intuition, Ntoulas et al. [44] do not formalize their input distribution. They also do not provide any guarantees (such as bounds on the false positive rate) about the performance of the anomaly detector.

PageRank: The PageRank distribution has been well studied in both theoretical and practical literature. Berkhin [13] provides a valuable survey. Other candidate representative distributions have been proposed for the web but, largely due to Google's success, PageRank has risen as a clear leader in site ranking algorithms. One of the earliest alternatives is the Hypertext Induced Topic Selection (HITS) algorithm introduced by Kleinberg [30]. This algorithm is considered a precursor to PageRank and was the first approach to weight links by "authority", something PageRank does implicitly through its random surfer model.

Part of the early motivation for PageRank was ease of computation [45]. Several results have focused on accelerated approaches [17], [33] since a full list of probabilities is required in actually using the distribution to rank webpages (i.e. a set returned from a web search). We are not aware of prior applications of sampling webpages by PageRank. Sampling has been considered in a more theoretical context for collecting a representative group of nodes from a general network [35].

VIII. Conclusion

Anomaly detectors are a critical component of many security systems. Despite the central role that they play, anomaly detectors are often designed and deployed with no precise understanding of their behavior. We present a novel technique and implemented system, Fortuna, for obtaining probabilistic bounds on the actual false positive rate that deployed anomaly detectors will incur. We demonstrate that Fortuna can provide tight bounds for three anomaly detectors designed to process input files from the Internet. Fortuna takes an important step towards placing anomaly detection on a firmer theoretical foundation and is designed to help practitioners better understand the behavior and consequences of the anomaly detectors that they build and deploy.

Acknowledgment

The authors would like to thank our anonymous reviewers for many helpful comments and suggestions. This research was supported by DARPA grant No. FA8650-11-C-7192. B. Juba was supported by ONR grant No. N000141210358. C. Musco was partially supported by NSF Graduate Research Fellowship grant No. 1122374.

References

[1] "Alexa Internet," http://www.alexa.com/.

[2] "Alexa Top Million Sites," http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[3] "Common Crawl," http://commoncrawl.org/.

[4] "Common Crawl URL Index," https://github.com/trivio/common_crawl_index.

[5] "Dillo," http://www.dillo.org/.

[6] "Google Toolbar," http://toolbar.google.com/.

[7] "Imagemagick," http://www.imagemagick.org/.

[8] "Swftools," http://www.swftools.org/.

[9] "Wepawet," https://wepawet.cs.ucsb.edu/.

[10] "Netcraft Web Server Survey," http://www.alexa.com/, June 2014.

[11] R. Andersen, F. Chung, and K. Lang, "Using PageRank vectors to locally partition graphs," *Internet Mathematics*, vol. 4, no. 1, pp. 35–64, 2007.

[12] V. S. Arvind Narayanan, "Robust de-anonymization of large sparse datasets (how to break anonymity of the netflix prize dataset)," in *IEEE Security and Privacy (Oakland)*, 2008, pp. 111–125.

[13] P. Berkhin, "A survey on PageRank computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[14] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Techn.*, vol. 5, no. 1, pp. 92–128, 2005.

[15] D. Brumley, H. Wang, S. Jha, and D. Song, "Creating vulnerability signatures using weakest preconditions," in *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, ser. CSF '07'. Washington, DC, USA: IEEE Computer Society, 2007, pp. 311–325. [Online]. Available: http://dx.doi.org/10.1109/CSF.2007.17

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[17] Y.-Y. Chen, Q. Gan, and T. Suel, "I/O-efficient techniques for computing pagerank," in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, ser. CIKM '02. ACM, 2002, pp. 549–557. [Online]. Available: http://doi.acm.org/10.1145/584792.584882

[18] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10'. New York, NY, USA: ACM, 2010, pp. 281–290. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772720

[19] A. Croll and S. Power, *Complete Web Monitoring - Watching your visitors, performance, communities and competitors.* O'Reilly, 2009.

[20] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "Zozzle: Fast and precise in-browser javascript malware detection," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028070

[21] D. Denning, "An intrusion-detection model," *Software Engineering, IEEE Transactions on*, vol. SE-13, no. 2, pp. 222–232, Feb 1987.

