

Impact of Packet Sampling on Portscan Detection

Jianning Mai, *Student Member, IEEE*, Ashwin Sridharan, *Member, IEEE*, Chen-Nee Chuah, *Member, IEEE*, Hui Zang, *Senior Member, IEEE*, and Tao Ye, *Member, IEEE*

Abstract—Packet sampling is commonly deployed in high-speed backbone routers to minimize resources used for network monitoring. It is known that packet sampling distorts traffic statistics and its impact has been extensively studied for traffic engineering metrics such as flow size and mean rate. However, it is unclear how packet sampling impacts anomaly detection, which has become increasingly critical to network providers. This paper is the first attempt to address this question by focusing on one common class of non-volume based anomalies, *portscans*, which are associated with worm/virus propagation. Existing portscan detection algorithms fall into two general approaches: target-specific and traffic profiling. We evaluated representative algorithms for each class, namely (a) *TRWSYN* that performs stateful traffic analysis, (b) *TAPS* that tracks connection pattern of scanners, and (c) *Entropy-based* traffic profiling. We applied these algorithms to detect portscans in both the original and sampled packet traces from a Tier-1 provider’s backbone network. Our results demonstrate that sampling introduces fundamental bias that degrades the effectiveness of these detection algorithms and dramatically increases false positives. Through both experiments and analysis, we identify the traffic features critical for anomaly detection that are affected by sampling. Finally, using insight gained from this study, we show how portscan algorithms can be enhanced to be more robust to sampling.

Index Terms—sampling, portscan detection, TRW, TAPS, entropy-based profiling.

I. INTRODUCTION

Traffic measurement and analysis are essential for effective traffic engineering (e.g., traffic matrix estimation, route optimization), capacity planning, and more recently, network security (e.g., anomaly detection). However, detailed payload capture of backbone traffic does not scale with the high link speeds. Therefore, packet sampling such as Cisco’s NetFlow [1] and Juniper’s Traffic Sampling [2], is often deployed in the routers and the *thinned* traffic is taken as the input for analysis. With the increasing impact of network wide attacks, this thinned traffic is now being extensively used to detect network anomalies. Two classes of anomalies that are most studied in the intrusion detection literature are volume-based anomalies, e.g., Denial-of-Service (DoS) attacks, and port scanning, which typically precedes worm/virus propagation. There are a number of reasons for detecting anomalies such as worm scans in the backbone transit networks, as opposed to at the stub networks. An Internet Service Provider (ISP) is often interested in anomalous traffic originating from its peers. A transit network carries a far more diverse traffic mix and can detect a wider range of scanning activities than an individual

stub network. For example, the study in [3] suggests that a global view of the traffic could better capture the scanning patterns. Finally, a stub network such as an enterprise may out source the detection task to its upstream provider due to lack of resources or expertise.

The impact of sampling has been extensively studied in terms of well known statistical metrics, e.g., mean rate and flow size distribution, from the perspective of determining the volume characteristics of the traffic as a whole [4]–[7]. However, anomaly detection (e.g., worm scan detection) often depends on a diverse set of metrics such as address access pattern, connection status, and distinct per source behaviors. How packet sampling impacts these traffic features has not been previously addressed. This paper presents a first attempt to address this important open question: *Does packet sampling distort or lose pertinent information from the original traffic profile that affects the effectiveness of existing anomaly detection techniques? If so, by how much?*

There is a rich set of literature on two general approaches to anomaly detection: specialized detection algorithms that target specific types of anomalies, and generalized traffic profiling algorithms. Example target specific algorithms include [8]–[10] designed primarily to detect portscans. On the other hand, traffic classification algorithms, such as [11], [12], are generalized algorithms that do not target a specific anomaly. Instead they classify different traffic features and raise alarm flags when they detect large variations. Algorithms from both categories typically assume the availability of detailed packet payload, e.g., at the network edge. However, it is not clear how their performance is impacted if the same solutions utilize only sampled packet header data.

We note that it is clearly infeasible to perform an exhaustive study on the impact of sampling for every anomaly detection algorithm presented in literature. Instead, we focus on one common class of non-volume based anomalies, *portscans*, which causes increasing security concerns. We choose representative algorithms aimed at portscan detection from the two categories of detection algorithms mentioned above. Specifically, this paper presents a detailed study that quantifies the effect of packet sampling on two target-specific and one traffic-profiling algorithms: (a) Threshold Random Walk (TRW) [9], (b) Time Access Pattern Scheme (TAPS) [10], and (c) Entropy-based behavior modeling proposed recently [11].

TRW performs stateful analysis of the traffic to identify connection status, while TAPS exploits the knowledge of the “connection patterns” of scanners. The general traffic profiling algorithms compute entropy values of each of the four “features” of the IP header in order to identify “significant flows” and capture abrupt changes in the feature set. We believe that these algorithms cover a wide range of anomaly

Manuscript received October 1, 2005; revised May 1, 2006. This work was supported in part by NSF CAREER Award #0238348 and Sprint Advanced Technology Labs.

J. Mai and C.-N. Chuah are with University of California, Davis. A. Sridharan, H. Zang, and T. Ye are with Sprint Advanced Technology Labs.

pattern metrics like connection rate, scanning rate, address range distribution, and flow number distribution that would be typically used for anomaly detection. Our analysis not only characterizes the impact of sampling on the performance of these algorithms but also tries to identify how the critical decision metric is affected. Hence, we expect our results to be widely applicable in terms of inferring how other algorithms would perform using sampled traffic.

Our data set consists of GPS-timestamped packet header traces captured from Sprint’s backbone and wireless networks, which are then sampled at different rates. Each of these portscan detection techniques uses the original as well as the sampled traffic traces as input. Further details of the methodology are presented in Section II. By comparing the performance of these three schemes under different sampling rates, we can quantify the potential distortion introduced by sampling on traffic “fingerprints” that are pertinent for anomaly detection.

The contributions of this paper are summarized as follows:

- Through experiments using real traces, we quantify how sampling affects the performance of the portscan detection algorithms as well as the entropy-based traffic profiling techniques. We show that not only does the successful detection rate drop as sampling interval increases (which should be expected), but also that the false positive rate increases for all the techniques studied. This is undesirable from a network administrator point of view. For example, in one of our case studies, the false positive rate of TRW-based portscan detection increases by eight fold when the input trace is sampled at rate 1/10.
- Through analysis, we pinpoint the specific “distortion” of traffic features due to sampling that have adverse impact on the effectiveness of different detection schemes. We demonstrate that sampling causes the following fundamental bias in data: (a) flow thinning (smaller flow size distribution) that causes high false positive and false negative rates for scanner detection that relies on statistical hypotheses testing (Section III), and (b) traffic feature set distortion towards either uniformity or biased towards large sized flows, which reduces the number of significant events detected and increases false positives (Section IV).
- We discuss the implications of our results on designing anomaly detection and sampling algorithms. For portscan detection, our analysis indicates that inferences based on connection states and access patterns complement each other under packet sampling. Utilizing this insight, we were able to develop a TCP portscan detection algorithm whose performance is robust to sampling.

The remainder of the paper is organized as follows. Section II discusses related work, experimental data, and specific algorithms we analyzed. In Section III, we compare the impact of sampling on two target-specific portscan detection schemes (TRW and TAPS). We also present detailed analysis to explain the observed effect of sampling on the detection rate and false positives and introduce a new TCP portscan detection algorithm. Section IV explores the impact of sampling on entropy-based traffic profiling algorithms. We discuss the implications of this study in Section VI.

II. BACKGROUND AND METHODOLOGY

A. Sampling

Sampling is typically used in network measurements to avoid large memory and CPU processing requirements on routers as well as the high bandwidth required to transport flow records [7]. Two types of sampling that have been widely discussed in literature are: packet sampling and flow sampling. Packet sampling is simple to implement with low CPU power and memory requirements. However, extensive research has shown it to be inaccurate for inference of flow statistics like size distribution of the original flows [4], [5]. Adaptive packet sampling techniques that change the sampling rate to further reduce memory consumption and limit loss of existing accuracy have been presented in [13], [14]. The former improves the performance of NetFlow, while the latter proposes techniques for varying sampling rate to control the variance. Flow sampling has been proposed as an alternative to overcome the limitations of packet sampling. It is shown to improve accuracy [5] but suffers from prohibitive memory and CPU power requirements. Though reduced complexity flow sampling techniques like Sample-and-hold [15] have been proposed, they are still too complex for high speed links. Consequently, in practice, packet sampling is the predominant method of choice for high speed backbone links. We focus on packet sampling for the rest of our discussion in this paper.

B. Portscan Detection

Several portscan detection techniques have been proposed in literature. Snort [8] is a flexible rule language that issues alerts based on user-defined connection patterns and rates. Hence we view Snort as more of a system rather than technique. SPICE [16] is a complex off-line technique based on Bayesian analysis that can detect stealthy portscans. The focus of this paper, however, is on two “on-line” portscan detection techniques: TRW [9] (Threshold Random Walk) and TAPS [10] (Time Access Pattern Scheme), which can process traffic in real-time to identify portscans and have been shown to be highly effective. In addition, we will also evaluate an entropy-based portscan detection technique proposed in [17]. We describe the three algorithms in more detail below.

TRW and TRWSYN: TRW performs hypotheses testing on the observed connection status of each source to decide whether it is more likely to be a scanner or a benign host. Specifically, let H_0 represent the hypothesis that a source is a benign host, and H_1 be the hypothesis that a source is a scanner. Each potential observable event is assigned a different likelihood probability under the two different hypotheses. For example, in TRW, the observation used is the *connection status*. Let

$$Y_i = \begin{cases} 0 & \text{if connection successful} \\ 1 & \text{if connection failed} \end{cases}$$

denote the event variable associated with each connection. Each of the two probable events is given a specific probability under each given hypothesis $H_j, j = 0, 1$, e.g., $\theta_0 = P_r[Y_i = 0|H_0]$ and $\theta_1 = P_r[Y_i = 0|H_1]$. For each observed event $i = 1, 2, \dots$, the test updates the likelihood ratio defined as:

$$\Lambda(Y) = \prod_{i=1}^n \frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]}. \quad (1)$$

As the likelihood ratio increases/decreases with each observed event i , it may cross either one of the pre-specified thresholds $\eta_0 < 1 < \eta_1$. If $\Lambda(Y) > \eta_1$, the source IP belongs to set H_1 ; else if $\Lambda(Y) < \eta_0$ it is in set H_0 . The choice of the two thresholds defines the accuracy of the test. It has been shown that TRW only requires a low number (~ 6) of observed events to detect scanners successfully [9]. Note that for the test to yield quick results, θ_0 must substantially differ from θ_1 .

