Portscan Detection with Sampled NetFlow

Ignasi Paredes-Oliva, Pere Barlet-Ros, and Josep Solé-Pareta

Universitat Politècnica de Catalunya (UPC), Computer Architecture Dept.
Jordi Girona, 1-3 (Campus Nord D6), Barcelona 08034, Spain
{iparedes,pbarlet,pareta}@ac.upc.edu

Abstract. Sampling techniques are often used for traffic monitoring in high-speed links in order to avoid saturation of network resources. Although there is a wide existing research dealing with anomaly detection, few studies analyzed the impact of sampling on the performance of portscan detection algorithms. In this paper, we performed several experiments on two already existing portscan detection mechanisms to test whether they are robust enough to different sampling techniques. Unlike previous works, we found that flow sampling is not always better than packet sampling to continue detecting portscans reliably.

1 Introduction and Related Work

Traffic monitoring and analysis is essential for security and management tasks. In high-speeds links it is not always possible to process all the incoming packets and sampling techniques (e.g., Sampled NetFlow [1]) must be applied to reduce the load on routers. Robustness against sampling is very important since network operators tend to apply aggressive sampling rates when using NetFlow (e.g., 1/1000) in order to handle worst case scenarios. For this reason, it is fundamental to build sampling-resilient anomaly detection mechanisms.

We focus our study on portscan detection algorithms due to two main reasons. Firstly, they are one of the most common attacks (e.g., they usually precede worm propagation) and, therefore, there is general interest in detecting them reliably. Secondly, portscan attacks can put NetFlow-based monitoring platforms in serious trouble (the nature of this sort of anomalies can overflow flow tables due to the potentially large set of new flows generated by a scanner). Several methods for portscan detection exist. The most basic one flags a scanner when it connects to more than a certain number of destinations during a fixed interval of time. For example, this is the portscan detection algorithm implemented by the Snort IDS [2]. The mechanisms tested in this paper (TRW [3] and TAPS [4]) are more complex and have shown to be reasonably effective. In particular, TRW is implemented in the Bro IDS [5]. Few recent studies have analyzed the impact of sampling on anomaly detection [6–8]. Mai et al. studied the impact of packet sampling on TRW and TAPS in [6]. In the case of TRW, they found out that the flow size became lower in the presence of sampling, thus resulting in more false positives and negatives. They also showed that the metric used by TAPS is less affected, thus concluding that TAPS is more resilient to sampling than

TRW. They also observed that, while TRW had better success ratio, TAPS exhibited a lower ratio of false positives. In [7], they tested packet sampling and three flow-based sampling mechanisms. They concluded that flow sampling was the best choice for anomaly detection under sampling. Finally, Brauckhoff et al. studied how specific metrics are affected by sampling looking at counts of bytes, packets and flows, together with feature entropy metrics [8]. They concluded that entropy summarization is more resilient to sampling than volume-based metrics.

In this study, we analyze the impact of sampling on TRW and TAPS portscan detection algorithms. In particular, we evaluated three sampling techniques: packet sampling, flow sampling and sample and hold. One of the main objectives of this paper is to validate previous results in our network scenario when using Sampled NetFlow data. We also aim to evaluate the impact of the different sampling techniques on portscan methods taking the same fraction of packets, while previous works (e.g., [7]) used instead the portion of sampled flows as the common metric to compare the different sampling methods. Although the amount of memory used by NetFlow to keep the flow tables is directly proportional to the number of flows, we focused on another relevant resource: the CPU cycles. Since in NetFlow every packet must be processed, it is also important to compare the accuracy of all sampling methods according to the ratio of sampled packets. The motivation of this study came from the fact that given a flow sampling rate, the fraction of analyzed packets is significantly different among the sampling methods, which results in an unfair comparison, specially for packet sampling. For instance, according to our traces, sampling 10% of flows results only in 2.86% of sampled packets, while flow sampling gets 10.90% and sample and hold takes even a larger proportion of packets (15.58%).

The rest of this paper is organised as follows. Section 2 presents the tested sampling methods together with the evaluated portscan detection algorithms. In Section 3, we describe our network scenario and the followed methodology. Section 4 shows and discusses the obtained results using real-world NetFlow data from a large university network. Finally, Section 5 concludes the paper and summarises our future work.