[22] A. Farahat, T. Lofaro, J. C. Miller, G. Rae, and L. A. Wart, "Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness and effect of initialization," *SIAM J. Sci. Comput.*, vol. 27, no. 4, pp. 1181–1201, 2006.

[23] H. Feng, J. Giffin, Y. Huang, S. Jha, W. Lee, and B. Miller, "Formalizing sensitivity in static analysis for intrusion detection," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, May 2004, pp. 194–208.

[24] D. Gao, M. K. Reiter, and D. Song, "On gray-box program tracking for anomaly detection," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. USENIX Association, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251375.1251383

[25] J. T. Giffin, S. Jha, and B. P. Miller, "Detecting manipulated remote call streams," in *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 61–79. [Online]. Available: http://dl.acm.org/citation.cfm?id=647253.720282

[26] T. H. Haveliwala and S. D. Kamvar, "The second eigenvalue of the Google matrix," Stanford University, Tech. Rep. 582, 2003.

[27] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[28] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna, "Revolver: An automated approach to the detection of evasive web-based malware," in *Proceedings of the 22Nd USENIX Conference on Security*, ser. SEC'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 637–652. [Online]. Available: http://dl.acm.org/citation.cfm?id=2534766.2534821

[29] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. Cambridge, MA, USA: MIT Press, 1994.

[30] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/324133.324140

[31] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. ACM, 2003. [Online]. Available: http://doi.acm.org/10.1145/948109.948144

[32] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton, NJ: Princeton University Press, 2006.

[33] C. P. Lee, "A fast two-stage algorithm for computing pagerank and its extensions," Tech. Rep., 2003.

[34] M. Leonard, "Flawed process," *Indiana Herald-Times*, August 10–12, 2008.

[35] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 631–636. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150479

[36] Z. Li, S. Alrwais, X. Wang, and E. Alowaisheq, "Hunting the red fox online: Understanding and detection of mass redirect-script injections," in *Proceedings of 2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2014.

[37] F. Long, V. Ganesh, M. Carbin, S. Sidiroglou, and M. Rinard, "Automatic input rectification," in *Proceedings of the 2012 International Conference on Software Engineering*, ser. ICSE 2012. IEEE Press, 2012, pp. 80–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337233

[38] F. Long, S. Sidiroglou-Douskos, D. Kim, and M. Rinard, "Sound input filter generation for integer overflow errors," in *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '14'. New York, NY, USA: ACM, 2014, pp. 439–452. [Online]. Available: http://doi.acm.org/10.1145/2535838.2535888

[39] M. W. Mahoney and L. Orecchia, "Implementing regularization implicitly via approximate eigenvector computation," in *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, 2011, pp. 121–128.

[40] M. R. Meiss, B. Gonçalves, J. J. Ramasco, A. Flammini, and F. Menczer, "Modeling traffic on the web graph," in *Proc. 7th Workshop on Algorithms and Models for the Web Graph (WAW)*, R. Kumar and D. Sivakumar, Eds. Berlin / Heidelberg: Springer, 2010, pp. 50–61.

[41] M. R. Meiss, F. Menczer, S. Fortunato, A. Flammini, and A. Vespignani, "Ranking web sites with real user traffic," in *Proc. WSDM'08*, 2008, pp. 65–75.

[42] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *IEEE Security and Privacy*, 2009, pp. 173–187.

[43] ——, "Myths and fallacies of personally identifiable information," *Communications of the ACM*, vol. 53, no. 6, pp. 24–26, 2010.

[44] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *Proceedings of the 15th International Conference on World Wide Web*, ser. WWW '06'. New York, NY, USA: ACM, 2006, pp. 83–92. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135794

[45] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," in *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998, pp. 161–172. [Online]. Available: citeseer.nj.nec.com/page98pagerank.html

[46] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "Mcpad: A multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864 – 881, 2009, traffic Classification and Its Applications to Modern Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128608003927

[47] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516682

[48] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, ser. SP '11'. Washington, DC, USA: IEEE Computer Society, 2011, pp. 447–462. [Online]. Available: http://dx.doi.org/10.1109/SP.2011.25

[49] F. Valeur, D. Mutz, and G. Vigna, "A learning-based approach to the detection of sql attacks," in *DIMVA 2005*, 2005.

[50] D. Wagner and D. Dean, "Intrusion detection via static analysis," in *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, 2001, pp. 156–168.

[51] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *RAID*, 2004.