The authors of [10] adapted TRW to the backbone scenario by modifying the definitions for connection “success” and “failure”. The new definition marks flows consisting of a single TCP SYN packet as failed connections. Otherwise, the flows are considered successful connections. We refer to this adaptation as TRWSYN, and evaluate its ability to detect portscans in the backbone trace.

TAPS: TAPS, proposed in [10], is a rate limiting scheme that is designed for portscan detection in the backbone. Specifically, it utilizes the *access pattern* of each source for hypotheses testing. The intuition behind this approach is that scanner access a larger spread of destination IP addresses (or ports) as compared to non-scanners. To quantify this notion, let C_{DstIP} (or C_{DstPORT}) be the cardinality of the set of addresses (or ports) accessed by a source in a given time bin t . The number of *distinct connections*, approximated by $\max(\frac{C_{\text{DstIP}}}{C_{\text{DstPORT}}}, \frac{C_{\text{DstPORT}}}{C_{\text{DstIP}}})$, is $\gg 1$ for scanners and quite small for non-scanners (see [10] for empirical evidence).

TAPS utilizes time bins as the event generation mechanism for its hypotheses testing, i.e., the event variable Y_t is associated with a time bin t . At the end of each time bin (say t), the $\frac{C_{\text{DstIP}}}{C_{\text{DstPORT}}}$ ratio (or its reciprocal, whichever is larger) is calculated for each source IP. Y_t is then accordingly set to either 0 or 1 depending on whether or not the ratio exceeds a predefined threshold k . The likelihood ratio, Eqn. (1), for the source is then updated with the corresponding probabilities of the observed event Y_t under hypotheses H_0 and H_1 . In TAPS the time bin setting plays a crucial role in defining the detection success rate, while the choice of the threshold is important in deciding the false positive rate.

Entropy-based Scan Detection: Recently, entropy-based techniques have been proposed [11], [12] to profile traffic and detect anomalies. These techniques are motivated by the observation that a majority of anomalous and interesting events induce a change in the distribution of traffic communication pattern, which can be captured by the *sample entropy*. We will explain in detail how sample entropy is computed and applied to portscan detection.

Let X be a discrete random variable that is observed \mathcal{N} times and takes values from the set $\{x_i\}$ of cardinality M . Let n_i denote the number of times it takes value x_i during the observation window. The sample entropy is then defined as:

$$H(X) = - \sum_{i=1}^M p(x_i) \log_2 p(x_i), \quad (2)$$

where $\mathcal{N} = \sum_{i=1}^M n_i$, and $p(x_i) = n_i/\mathcal{N}$ stands for the sample probability of $X = x_i$. Note that $0 \leq H(X) \leq \log_2 M$. $H(X) = 0$ if X always takes the same value and $H(X) = \log_2 M$ if X is uniformly distributed, i.e., $p(x_i) = 1/M, i = 1, \dots, M$. For traffic analysis purposes, X represents the number of flows with a particular traffic feature, or *dimension*. Four typical “dimensions” are the number of flows for a given source IP address (SrcIP), destination IP address (DstIP), source port number (SrcPORT), and destination port number (DstPORT), respectively.

Both [11], [12] share the same entropy definition, but they take different approaches to profiling and mining the traffic data. In [12], the entropy time series along each of the four dimensions mentioned above (*SrcIP, DstIP, SrcPORT, DstPORT*) is constructed across network-wide traffic, and principal component analysis (PCA) is applied to expose unusual traffic behavior. The authors of [11] emphasize the analysis of communication patterns for *significant clusters* (SCs), which are defined as entities that contribute a large mass to the sample distribution along each dimension.

Below, we summarize the method of [11] and the guidelines in [17] on using entropy to detect portscans. To extract significant clusters, the set $\{x_i, 1 \leq i \leq M\}$ is first rearranged in decreasing order as $\{x'_i\}$ according to probability distribution $p(x_i)$. The *Relative Uncertainty* (RU) is then defined as:

$$RU(X) = \frac{H(X)}{H_{\max}(X)} = \frac{- \sum_{i=1}^M p(x_i) \log p(x_i)}{\log_2 |M|}. \quad (3)$$

RU is actually a relative entropy and normalizes the degree of uniformity of the distribution. This is because $0 \leq RU(X) \leq 1$, with $RU(X) = 0$ if X takes a single value, and $RU(X) = 1$ if X has a uniform distribution over the sample space. The top ones in the set $\{x'_i\}$ are then extracted one by one and labeled as SCs until the remaining set X_r has an $RU(X_r)$ close to 1, i.e., almost uniformly distributed and hence not significant. The threshold that indicates $RU(X_r) \approx 1$ is denoted as β , and the default value of β picked for SC extraction is $RU(X_r) > \beta = 0.9$ [11].

Clusters are then categorized based on similarity or dissimilarity of communication patterns into *behavior classes* (BCs). For example, for each SC in *SrcIP* dimension, we calculate $RU(\text{SrcPORT})$, $RU(\text{DstPORT})$, and $RU(\text{DstIP})$, and mapped the values to three levels: 0 (low), 1 (medium), and 2 (high) denoted by L_{RU} . The clusters can be categorized into different BCs represented by a vector of $[L_{RU(\text{SrcPORT})}, L_{RU(\text{DstPORT})}, L_{RU(\text{DstIP})}]$. Sources correspond to BCs of $[*, 2, 0]$ or $[*, 0, 2]$ are scanners that send probes to a large number of random ports on a few IP addresses, or to a few particular port numbers on lots of random IP addresses.

C. Trace Data and Experiment Setup

Our experiments used packet traces collected on two links in a Tier-1 ISP’s backbone network: (**BB-West**) an OC-48 link between two backbone routers on the west coast, and

(**Wireless**) a link from a gateway router to a nation-wide cellular network. These traces are collected by IPMON [18], a passive monitoring system that captures the first 44 bytes of the IP header of every IP packet traversing a monitored link. The *Wireless* trace has a relatively low traffic volume but contains a large percentage of scanning traffic, making it an interesting trace for portscan analysis. Statistics of the traces are presented in Table I.

TABLE I
TRACE DATA STATISTICS

Trace	Date	Average Rate	Duration
BB-West	03-08-2003	55 Mbps	1 hour
Wireless	04-01-2004	7 Mbps	3 hours

We generated *sampled traces* from the *original traces* using both systematic packet sampling and simple random packet sampling [19], [20]. Experiments where the same detection algorithm is applied to traces sampled with the two different techniques did not result in noticeable differences. Hence we present the results only for the simple random sampling scheme, where a packet is selected with probability $p = 1/N$ from the *original trace* to form the *sampled trace*. We use a set of average sampling intervals N chosen from the vector [10, 20, 50, 100, 200, 500, 1000]. Similar to NetFlow, traffic was classified into flows based on the well-known five-tuple: (source IP address, destination IP address, source port, destination port, protocol). Flows were terminated by either a default timeout of 1 minute or the TCP protocol semantics.

III. TRWSYN AND TAPS UNDER PACKET SAMPLING

We begin by studying the impact of packet sampling on the performance of TRWSYN and TAPS since they share similar features. Both algorithms have been shown to work very well in their respective target environments when complete packet traffic capture is possible. However, packet sampling may distort traffic fingerprints used by such algorithms, resulting in fundamental inaccuracies. In the following subsections, we validate this through experimental evaluation of TRWSYN and TAPS using traffic traces sampled at different rates and analyze the source of such inaccuracies.

We quantify the performance of these algorithms using the following metrics, which were previously defined in [10]:

$$\begin{aligned} \text{Success Ratio} \quad R_s &= \frac{\text{No. of true scanners detected}}{\text{No. of true scanners}} \\ \text{False Negative Ratio} \quad R_{f-} &= \frac{\text{No. of true scanners missed}}{\text{No. of true scanners}} \\ \text{False Positive Ratio} \quad R_{f+} &= \frac{\text{No. of false scanners detected}}{\text{No. of true scanners}}. \end{aligned}$$

The *success ratio* R_s measures the effectiveness of a detection algorithm, while the *false positive ratio* R_{f+} indicates its correctness. High R_s and low R_{f+} are desirable properties of an anomaly detection algorithm. Since $R_s + R_{f-} = 1$, we only discuss the success and false positive ratios.

Note that these metrics are not specific to any particular detection technique, hence can be used to evaluate different algorithms under various sampling rates. However, computation of the metrics requires a priori knowledge of the true scanners, which is a challenging task. Given the immense variation

in Internet traffic patterns and protocol state, it is virtually impossible to guarantee that any current portscan detection algorithm can correctly identify *all* scanners in a traffic stream. Consequently, one must resort to an approximation of the ground truth (true scanners) through bootstrapping.

In [10], for each traffic trace, the authors applied several portscan algorithms with various parameter settings to identify a superset of potential scanners. Each source from this superset was then manually verified by checking its flow size, scanning rate, and destination address spread against the intuitive notion of how a scanner should behave. We use the final list of scanners generated by this process as the ground truth. The scanner list returned by any specific portscan algorithm is compared against this ground truth to determine the successful detections, false negatives, and false positives. We note that similar bootstrapping methods have also been used previously in [21] and [9]. Even though the ground truth is an approximation of the *actual* set of scanners, we believe it suffices for the purpose of this work. This is because our focus is on the *relative* performance of an algorithm as a function of packet sampling rather than its *absolute* accuracy.

To validate our inferences, we have also conducted a *controlled* experiment where artificial scanners were generated using Nmap [22] and combined with the BB-West trace using TRReplay [23]. For these experiment, the ground truth was assumed to comprise of the artificially injected scanners only and the performance of TRWSYN and TAPS was evaluated on the mixed trace sampled at different rates.