2 Background

In this section, we briefly describe the three sampling methods and the two portscan detection algorithms analyzed in this work.

2.1 Sampling Methods

We experimented with three different sampling methods: packet sampling (PS), flow sampling (FS) and sample and hold (SH). PS is widely used because of its low CPU consumption and memory requirements. Flow-based approaches (e.g., FS and SH) overcome some of the shortcomings of PS but, in exchange, they have higher resource requirements. Thus, some trade-off between accuracy and resource requirements is needed.

- **Random packet sampling** takes each packet with probability $p < 1$.
- **Random flow sampling** takes each flow with probability $p < 1$. This technique is usually implemented hashing the flow ID (e.g., the 5-tuple formed by the source and the destination IP addresses and ports, and protocol field). The flow is then selected if the resulting value (mapped to the [0..1) range) is below p [9].
- **Sample and Hold** takes the packet directly if its flow ID belongs to an already seen flow. Otherwise, the packet is sampled with probability $p < 1$. p is computed as $h \cdot s$ (s is the size of the packet and h is the probability of sampling a single byte) [10].

2.2 Portscan Detection Algorithms

Simple portscan detection algorithms, like the one used by the Snort IDS, are not very effective nowadays since attackers can easily evade detection by reducing their scanning rate. There are many other techniques capable of achieving higher rates of detection, such as TRW and TAPS, which we analyze in this paper.

- **Threshold Random Walk (TRW)** [3]. The main idea behind this technique is that one scanner will fail more connections than a legitimate client when trying to establish a connection. Since it is possible to fail some connections even being a good client, the decision of flagging a host as a scanner is not taken just after the first failure. For each source there is an accumulated ratio that is updated each time a flow ends. The update is done according to the flow state: connection established or failed attempt. We did our experiments with an unidirectional trace, so we used the proposed modification of TRW, called TRWSYN [4], that identifies a failed connection when an ended flow is a single SYN-packet. Eventually, if any source IP keeps scanning, it will fail more and more connections and finally it will exceed the established threshold, thus being recognised as a scanner.
- **Time-based Access Pattern Sequential hypothesis testing (TAPS)** [4]. This method is based on the observation that the ratio between the number of destination IPs and the number of destination ports (or the reverse) when the source IP is an scanner is significantly higher than the same ratio when there is no scanning activity. When this relationship is higher than a preconfigured threshold, the per-source IP ratio is updated accordingly. When this accumulated value reaches a certain limit, that source is considered to be a scanner.

3 Scenario and Methodology

We collected a 30-minute NetFlow traffic trace from the Gigabit access link of the Universitat Politècnica de Catalunya (UPC) (see Table 1 for more detailed information). This link connects about 10 campuses, 25 faculties and 40 departments to the Internet through the Spanish Research and Education network

Table 1. Detailed information about the NetFlow trace used in the evaluation and the absolute number of port scanners detected by TRW and TAPS

Date	Start time	Duration	Packets	Bytes	Flows	Total scanners	
						TRW	TAPS
06-11-2007	16:30	30min.	105.38×10^6	61.86×10^9	5.26×10^6	1457	4315

(RedIRIS). Real-time statistics about the traffic of this link are available on-line at [11].

We first implemented the portscan detection techniques and the sampling methods described in Section 2 on the SMARTxAC monitoring system [12]. Then, we ran several tests with varying sampling rates, sampling methods and portscan detection algorithms. In order to have some ground of truth to check our results, we first ran each portscan detection algorithm without sampling (see Section 4 for more details about the used ground truth). After that, we can compare which attacks were missed in each case. We used the following sampling intervals $N = \{1, 10, 50, 100, 500, 1000\}$ to do our experiments.

We configured TRW and TAPS with a false positive ratio of 0.01, probability of detection to 0.99, probability of having a successful connection being a scanner to 0.2 and to 0.8 for a legitimate host as recommended by [3, 4]. After some tests, we fixed the ratio used by TAPS to detect suspicious sources to $Z = 3$.