A. Impact of Sampling on TRWSYN

Recall that TRWSYN associates a probability $1 - \theta_1$ ($1 - \theta_0$) of a failed connection with a scanner (benign host), as described in Section II. As we vary the sampling rate, the hypotheses test parameters $1 - \theta_1 = 0.8$ and $1 - \theta_0 = 0.2$ used for non-sampled traffic [9], [10] are kept unchanged. This was done primarily to observe how an algorithm designed for non-sampled traffic performs under packet sampling. Later in this section, we analyze how these parameters are affected by sampling and explain why even modifying them may not yield the desired results.

Tables II and III summarize the R_s and R_{f+} ratios at different sampling intervals N for the *Wireless* and *BB-West* traces, respectively. Surprisingly, for both traces, the success ratio R_s of TRWSYN initially *increases* (albeit by a small amount) for low sampling intervals before dropping off for larger values of N as the traffic is increasingly thinned. For example, R_s for *Wireless* trace initially increases from 77% for the non-sampled trace to 95.4% with $N = 10, 20, 50$ before dropping off to 70% at $N = 1000$ ¹. While this may seem advantageous, we observe that the false positive ratio R_{f+} also follows similar behavior but on a much larger scale. It increases from 30% for the original trace to 250% for the sampling interval $N = 10$, then drops back to 30% at $N = 1000$. Specifically, the false positive ratio increases by

¹The *Wireless* trace contains a large number of scans, comprising up to 30% of the instantaneous flow rate. Hence, even with $N = 1000$, we do not see a severe degradation in results as seen in the *BB-West* Trace.

TABLE II
TRWSYN DETECTION RESULTS FOR THE *Wireless TRACE* (GROUND TRUTH = 65)

Sampling Rate	Original	N=10	N=20	N=50	N=100	N=200	N=500	N=1000
Total Detections	70	229	196	134	120	94	82	65
Success (R_s)	50 (76.9%)	62 (95.4%)	62 (95.4%)	62 (95.4%)	57 (87.7%)	55 (84.6%)	53 (81.5%)	45 (69.2%)
False + (R_{f+})	20 (30.8%)	167 (256.9%)	134 (206.2%)	72 (110.8%)	63 (96.9%)	39 (60.0%)	29 (44.6%)	20 (30.8%)

TABLE III
TRWSYN DETECTION RESULTS FOR THE *BB-West TRACE* (GROUND TRUTH = 447)

Sampling Rate	Original	N=10	N=20	N=50	N=100	N=200	N=500	N=1000
Total Detections	679	1362	1033	687	528	388	247	152
Success (R_s)	356 (79.6%)	367 (82.1%)	356 (79.6%)	340 (76.1%)	309 (69.1%)	276 (61.7%)	203 (45.4%)	136 (30.4%)
False + (R_{f+})	323 (72.3%)	995 (222.6%)	677 (151.5%)	347 (77.6%)	219 (49.0%)	112 (25.1%)	44 (9.8%)	16 (3.6%)

a huge factor for low sampling intervals, indicating a large number of sources erroneously tagged as scanners, before decreasing with higher values of N .

The observed behavior can be attributed to *flow thinning* at small sampling intervals, which do not reduce the number of flows dramatically for TRWSYN to make a decision. However, small sampling intervals can substantially thin the flow size (in number of packets), thus increasing the likelihood of a multi-packet flow being reduced to a single packet flow. Recall that TRWSYN associates a *single SYN-packet* flow with a failed connection attempt, which is assumed more likely for scanners. The increase in single SYN-packet flows after sampling has a twofold impact. First, some ground truth scanners that may have transmitted multi-packet flows will be initially missed by TRWSYN applied to the original trace. After sampling, these multi-packet scanners would be thinned and are now more likely to be detected, which explains the slight increase in the success ratio of TRWSYN. Secondly, sampling causes a significant increase in false positives because flow thinning results in large number of single SYN-packet flows, that are now erroneously tagged as scanners.

When the sampling interval N increases, the decrease of R_s and R_{f+} is to be expected. Specifically, the number of flows observed from a source itself decreases for large values of N and becomes the dominant factor. Consequently, the algorithm makes fewer decisions and hence both R_s and R_{f+} decrease.

The effect of flow thinning and the reduction in the number of flows is empirically demonstrated by showing how sampling changes the flow characteristics of (a) the detected scanners and (b) the false positives (i.e., benign hosts erroneously tagged as scanners). Towards this end, we calculate two metrics for each SrcIP in these two groups: (1) flow size change ratio R_{fs} , defined as the average flow size (in number of packets) after sampling over the mean size in the original trace, and (2) number of flows change ratio R_{fn} , similarly defined to describe how the number of flows changes after sampling.

Figures 1 and 2 show R_{fs} and R_{fn} for each source in the *BB-West* trace with sampling intervals of $N = 10$ and $N = 100$, respectively. The top half of each graph marks the flow size change ratios R_{fs} , while the bottom half shows R_{fn} . We observe that $R_{fs} \approx 0.9$ for the detected scanners. This validates the hypothesis that scanners comprise mostly of

single packet flows whose flow size is largely unaffected by sampling. With a sampling interval of $N = 10$, there are still sufficient flows to make decisions, resulting in higher success ratio.

In the case of false positives, $R_{fs} \approx 0.3$ at $N = 1/10$ (as shown in Fig. 1(b)), indicates substantial thinning of flows even for low sampling intervals. In other words, multi-packet flows originating from these sources have a large likelihood of being reduced to single packet flows. At $N = 1/100$ (Figs. 2(a) and 2(b)) the number of flows is reduced significantly, which lowers the detection and false positive ratios. The following analysis shows that such flow thinning has a significant impact on the likelihood ratio computed by TRWSYN, causing it to erroneously tag the sources as scanners.

1) Impact of Sampling on Connection-Inference: TRWSYN assumes that a failed connection is more likely to be generated by a scanner, which is true in the original traffic, but may not hold for sampled traffic. We are interested in the amount of error introduced by packet sampling and how it affects the likelihood ratio under the different hypotheses in Eqn. (1). Note that the likelihood ratio for the alternate event, a multi-packet flow, is not adversely affected by sampling. It is still more likely for a multi-packet flow originated from a benign host rather than a scanner in sampled traffic.

We begin by defining some notations:

- 1) Let $\{f\}$ be the flow size distribution of a source in the *original* traffic; $f^{(n)}$ be the probability that the source generates a flow with n packets in the original trace.
- 2) Let $p = \frac{1}{N}$ be the probability that a packet is sampled.
- 3) TRWSYN associates a source with a hypothesis $H_i, i \in \{0, 1\}$ where $i = 0$ indicates a benign host and $i = 1$ a scanner. Consequently, we shall also index the flow size distribution $\{f\}$ with the hypotheses under consideration, i.e., the flow size distribution of a source under hypothesis H_i is labeled as $\{f_i\}$.

We first look at the specific case of TCP portscans. A TCP connection is associated with a failed connection if we observe a single SYN-packet. Let this event be denoted by $S-SYN$. The probability of observing this event for a benign host (i.e., under hypothesis H_0) when traffic is sampled with parameter

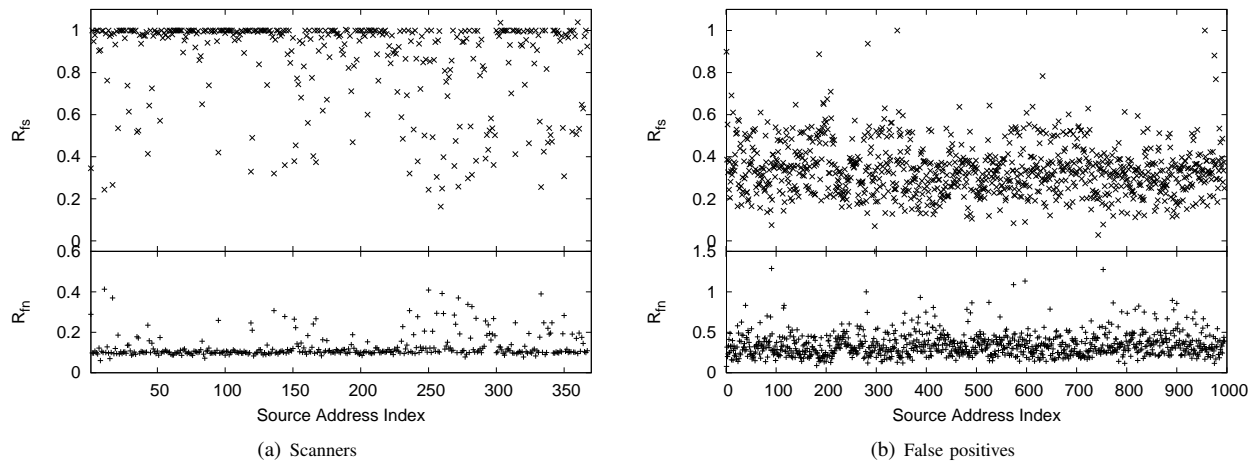


Fig. 1. The *BB-West* Trace: Flow size change ratio (R_{fs}) and number of flows change ratio (R_{fn}) after 1/10 sampling.

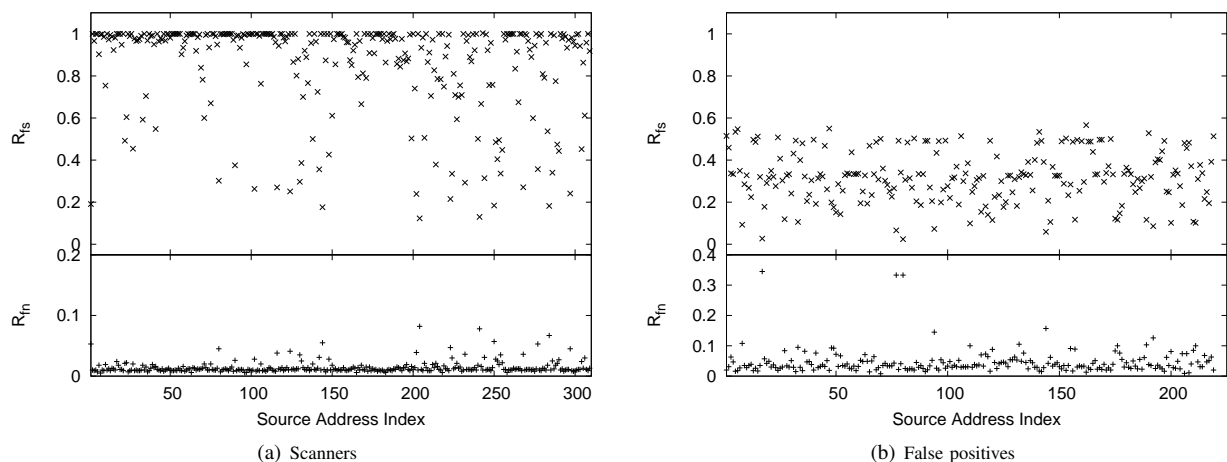


Fig. 2. The *BB-West* Trace: Flow size change ratio (R_{fs}) and number of flows change ratio (R_{fn}) after 1/100 sampling.

p is given by:

$$\Pr\{S\text{-}SYN|H_0\} = p \cdot f_0^{(1)} + p \sum_{n=2}^{\infty} f_0^{(n)} (1-p)^{n-1}. \quad (4)$$

The first term $f_0^{(1)}$ is the probability of a benign host generating a single-packet flow, which is quite small in practice (see Fig. 3). The summation on the right side of Eqn. (4) represents the probability of observing a single SYN-packet flow due to *thinning* of a multi-packet TCP flow. In other words, it represents the factor that induces false positives.

For small sampling probabilities, the term $(1-p)^n$ decays slowly. Hence, if the distribution $\{f_0\}$ has most of its mass in the small flow size region (i.e., small n), there would be relatively little distortion of the original distribution function $f^{(n)}$. To demonstrate this effect, we have plotted the original TCP flow size distribution $f^{(n)}$ of non-scanners (identified using the ground truth) as well as the distorted function $f^{(n)}(1-p)^{n-1}$, in Fig. 3 for the *BB-West* trace with $p = 0.1$ and 0.01. The distorted function $f^{(n)}(1-p)^{n-1}$ still captures 45% of the original distribution mass with $p = 0.1$, while with $p = 0.01$, it almost overlaps with the original flow size distribution. Consequently, the contribution of the summation

term in Eqn. (4) can be quite significant when the original flow size distribution mass is concentrated at small flow sizes.

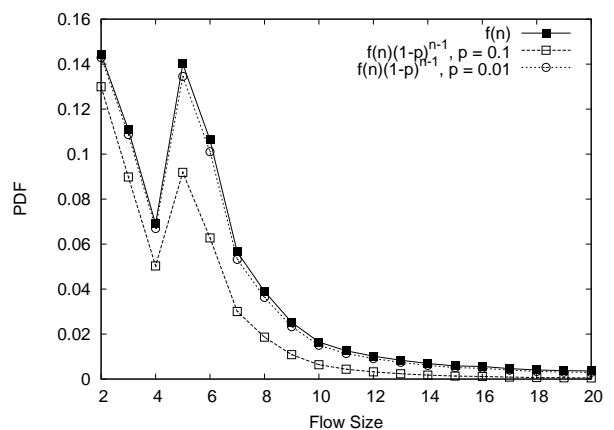


Fig. 3. The *BB-West* Trace: Flow size distribution for non-scanners

Under these conditions, we can approximate Eqn. (4) by

$$p(f_0^{(1)} + \sum_{n=2}^{\infty} f_0^{(n)} (1-p)^{n-1}) \approx p \sum_{n=1}^{\infty} f_0^{(n)} \approx p. \quad (5)$$

For a scanner, the flow size distribution is concentrated almost exclusively at $f_1^{(1)} \approx 1$. Hence the probability of S - SYN is given by:

$$\Pr\{S-SYN|H_1\} \approx p \cdot f_1^{(1)} \approx p. \quad (6)$$

Therefore, for small values of p , we have :

$$\Pr\{S-SYN|H_0\} \approx \Pr\{S-SYN|H_1\}. \quad (7)$$

The relation indicates that at low sampling rates and when benign hosts have relatively small flows, it is nearly impossible to distinguish between the likelihood of a single SYN-packet under the two different hypothesis. In other words, both scanner and benign hosts are almost equally likely to be associated with failed flows by TRWSYN under packet sampling. As an example, for the *BB-West* trace, we obtained the likelihood ratio Eqn. (1) in the event of a single SYN-packet to be:

$$\frac{\Pr\{S-SYN|H_1\}}{\Pr\{S-SYN|H_0\}} \approx \frac{0.99}{0.96} \approx 1.03. \quad (8)$$

The above analysis also sheds some light on the impact of the TRWSYN hypothesis test parameters θ_0 and θ_1 . If the parameters θ_0 and θ_1 were corrected for sampling, Eqn. (8) indicates we must have $\theta_0 = \theta_1$, in which case the TRWSYN hypotheses test would be inconclusive in detecting scanners. On the other hand, using the parameters for non-sampled traffic ($\theta_0 = 0.8, \theta_1 = 0.2$) will result in a large number of false positives since it artificially inflates the probability of observing a failed connection. This is precisely what we observed in our evaluation of TRWSYN.

The situation is exacerbated even further if we try to extend the (single-packet flow \rightarrow failed connection) analogy to protocols that do not keep state (this will be useful when we analyze the impact of flow thinning on TAPS). For a benign host we have:

$$\Pr\{S-PKT|H_0\} = f_0^{(1)}p + \sum_{n=2}^{\infty} f_0^{(n)}np(1-p)^{n-1}. \quad (9)$$

If we compare the likelihood ratios, we have:

$$\frac{\Pr\{S-PKT|H_1\}}{\Pr\{S-PKT|H_0\}} \approx \frac{f_1^{(1)}}{f_0^{(1)} + \sum_{n=2}^{\infty} f_0^{(n)}n(1-p)^{n-1}}. \quad (10)$$

For $n < 1/p$, the term $n \cdot (1-p)^{n-1}$ increases with n and hence amplifies the distribution for flow sizes less than $1/p = 100$. This implies that for small flow sizes, it is more likely that a sampled single packet flow is from a successful multi-packet flow than a failed connection attempt. To illustrate this, we compute the likelihood of observing a single-packet flow under the two different hypotheses for the *BB-West* trace, which yields:

$$\frac{\Pr\{S-PKT|H_1\}}{\Pr\{S-PKT|H_0\}} \approx \frac{2.05}{14.9}. \quad (11)$$

The above equation demonstrates the bias introduced by sampling towards the non-scanner hypothesis H_0 under the assumption of small flow sizes. In other words, a single packet flow is much more likely to be *thinned* from a multi-packet flow than a failed flow.

B. Impact of Sampling on TAPS

As with TRWSYN, we compared the performance of TAPS applied to traffic traces sampled at various rates. In each case, we also varied the size of the time bin over a range of values in order to understand how R_s and R_{f+} changes as a function of the time bin and sampling interval.

Figures 4(a) and 4(b) show the success and false positive ratios respectively of TAPS with different time bin sizes for the *BB-West* trace. Each curve represents R_s or R_{f+} for a particular sampling interval as a function of the time bin size. We first discuss the relation between the time bin size and sampling interval. Fig. 4(a) indicates that there is an optimal time bin at which R_s is the highest for each sampling interval. Furthermore, the value of the optimal time bin size is an increasing function of the sampling interval. This is not altogether surprising since for a fixed time bin, packet sampling *reduces* the number of observed events. Consequently, the size of the time bin must be increased as a function of the sampling interval to compensate for this aspect². This explains the rise in R_s when we initially increase the size of the time bins for each sampling interval. However, beyond a certain size, we run into edge-effects because we are using finite length traces, which reduces the success ratio.

Observe that R_{f+} also increases with time bin size. Although large time bin sizes compensate for sampling by increasing the number of observed events, flow thinning can severely distort the observed events, which increases the probability of an erroneous decision. We will evaluate the impact of sampling on R_s and R_{f+} next.

One can infer from Fig. 4(a) that the success ratio for TAPS monotonically decreases with the increasing sampling intervals (the curves for different N do not overlap or cross each other). TAPS has an extremely high success ratio for the original traffic, detecting almost all scanners. Hence, even moderate sampling causes only a reduction in its efficacy (unlike TRWSYN, whose success ratio increased initially). This behavior is also illustrated in Fig. 5 where we plot the R_s and R_{f+} curves for TAPS as a function of the sampling intervals. For each sampling interval we chose the time bin value that yields the highest success ratio. Observe that the R_s curve monotonically decreases as a function of the sampling interval.

The behavior of R_{f+} , shown in Fig. 4(b) is similar to TRWSYN. Specifically, low values of N result in high false positive ratio. To see this, note that in Fig. 4(b) the R_{f+} curves for small values of N lie above the R_{f+} curve of the original traffic. This can also be seen from Fig. 5 where the R_{f+} curve for TAPS initially increases with sampling. At large sampling intervals, the false positive ratio decreases. The reasoning for the behavior of R_{f+} is also similar to that for TRWSYN since TAPS also associates single packet flows with failed connections. Specifically, at low sampling intervals, *flow thinning* rather than reduction in the volume of flows is a dominant factor, resulting in increased false positives. At larger sampling intervals, the reduction in flow volume

²A quantitative relationship between the time bin size and sampling interval is presented in [24].

becomes dominant. This results in fewer observed events and hence fewer false positives.

Figure 5 compares the performance of TRWSYN and TAPS under sampling. We note that TRWSYN yields a far higher R_s compared to TAPS on *sampled* traffic. However, in terms of R_{f+} , which is often more critical, we see that the reverse is true. Specifically, although flow thinning adversely affects both TRWSYN and TAPS, a comparison of these two techniques in Fig. 5 indicates that TAPS results in far *fewer* false positives than TRWSYN. We shall delve further into the comparison between TRWSYN and TAPS under sampling in Section III-D. Before that, we analyze the impact of sampling on TAPS and show that the additional metrics related to the access pattern it uses is robust to sampling, which can explain the low rate of false positives.