It is important to note that the sampling rate in the case of *PS* and flow-based sampling techniques has different meanings. While in the first case it refers to the fraction of sampled packets, in the latter case it indicates the portion of sampled flows. This results in a very different number of sampled packets and flows among the different sampling methods. In order to make all the sampling methods comparable, we used the following two metrics:

- **Equal portion of packets**. We first computed the packet sampling rate as $1/N$ for *PS*. Given this fraction of packets to keep, we then performed several tests to find the suitable sampling rates for the other sampling techniques in order to select the same portion of packets.
- **Equal portion of flows**. We computed the flow sampling rate as $1/N$ for *FS*. Given the portion of flows to take, we ran various tests to obtain the correct sampling rate values for *PS* and *SH* in order to sample the same portion of flows.

Tables 2 and 3 present the selected sampling rates that assure that the same portion of packets or flows is selected for all the sampling methods.

4 Performance Evaluation

In this section, we study the impact of *PS*, *FS* and *SH* sampling techniques on TRW and TAPS portscan detection algorithms. We used the following performance metrics:

Table 2. Percentage of selected flows given a portion of sampled packets

N	%packets	*PS*		*FS*		*SH*	
		p	%flows	p	%flows	h	%flows
10	10%	0.1	25.89%	0.092	10.24%	1.06×10^{-4}	6.84%
50	2%	0.02	7.95%	0.026	2.78%	2.8×10^{-5}	2.03%
100	1%	0.01	4.70%	0.015	1.85%	1.5×10^{-5}	1.05%
500	0.2%	0.002	1.44%	0.0036	0.95%	4×10^{-6}	0.53%
1000	0.1%	0.001	0.88%	0.0018	0.77%	2.7×10^{-6}	0.49%

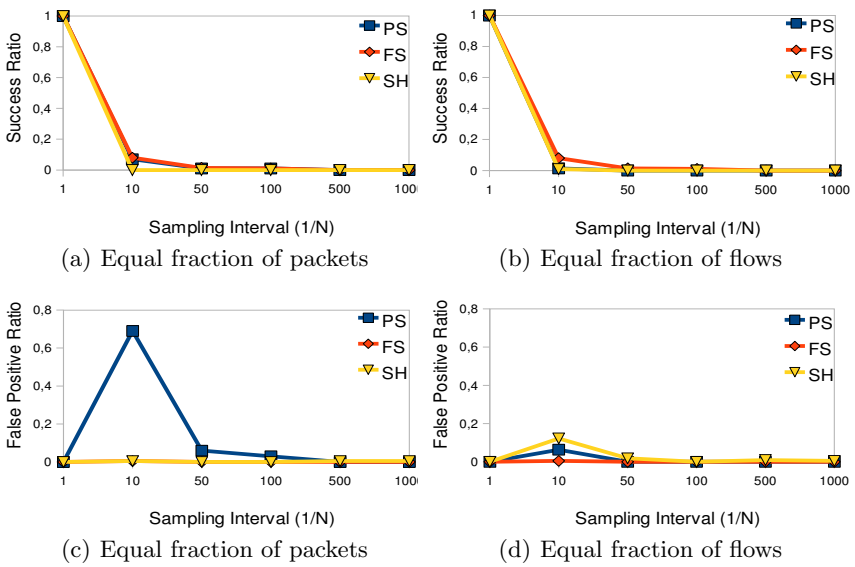
Table 3. Percentage of selected packets given a portion of sampled flows

N	%flows	*PS*		*FS*		*SH*	
		p	%packets	p	%packets	h	%packets
10	10%	0.028	2.86%	0.1	10.90%	1.8×10^{-4}	15.58%
50	2%	0.003	0.33%	0.02	1.60%	2.8×10^{-5}	1.98%
100	1%	1.2×10^{-3}	0.12%	0.01	0.59%	1.5×10^{-5}	1.05%
500	0.2%	1.3×10^{-4}	0.013%	0.002	0.11%	9.511×10^{-7}	0.02%
1000	0.1%	6.2×10^{-5}	0.0062%	0.001	0.05%	9.456×10^{-7}	0.018%

$$success\_ratio = \frac{true\_scanners}{total\_scanners} \qquad \text{and} \qquad false\_positive\_ratio = \frac{false\_scanners}{total\_scanners},$$

where *total_scanners* accounts for our ground truth of scanners (scanners detected by TRW/TAPS without sampling, which are not necessarily real scanners). While *true_scanners* stands for the scanners detected under sampling that also belong to the ground truth, *false_scanners* refers to those detected scanners that fall out of that set. Note that our metrics differ from the classical definitions of success and false positive ratios in that we do not check whether the detected scanners by TRW and TAPS (without sampling) are real scanners or not. This choice lies in the fact that we are interested in evaluating the degradation of the portscan detection algorithms in the presence of sampling rather than in their actual detection accuracy. Table 1 presents the absolute number of portscans in our ground truth (i.e., without sampling).