1) *Impact of Sampling on Source Access Pattern:* As in Section III-A.1, we analyze the impact of flow thinning on the “event” that is used to differentiate a scanner from a benign host in the original traffic and show how this bias changes with sampling.

In case of TAPS, the “event” refers to a source transmitting at least k *single packet* flows to *distinct* addresses or ports. A simplified expression for the likelihood of this event, which we denote as E , can be derived as follows. For the purpose of analytical tractability we assume that the threshold (see Section II) is set to $k = 2$, i.e., at least two distinct *failed* connections are required to favor the scanner hypothesis. In a given time bin, let g_M denote the probability that a source s generates M connections. Let $P(p, H_i)$ denote the probability under the hypothesis H_i and the sampling probability p , that exactly one packet is sampled from any of the M flows³. Then the probability that the source sampled only one packet from exactly K flows, given that M flows were generated in a time bin, can be approximated by the binomial distribution with parameters $(M, P(p, H_i))$.

We assume that the source picks addresses (or ports) for each of the M connections uniformly from a set $\{\mathcal{N}\}$ with probability $1/N$, where $N = \|\mathcal{N}\|$. Under these assumptions, if K single packet flows were observed in the time bin, the probability that at least two of the K flows have distinct addresses (or ports) is simply $1 - (1/N)^{K-1}$.

Combining the above arguments, we obtain the probability of the event E under hypothesis H_i to be:

$$\Pr\{E|H_i\} = \sum_{M=K}^{\infty} g_M \left(1 - \left(\frac{1}{N_i}\right)^{K-1}\right) \binom{M}{K} \cdot P_s(p, H_i)^K (1 - P_s(p, H_i))^{M-K}.$$

After some re-arrangement this can be written as

$$\Pr\{E|H_i\} = \left(1 - \left(\frac{1}{N_i}\right)^{K-1}\right) P_s(p, H_i)^K \cdot \Gamma_K(p, H_i), \quad (12)$$

where

$$\Gamma_K(p, H_i) = \sum_{M=K}^{\infty} \binom{M}{K} g_M (1 - P_s(p, H_i))^{M-K}. \quad (13)$$

³Strictly speaking $P(p, H_i)$ is also a function of the flow sizes and sampling probability, but we ignore this dependency for tractability.

Plugging Eqn. (12) in Eqn. (1), the ratio of the likelihood of event E under two different hypotheses, is given by:

$$\frac{\left(1 - \left(\frac{1}{N_1}\right)^{K-1}\right) \cdot P_s(p, H_1)^K \cdot \Gamma_K(p, H_1)}{\left(1 - \left(\frac{1}{N_0}\right)^{K-1}\right) \cdot P_s(p, H_0)^K \cdot \Gamma_K(p, H_0)}. \quad (14)$$

Observe that the first fraction depends only on N_i and is invariant to sampling, while the second and third fractions do depend on the sampling probability p . In the case of TAPS, which makes its decision based on single packet flows, we have from Eqn. (9) that

$$P_s(p, H_i) = p \cdot \sum_{n=1}^{\infty} f_i^{(n)} n (1-p)^{n-1}.$$

For reasonably small values of p , $P_s(p, H_i)$ is quite small for both hypotheses and hence from Eqn. (13) we have $\Gamma_K(p, H_1) \approx \Gamma_K(p, H_0)$. However, as observed from Eqn. (11) in the previous sub-section, due to flow thinning we can have the ratio $(P_s(p, H_1))^K / (P_s(p, H_0))^K \ll 1$ for small p . This relation tells us that one is more likely to observe K single-packet flows in sampled traffic from a benign host that originally had multi-packet flows than from a scanner. This is the factor that contributes to the increase in the false positive ratio.

On the other hand, the first metric, which captures the communication patterns has the property that $N_1 \gg N_0$, i.e., scanners access a far larger set of addresses (or ports) than benign hosts [10]. In other words, observation of k *distinct* identifiers is more likely to be associated with a scanner. Hence we have

$$1 - \left(\frac{1}{N_1}\right)^{K-1} \gg 1 - \left(\frac{1}{N_0}\right)^{K-1}.$$

Therefore, by also requiring this condition to identify a scanner, TAPS counter-balances the negative impact of flow thinning.

The impact of both factors, flow thinning and the address range metric, are evident in Fig 5. Flow thinning due to packet sampling increases the false positive ratio of TAPS. However, compared to TRWSYN, which relies exclusively on single-packet flows, the incorporation of an additional metric in TAPS based on the address range distribution reduces the impact of flow thinning, resulting in far lower false positives than TRWSYN. We use the insight from the above analysis to propose a simple modification to TAPS for TCP portscan detection (described in the next sub-section), which reduces false positive ratios even further in a packet sampling environment.

C. Making Portscan Detection Robust to Sampling

Our analysis of the impact of sampling on TRWSYN and TAPS can be summarized into two key observations :

- 1) Packet sampling induces *Flow Thinning* which impacts the inference related to single packet flows in two ways. For TCP flows, the event that a single SYN-packet came from a failed connection is nearly indistinguishable statistically from the case where a multi-packet flow got

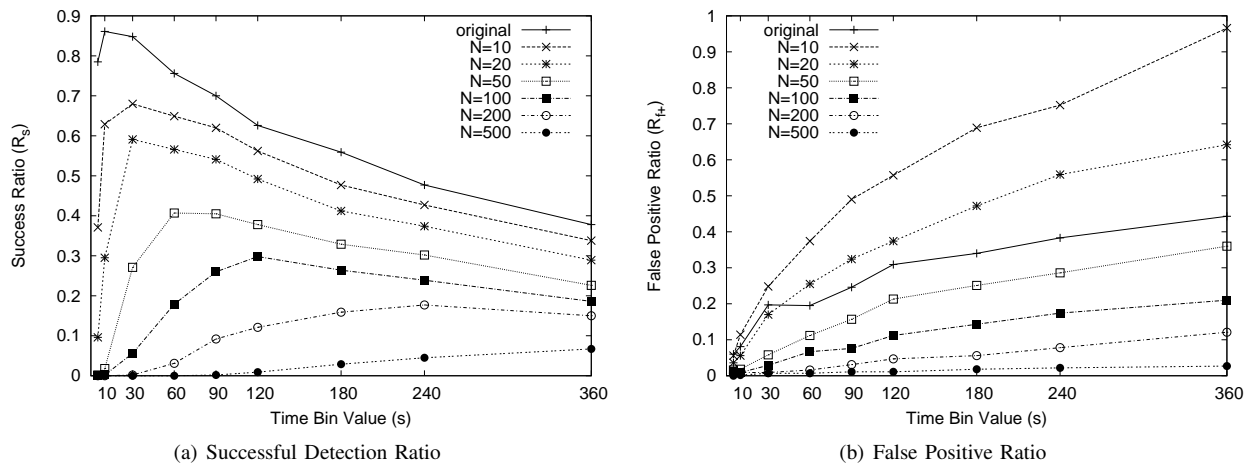


Fig. 4. The *BB-West* trace: Ratio of successful detection and false positive under various sampling rate and time bin size for TAPS.

thinned, as shown in Eqn. (7). However, for non-TCP flows, Eqn. (11) indicates that a single-packet flow in the sampled traffic is much more likely to be associated to a multi-packet flow than a failed connection attempt.

- 2) The *address range* distribution metric used by the TAPS algorithm is invariant (in terms of bias) to sampling as shown by the first component of Eqn. (14). Hence, it is a desirable indicator to use in a sampling environment.

Motivated by these two observations, we propose a simple modification to the TAPS algorithm to detect *TCP portscans*, called TAPS-SYN. As the name suggests, it is essentially a simple modification of TAPS, with the aim of further reducing false positives under packet sampling.

TAPS-SYN works as follows. Similar to TAPS, it performs hypotheses testing on events observed in a time bin. Furthermore, it sets its hypotheses parameters θ_0, θ_1 such that an event comprising at least k failed connections to *distinct* addresses (or ports) is more likely to originate from a scanner. However, to identify failed connections attempts, it borrows from TRWYSN that only single SYN-packet flows are associated with failed TCP connections.

We now explain the rationale for the above technique. As observed previously, with sampling, the likelihood of a single SYN-packet coming from a failed TCP flow is almost the same as from a thinned multi-packet flow, i.e., $P(p, H_1) \approx P(p, H_0)$. Plugging this relation in Eqn. (14) yields:

$$P(p, H_1)^K \Gamma_K(p, H_1) \approx P(p, H_0)^K \Gamma_K(p, H_0).$$

Consequently, the likelihood ratio in Eqn. (14) is dominated by only the address distribution metric. Since this metric is unaffected by sampling, it correctly reflects the association of k failed TCP flows to distinct destination addresses (or ports) with a scanner. In comparison, TAPS associates any single-packet flow with a failed connection, which increases the negative impact of flow thinning, as observed from Eqn. (11). Based on these arguments, we expect TAPS-SYN to yield lower false positives in a sampled environment.

This was indeed found to be true by comparing the performance of this algorithm with TRWSYN and TAPS applied to the *BB-West* trace. The results for all three algorithms are

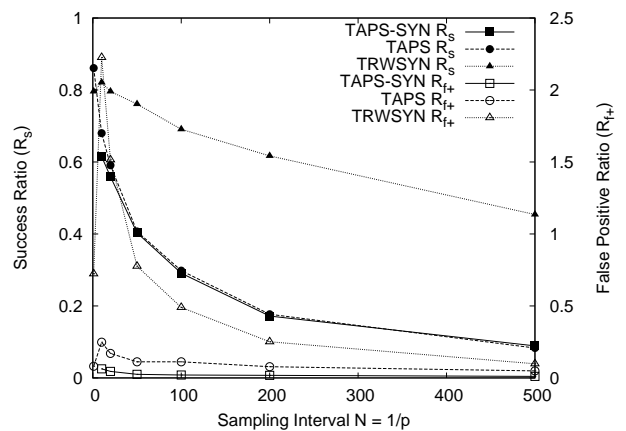


Fig. 5. The *BB-West* Trace: comparison of TRW, TAPS and TAPS-SYN

shown in Fig. 5. It is apparent that TAPS-SYN has almost the same detection rates as TAPS but extremely low false positive rates, making it eminently feasible to use with sampled traffic. Similar results were observed for the *Wireless* trace [24] (which are omitted due to space limitations). We however note that this algorithm is limited to detecting only TCP portscans as opposed to TAPS which is much more broadly applicable. Our current work involves extending these observations to develop detection algorithms for non-TCP portscans that are robust to sampling.

D. Evaluation with Artificial Scanners

Finally, we evaluated TRWSYN and TAPS in a controlled environment where artificial scanners were injected into the *BB-West* trace. Note that for this experiment, the ground truth was the set of artificially injected scanners, which are precisely known. This allowed us to verify inferences made with the approximate ground truth derived from the original trace.

The R_s curves of both algorithms plotted in Fig. 6 for the artificial scanners show similar behavior as that for the approximate ground truth in the sampled *BB-West* traces (Fig. 5). In particular both the R_s curves drop along with the sampling rate, with TRWSYN maintaining a higher detection

ratio than TAPS for sampled traffic. Furthermore, the R_s (with respect to artificial scanners) for TRWSYN declines at a much slower rate than TAPS as a function of the sampling interval, again similar to the behavior observed previously with the approximate ground truth (Fig. 5). This is because TRWSYN makes decisions based on each *individual* connection and hence requires only a few connections to classify a source. TAPS, on the other hand requires a certain minimum number of connections in each *time bin* to arrive at a decision, i.e. many more connections than TRWSYN. Hence, R_s for TAPS is more sensitive to sampling.

For completeness, we also show the R_{f+} curves for both algorithms. Due to the fact that we did not inject any benign traffic, the number of false positives did not change from those observed in the *BB-West* trace. Specifically, both Figs. 5 and 6 show that TAPS is far superior to TRWSYN from the perspective of false positives, maintaining a low rate of erroneous decisions in sampled traffic. The analysis in Section III-B.1 demonstrated that this is because the address pattern metric used by TAPS is invariant under sampling.

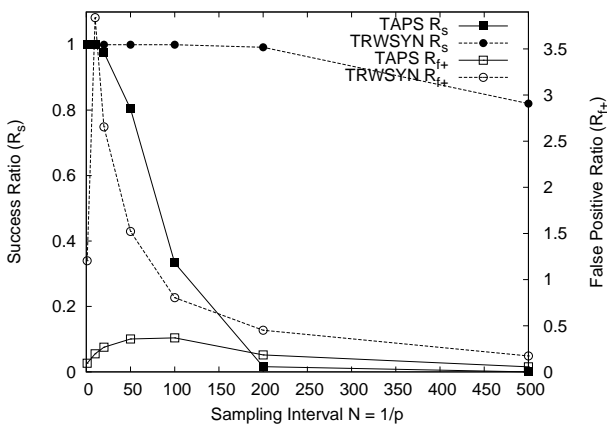


Fig. 6. The *BB-West* Trace injected with artificial scanners: Performance comparison between TRWSYN and TAPS

IV. IMPACT OF SAMPLING ON ENTROPY-BASED PORTSCAN DETECTION

A few recent studies [11], [12] have proposed to use entropy as a summarization tool to describe traffic and detect a wide range of abnormal behaviors, including portscans. In this section, we present a study to assess the impact of packet sampling on the entropy-based traffic profiling technique applied to portscan detection. The results show that the reliability of using entropy as an indicator for anomaly decreases significantly under packet sampling.

Following the methods in [11], [17], and the process described in Section II-B, we analyze traffic flows across four feature space dimensions: *SrcIP*, *SrcPORT*, *DstIP*, and *DstPORT*. A *feature value* therefore refers to either an IP address or port number. As an intermediate step, we demonstrate the effects of sampling on entropy and relative uncertainty (RU) calculation, as well as significant cluster (SC) extraction, for each five-minute time bin, using the same parameter settings in [11]. Due to page limits, we only

show the results from the 1st hour of the *BB-West* trace. We also present the analysis of post-sampling entropy, which gives insights to the portscan detection results. Finally, we evaluate the performance of portscan detection using entropy, by comparing the sources that fall into scanner behavior classes against the ground truth scanner set.

A. Impact of Sampling on Entropy and RUs

Figure 7 plots the time series of the entropy values before and after sampling, respectively. We observe that both the entropy values and RU of the sampled trace are higher than those of the original trace, with the RU in particular monotonically increasing with the sampling interval N . For instance, at 1/500 sampling, $RU(SrcIP)$ averages at 0.96, a 20% rise from the original value 0.8. We will show shortly that the scaling-up of the RUs has a huge impact on the significant cluster extraction.

We now discuss the reason behind the change in entropy values under packet sampling. Recall that packet sampling has an inherent bias towards big flows, i.e., flows with big sizes are more likely to be sampled than those of small sizes. Entropy, which summarizes the distributions of the number of flows across feature dimensions, will differ after packet sampling distorts the distributions. Let p be the probability that a packet is sampled, where $0 < p < 1$. Then a flow of n packets is sampled with probability:

$$p_n = 1 - (1 - p)^n \approx \begin{cases} p \cdot n & \text{if } n \ll \frac{1}{p} \\ 1 & \text{otherwise.} \end{cases}$$

Thus for any flow with size larger than $1/p$ packets, e.g. 10 packets for $p = 0.1$, at least one packet will be sampled and the flow recorded with a very high probability. A smaller flow, however, may only be sampled with a probability proportional to its size. In the extreme case of the single packet flows, each flow is randomly sampled at rate p . To evaluate the the impact of sampling on entropy, let $f(m)$ represent the probability distribution of source m ($1 \leq m \leq M$) along the *SrcIP* dimension, with average flow size of s_m . If there are \mathcal{N} flows originally, the number of flows after sampling becomes:

$$\mathcal{N}' \approx \mathcal{N} \cdot \left(\sum_{s_m < \frac{1}{p}} f(m)ps_m + \sum_{s_m \geq \frac{1}{p}} f(m) \right) = \mathcal{N} \cdot \rho < \mathcal{N},$$

where

$$\rho = \frac{\mathcal{N}'}{\mathcal{N}} \approx \sum_{s_m < \frac{1}{p}} f(m)ps_m + \sum_{s_m \geq \frac{1}{p}} f(m) < 1 \quad (15)$$

is the sampling factor. From Eqn. (2) the original *SrcIP* entropy $H_m = -\sum f(m) \log_2 f(m)$, one can obtain the entropy after sampling as:

$$H'_m \approx - \sum_{s_m < \frac{1}{p}} \frac{f(m)ps_m}{\rho} \log_2 \frac{f(m)ps_m}{\rho} - \sum_{s_m \geq \frac{1}{p}} \frac{f(m)}{\rho} \log_2 \frac{f(m)}{\rho}. \quad (16)$$

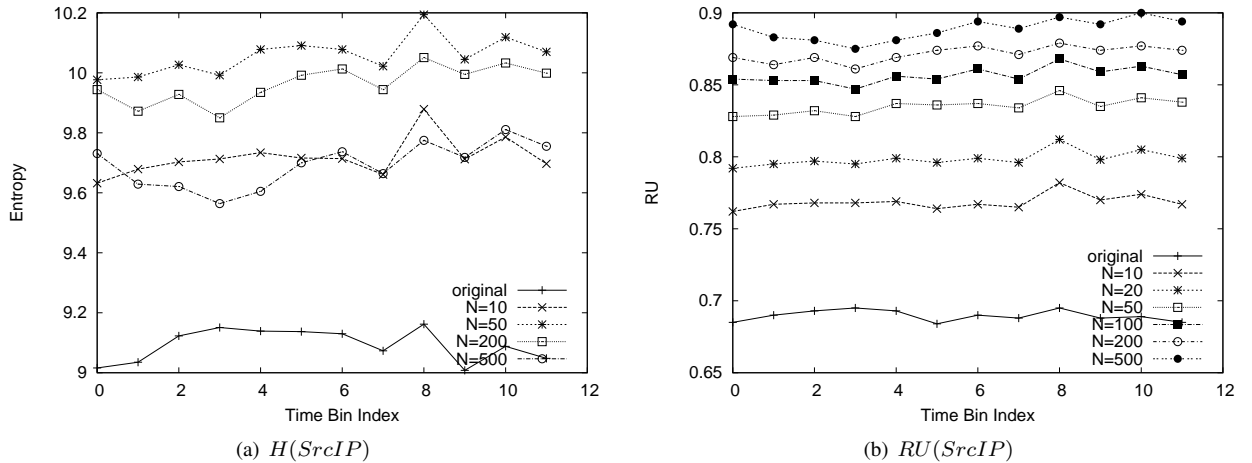


Fig. 7. Time series of *SrcIP* entropy and RU for the *BB-West* Trace (1st hour)

There are two approximations made to estimate the entropy at different sampling rates in Eqn. (16). First, we use the average flow size s_m to simplify the flow size distribution of the source m . Second, we replace the flow sampling probability with $p \cdot s_m$ for any $s_m < 1/p$. Nonetheless, after plugging in the empirical flow count distribution $f(m)$, the estimation results match closely to those of the sampled traces. Fig. 8 illustrates the changes of sampling factor ρ , entropy $H(\text{SrcIP})$, and $RU(\text{SrcIP})$ at different sampling intervals $N = 1/p$ for time bin 6 in the *BB-West* trace.