We first focus on the impact of sampling on TRW. As we can observe in Figures 1(a) and 1(b), the success ratio degrades dramatically for increasing sampling rates regardless of the common metric being used (portion of packets or flows). When the sampling rate is low, TRW still detects few scanners but when it goes up, the success ratio reaches zero rapidly. Regarding the false positives ratio, Figure 1(d) shows that it is relatively low when using the same ratio of flows. When using the same proportion of packets (Figure 1(c)), we can notice that *PS* presents a huge peak that almost reaches 70%, while the flow-based sampling techniques hardly reach 10% of wrongly flagged scanners. As previously pointed out by former works, this peak for $N = 10$ is because of multi-packet flows converted to single SYN-packet flows, thus being flagged as scanners.



(a) Equal fraction of packets (b) Equal fraction of flows

(c) Equal fraction of packets (d) Equal fraction of flows

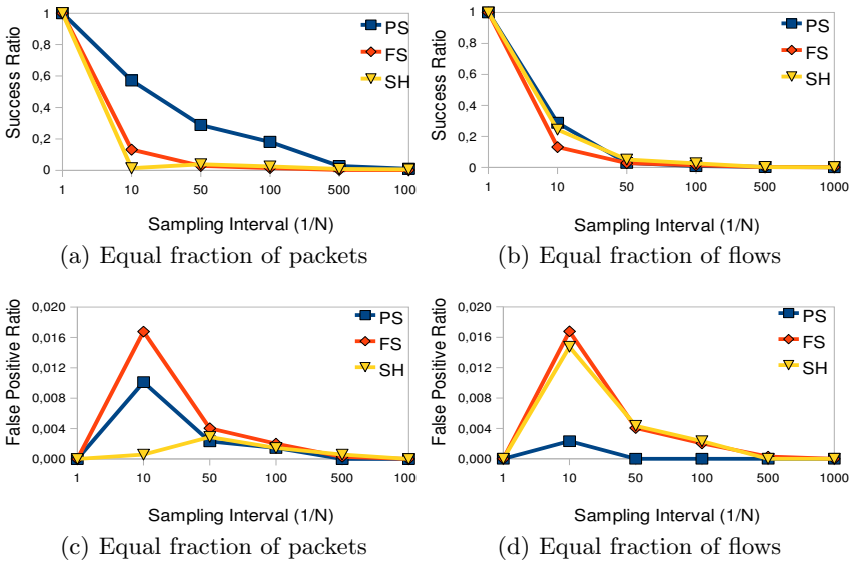
Fig. 1. Impact of sampling on TRW.

When switching to TAPS and looking at the success ratio in Figures 2(a) and 2(b), we can observe that the obtained accuracy is very distinct among the three sampling mechanisms. When performing the experiment under the same fraction of packets, *PS* is clearly the best method, but when the common metric to compare is the ratio of flows, the accuracy is almost equal for all of them. Concerning to the false positives (Figures 2(c) and 2(d)), we observe that it is minimal ($< 2\%$) regardless of the sampling method and the common metric used.

The obtained results using the same fraction of flows showed lower values for both the success ratio and the false positive ratio than previous studies. The variation of the success ratio can be partly explained due to the different traffic traces used. Concerning to the false positive ratio (fpr), its decrease is related to the different followed methodologies. While [7] had approximately an initial $fpr = 0.75$ for their unsampled traces, we considered our ground truth to be classified without any erroneously flagged scanner ($fpr = 0$), thus focusing exclusively on the performance degradation due to sampling. While their fpr reached a ratio of almost 2.5, our maximum value is 0.12 (using the fraction of sampled flows to compare). When using the proportion of sampled packets as the common metric, this ratio increases to 0.7.