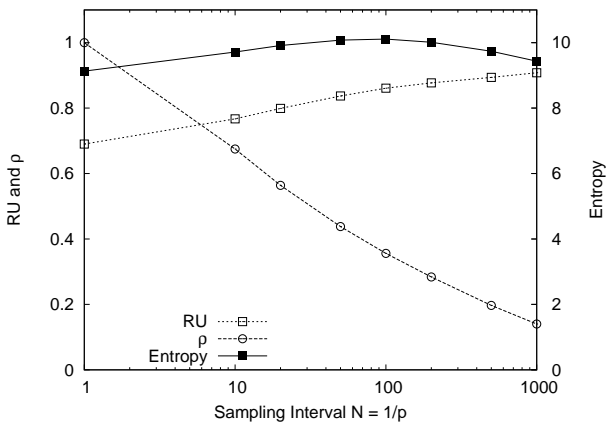


Fig. 8. Sampling effect on *SrcIP* entropy and RU of the 1st hour of the *BB-West* Trace (time bin 6)

Since most flows are short and only a relatively small number of flows have large sizes, the entropy of this non-uniform distribution is smaller than 1. The number of flows sampled is not proportional to the packets sampling rate, because large-sized flows are less likely to be missed by packet sampling compared to small flows. The first item on the right of Eqn. (15) shows the portion of flows with relative small sizes that will be heavily affected by sampling. Initially this portion contributes to a large percentage, however it gets reduced as the sampling interval increases. As a result, the sum distribution becomes more uniform. As the sampling interval increases even further, the distribution is strongly biased by

the few large flows (i.e., the second item on the right of Eqn (15)), which reduces entropy. That explains why the entropy value goes up then drops in Fig. 8. The RU curve increases monotonically due to the faster decline of source M sampled compared to the drop of entropy.

B. Impact of Sampling on Portscan BCs

We now focus our study on the impact of sampling on profiling scanners based on the entropy distribution [11]. Recall from Section II, it takes two steps to classify a source as a scanner. First, significant clusters (SCs) of sources in each time bin are extracted, according to a threshold β . Second, each *SrcIP* cluster is categorized into different behavior classes (BCs). Sources correspond to $[*, 2, 0]$ or $[*, 0, 2]$ (referred to as scanner BCs) are profiled as scanners that probes a large number of random ports or random IP addresses. To ensure the portscan detection result is comparable to the ground truth in Section III, we filter out any web traffic on port 80 and other non-TCP flows. We mark those sources belonging to either of these two BCs (in each time bin) as scanners and combine the results from all 12 time bins into a scanner list.

Table IV lists the number of scanners found in the SCs of scanner BCs, before and after sampling. We observe that the total number of detections declines with sampling. With sampling rate $\leq 1/50$ and default β value of 0.9, we detect no scanners at all. This is mainly due to the diminishing number of the SCs. After sampling, the majority of the original scanners are below the “clustering” threshold, resulting in fewer SCs. This is also shown in the increase of RU values with sampling in Fig. 7(b). Recall that RU close to 1 indicates that the distribution of flows generated by each IP is close to uniform rather than cluster-like. Since the number of flows generated by the scanners are no longer significant compared to other non-scanner traffic, there are very few SCs. When the remaining SCs are classified, some have changed from scanner BC to a non-scanner BC, simply because $RU(\text{DstIP})$ or $RU(\text{DstPORT})$ are also escalated by sampling. This further reduces the number of scanner BCs after sampling.

Since the primary factor that affects portscan detection is the low number of SCs extracted when the default value of the

TABLE IV
NUMBER OF SCANNER BCs DETECTED BY THE ENTROPY-BASED METHOD WITH $\beta = 0.9$ (GROUND TRUTH = 447)

Detections	Original	$N = 10$	$N = 20$	$N = 50$	$N = 100$	$N = 200$	$N = 500$
Total	443	166	35	0	0	0	0
Success (R_s)	384 (85.9%)	155 (34.7%)	31 (6.9%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
False + (R_{f+})	59 (13.2%)	11 (2.5%)	4 (0.9%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)

clustering threshold β is used, we explore how performance changes with higher β values. We show the changes in the number of SCs for time bin 6 in Table V. For each sampling rate, β is raised from 0.9 in steps of 0.01, until the number of SCs found exceeds that of the original trace (61). We notice a side effect: the number of false positives (shown inside the parenthesis) increases with the β value. This further proves that packet sampling distorts distribution of the *SrcIP* flow counts and makes it more uniform.

TABLE V
NUMBER OF *SrcIP* SCs IN TIME BIN 6 FOR INCREASING THRESHOLD β VALUES

β	Original	$N = 10$	$N = 20$	$N = 50$	$N = 100$
0.90	61	8	0	0	0
0.91	-	19(1)	4	0	0
0.92	-	41(6)	13(1)	0	0
0.93	-	81(30)	33(7)	4	0
0.94	-	-	76(30)	15(3)	2
0.95	-	-	-	44(15)	13(2)
0.96	-	-	-	113(71)	38(15)
0.97	-	-	-	-	116(75)

Figure 9 shows that as β increases, the successful detection ratio increases, but so does the false positive ratio. For instance, if we set $\beta = 0.99$, $R_s = 0.94$, while $R_{f+} = 2.24$ at sampling rate of $p = 1/10$. At sampling rate of $1/500$, the R_{f+} is close to R_s , which makes the detection results very undesirable. The high false positive ratio is caused by the increasing number of false SCs extracted when raising the β value, i.e., benign BC types are altered to become scanner BCs, especially at lower sampling rates.

In summary, we have shown in this section that packet sampling affects the entropy-based portscan detection, and the impact of sampling comes from two main sources:

- Abstraction using entropy condenses flow distribution into a single value. It summarizes the uniformity of the number of flows per address or port, and ignores the flow size information.
- Packet sampling is biased towards large sized flows, whose significance gradually grows as the sampling rates decrease, resulting in a more uniformed, as opposed to heavy tailed, distribution of flow counts across different flow sizes.

V. PERFORMANCE COMPARISON UNDER SAMPLING

In this section, we compare the performance of TRWSYN, TAPS, and the entropy-based portscan detection methods under packet sampling. Recall that high successful detection (R_s) and low false positive ratios (R_{f+}) are desirable properties for a portscan detection algorithm. However, as shown in this

study, all three algorithms experience a trade-off between R_s and R_{f+} . For example, in Section III-D, it was shown that TRWSYN has high success ratios but suffers from high false positive ratios, while TAPS has the opposite characteristics. Section IV shows that increasing the value of β improves the success ratio of the entropy-based algorithm, but at the price of increased false positives.

Therefore, for comparison purposes, one must utilize a metric that accounts for *both* the success ratio (R_s) and false positive ratio (R_{f+}). While there are several ways of doing so, we propose a simple metric of the form $R(\alpha) = \alpha R_{f-} + (1 - \alpha) R_{f+}$, where $0 \leq \alpha \leq 1$. Note that the false negative ratio $R_{f-} = 1 - R_s$. A low value of R implies low false negative and false positive ratios which, as indicated before is the desired goal. By specifying α , one can decide the importance of either ratio.

In the previous sections, we independently evaluated the special cases $\alpha = 1$ which corresponds to $R(1) = R_s$ and $\alpha = 0$ ($R(0) = R_{f+}$) for each algorithm. We now compare their performance directly with equal weight given to both R_s and R_{f+} , i.e., $\alpha = 0.5$. Fig. 10 plots $R(0.5)$ for TRWSYN, TAPS, and the entropy-based method at various sampling rates. For each sampling rate, we picked the optimal β value for the entropy-based portscan detection, and the optimal time bin size for TAPS such that R is minimized.

An interesting observation from the figure is the near-identical performance from TAPS and the entropy-based portscan detection. This is not altogether surprising since both TAPS and the entropy-based detection utilize the access pattern of scanners, i.e., spread of destination IP addresses or port numbers. Indeed, in Section IV, it was shown that a choice of β such that R_s of the entropy-based technique matches that of TAPS also results in similar R_{f+} for both. However, while a simple relation between the time bin size and sampling rate exists [24] for TAPS, identification of the optimal β for the entropy-based portscan detection is not straightforward. This implies that *a priori* configuration of the time bin for TAPS is feasible to optimize its detection performance, while such tuning is difficult for the entropy-based schemes.

For sampling rates lower than $1/100$, Fig. 10 indicates that both TAPS and the entropy-based portscan detection outperform TRWSYN. This is primarily due to the large volume of false positives, which results in a high R for TRWSYN. However, at higher sampling rates, TRWSYN performs best because it still maintains a high success ratio while those of the other two algorithms drops rapidly.

VI. CONCLUSIONS

Packet sampling is commonly deployed in high-speed backbone networks to reduce the measurement overhead. Not

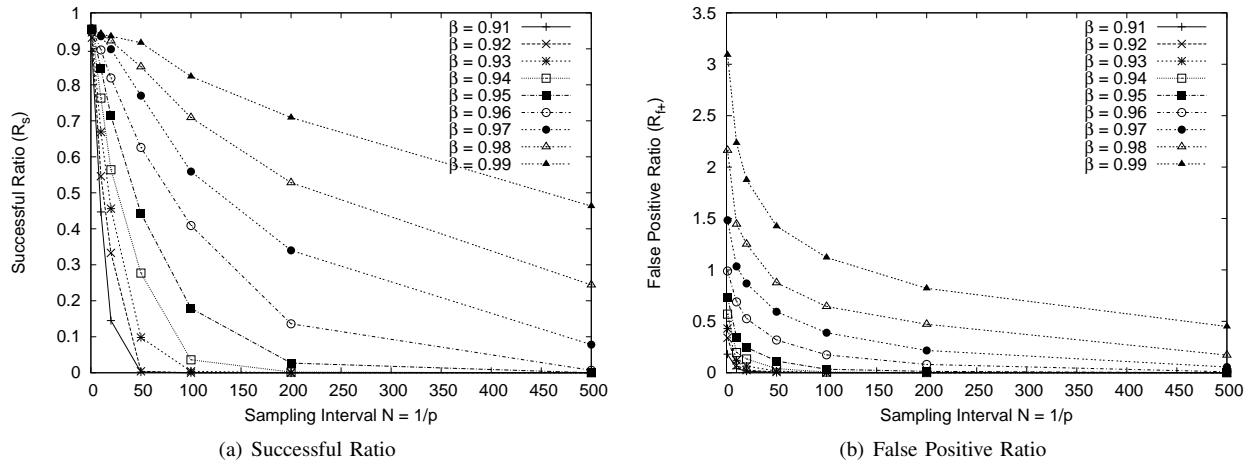


Fig. 9. The *BB-West* Trace: Portscan detection results R_s and R_{f+} with increasing threshold β

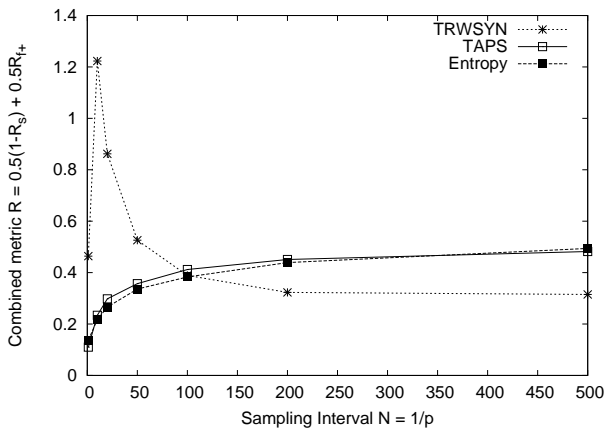


Fig. 10. Comparison of performance based on the combined metric of equally weighted R_s and R_{f+}

only is the measurement data used for accounting and billing purposes, it is also important to security-related tasks such as anomaly detection. There is a common belief that detection methods based on volume information are by nature sensitive to packet sampling. This paper investigates how packet sampling also affects the detection of non-volume-based anomalies like portscans. Portscan detection is a crucial component of security defense systems adopted by network operators, since portscans typically precede virus/worm propagation.