4.1 TRW vs. TAPS

As already noticed by previous studies, we were able to detect many more scanners using TAPS than TRW (see Table 1). This can be explained partially due to



(a) Equal fraction of packets (b) Equal fraction of flows

(c) Equal fraction of packets (d) Equal fraction of flows

Fig. 2. Impact of sampling on TAPS.

the fact that TRW only works with TCP scanners and TAPS is connectionless-oriented. Furthermore, recent studies have observed that TRW tends to incorrectly detect P2P activity as scanners [13].

TRW showed to be much less resilient to sampling, while TAPS detected some scanners even for $N = 1000$. TAPS does not depend on any specific packet feature like TRW (which looks for single SYN-packet flows), thus being less sensitive to the particular packet discarded. TAPS also showed less false positives regardless of the common metric and the sampling method used, and it always got lower false positive ratios than TRW (the highest ratio showed by TAPS was 0.017 while TRW reached 0.7). Therefore, we can conclude that TAPS is better under sampling as already noticed by previous works. On the contrary, when using the same fraction of packets as the common metric to compare the different sampling methods under TAPS, we obtained better results using *PS* than flow-based techniques (*FS* and *SH*).

5 Conclusions and Future Work

In this paper, we have performed different experiments on TRW and TAPS to test whether they are robust enough to continue detecting portscans under sampling. Regarding the detection algorithms, we observed that TAPS is significantly better in the presence of sampling. Concerning to the sampling techniques, while flow sampling exhibited better performance than the rest using TRW, with TAPS we observed that packet sampling outperformed the flow-based mechanisms. The results presented in this paper are not entirely aligned with those

obtained in former studies. In particular, it has been previously concluded that random flow sampling was always the most promising sampling method to detect portscans, but according to our results, we have not observed this superiority in all the experiments, thus confirming that this parallel study reveals new interesting information.

Our current work is centred in extending our study to other sampling methods and anomaly detection algorithms. We also plan to further validate the results of this work with more NetFlow traces from several networks.

Acknowledgements

This work was done under the framework of the *COST Action IC0703 "Data Traffic Monitoring and Analysis (TMA)"*. The authors thank UPCnet for the data traces provided for this study.

References

1. Cisco Systems: Sampled NetFlow http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html.
2. Roesch, M.: Snort–lightweight intrusion detection for networks. In: Proc. of USENIX Systems Administration Conference. (1999)
3. Jung, J., Paxson, V., Berger, A., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: Proc. of IEEE Symposium on Security and Privacy. (2004)
4. Avinash, S., Ye, T., Supratik, B.: Connectionless portscan detection on the backbone. In: Proc. of IEEE International Performance Computing and Communications Conference. (2006)
5. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer Networks **31**(23-24) (1999)
6. Mai, J., Sridharan, A., Chuah, C., Zang, H., Ye, T.: Impact of packet sampling on portscan detection. IEEE Journal on Selected Areas in Communications **24**(12) (2006)
7. Mai, J., Chuah, C., Sridharan, A., Ye, T., Zang, H.: Is sampled data sufficient for anomaly detection? In: Proc. of ACM SIGCOMM conference on Internet measurement. (2006)
8. Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A.: Impact of packet sampling on anomaly detection metrics. In: Proc. of ACM SIGCOMM conference on Internet measurement. (2006)
9. Duffield, N.: Sampling for passive internet measurement: A review. Statistical Science **19**(3) (2004)
10. Estan, C., Varghese, G.: New directions in traffic measurement and accounting: focusing on the elephants, ignoring the mice. ACM Transactions on Computer Systems **21**(3) (2003)
11. IST-Lobster sensor at UPC: http://loadshedding.ccaba.upc.edu/appmon.
12. Barlet-Ros, P., Solé-Pareta, J., Barrantes, J., Codina, E., Domingo-Pascual, J.: SMARTxAC: a passive monitoring and analysis system for high-speed networks. Campus-Wide Information Systems **23**(4) (2006)
13. Falletta, V., Ricciato, F.: Detecting scanners: empirical assessment on a 3G network. International Journal of Network Security **9**(2) (2009)