Using real packet traces sampled at different rates, we compared the performance of three specific algorithms: (a) TRW for detecting TCP scanners based on connection-based inference, (b) TAPS, which performs statistical hypotheses testing based on scanner access patterns of destination addresses/ports, and (c) an entropy-based profiling method applied to portscan detection. Our results demonstrated that sampling distorts various traffic features and degrades the performance of all three algorithms in terms of success detection ratio and false positives. Specifically, portscan detection based on connection inference (e.g., detecting single TCP SYN packet in TRW) are effective only if the flow size distributions for scanners and benign hosts are different enough to overcome

the sampling distortion. We showed that the thinning effect of sampling introduces significant errors for practical sampling rates lower than $1/100$. On the other hand, techniques such as TAPS that exploits access pattern of scanners, e.g., spread of destination IP addresses or ports, work well even if flow size distributions of scanners and non-scanners are similar, and hence is more robust to sampling. Based on this insight, we designed a hybrid algorithm TAPS-SYN that achieves lower false positives while maintaining a reasonably high success rate under packet sampling.

The authors of [11], [17] proposed a portscan detection algorithm based on profiling a source using entropy of the destination IP addresses and ports. We applied this method on the original trace and the sampled traces. Results showed that as the sampling rate decreases: (a) both the success and false positive ratios drop, and (b) the performance is equivalent to that of TAPS since both algorithms identify scanners based on access patterns. Through analysis, we demonstrated that packet sampling can introduce fundamental bias by changing *distributions of traffic features* such as flow size. This has serious implications on traffic profiling algorithms such as [11] that rely on these traffic features in identifying significant clusters.

We believe that the lessons learned in this paper can be leveraged to address the accuracy, efficiency, and scalability trade-offs in designing better sampling techniques. Another open challenge is to design anomaly detection algorithms that are effective at relatively low sampling rate to avoid the need for detailed packet trace collection.

REFERENCES

- [1] “Cisco IOS Software NetFlow,” <http://www.cisco.com/warp/public/732/Tech/nmp/netflow/>.
- [2] “Juniper Networks: JUNOS 7.2 Software Documentation,” <http://www.juniper.net/techpubs/software/junos/junos72/index.html>.
- [3] V. Yegneswaran, P. Barford, and J. Ullrich, “Internet Intrusions: Global Characteristics and Prevalence,” in *Proc. of ACM Sigmetrics 2003*, San Diego, CA, USA, June 2003.
- [4] N. Duffield, C. Lund, and M. Thorup, “Properties and Prediction of Flow Statistics from Sampled Packet Streams,” in *Proc. ACM SIGCOMM IMW’02*, Marseille, France, Nov. 2002.

- [5] N. Hohn and D. Veitch, "Inverting Sampled Traffic," in *Proc. ACM SIGCOMM IMC'03*, Miami Beach, Florida, USA, Oct. 2003.
- [6] N. G. Duffield, C. Lund, and M. Thorup, "Estimating Flow Distributions from Sampled Flow Statistics," in *Proc. ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.
- [7] N. Duffield, "Sampling for Passive Internet Measurement: A Review," *Statistical Science*, vol. 19, no. 3, pp. 472–498, 2004.
- [8] "Snort," <http://www.snort.org>.
- [9] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," in *Proc. of 2004 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2004.
- [10] A. Sridharan, T. Ye, and S. Bhattacharyya, "Connection Port Scan Detection on the Backbone," in *Malware Workshop held in conjunction with IPCC*, Phoenix, Arizona, USA, April 2006.
- [11] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Profiling Internet Backbone Traffic: Behavior Models and Applications," in *Proc. ACM SIGCOMM '05*, Philadelphia PA, USA, Aug. 2005.
- [12] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," in *Proc. ACM SIGCOMM '05*, Philadelphia, PA, USA, Aug. 2005.
- [13] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," in *Proc. of SIGCOMM'04*, Portland, Oregon, USA, Aug. 2004.
- [14] B.-Y. Choi, J. Park, and Z.-L. Zhang, "Adaptive Random Sampling for Traffic Load Measurement," in *Proc. IEEE International Conference on Communications (ICC'03)*, Anchorage, Alaska, USA, May 2003.
- [15] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Trans. on Computer Systems*, vol. 21, no. 3, pp. 270–313, 2003.
- [16] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *J. of Computer Security*, vol. 10, no. 1-2, pp. 105–136, 2002.
- [17] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Reducing Unwanted Traffic in a Backbone Network," in *Proc. of SRUTI'05*, Cambridge, MA, USA, July. 2005.
- [18] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the sprint IP backbone," *IEEE Network*, 2003.
- [19] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics," RFC 2330, May 1998.
- [20] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection," draft-ietf-psamp-sample-tech-06.txt, Feb. 2005.
- [21] P. Barford and D. Plonka, "Characteristics of Network Traffic Flow Anomalies," in *Proc. ACM SIGCOMM IMW'01*, San Francisco, CA, USA, Nov. 2001.
- [22] "Network Mapper," <http://www.insecure.org/nmap>.
- [23] T. Ye, D. Veitch, G. Iannaccone, and S. Bhattacharyya, "Divide and Conquer: PC-based Packet Trace Replay at OC-48 Speeds," in *Tridentcom*, February 2005.
- [24] J. Mai, A. Sridharan, C.-N. Chuah, T. Ye, and H. Zang, "Impact of Packet Sampling on Portscan Detection," Sprint ATL, Tech. Rep. RR06-ATL-043166, 2006.



Jianning Mai (S'03) received his B.Eng. from Tsinghua University, Beijing, China in 1995, and his M.Eng. from National University of Singapore, Singapore in 1998.

He is currently a Ph.D. candidate at Department of Electrical and Computer Engineering, University of California, Davis. Before joining UCD in 2003, he worked as networking protocol software developer at IBM Singapore, and several start-ups in Silicon Valley. His research interests are in Internet measurements and network anomaly detection.



Ashwin Sridharan (S'99, M'04) has been a research scientist with Sprint Advanced Technology Labs since 2004. He obtained his Masters from the Indian Institute Of Science, Bangalore in 1999 and his Ph.D from the University of Pennsylvania in 2004 where he worked on traffic engineering and routing algorithms.

His current research interests are in the area of traffic measurement, performance optimization and hardening of IP and wireless data networks. He has participated as a program committee member as well

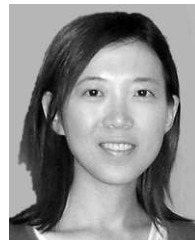
as a reviewer in various networking conferences.



Chen-Nee Chuah (S'92, M'01) received her B.S. in Electrical Engineering from Rutgers University in 1995, and her M. S. and Ph.D. in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 1997 and 2001, respectively.

Chuah is currently an Associate Professor in the Electrical and Computer Engineering Department at the University of California, Davis (UCD). Before joining UCD, she held a visiting researcher position at Sprint Advanced Technology Laboratories. Her

research interests are in computer networks and distributed systems, Internet measurements, P2P systems, network security, and wireless/mobile networking. Chuah received the National Science Foundation CAREER Award in 2003 and the UC Davis College of Engineering Outstanding Junior Faculty Award in 2004. She has been the Technical Program Co-Chair for the ACM Mobicom 2004 Workshop on Vehicular Ad Hoc Networks (VANET), and the Vice-Chair for IEEE Globecom 2006. She has also served on the technical program committee of several ACM- and IEEE-sponsored conferences.



Hui Zang (S'97, M'02, SM'04) received the B.S. degree in computer science from Tsinghua University, Beijing, China in 1997, and the M.S. and Ph.D. degrees in computer science from the University of California, Davis in 1998 and 2001, respectively.

In 2000, she joined Sprint Advanced Technology Laboratories, Burlingame, CA, where she is a research scientist. She was one of the guest editors of IEEE Network special issue on "Traffic Engineering in Optical Networks." She is the author of the book

"WDM Mesh Networks – Management and Survivability" (Kluwer Academic, 2002). She has published over 30 conference papers and journal articles and currently has one US patent granted and three pending in the field of networking and communications. Her research interests include performance and security issues in wireless, IP and optical networks. She serves or has served as a Technical Committee member of a number of conferences. She also helped organizing OptiComm'02 as a panel co-chair and Broadnets'04 as a publication co-chair.



Tao Ye has been a Research Scientist at Sprint Advanced Technology Labs since 2003. She received her Master of Science degree in Computer Science from UC Berkeley, and a dual Bachelor of Science degree in Computer Science and Engineering Chemistry from Stony Brook University. She is interested in large scale network (both wireless and wireline) measurement and monitoring, particularly for security purposes. Prior to Sprint, she worked on introducing JavaTV into European interactive TV standards DVB-MHP at Sun Microsystems and business application platform security at Consilient Inc.