Evaluating the Impact of Traffic Sampling on AATAC's DDoS Detection

Gilles Roudiere, Philippe Owezarski

Evaluating the Impact of Traffic Sampling on AATAC's DDoS Detection

Gilles Roudière, Philippe Owezarski
CNRS, LAAS, 7 avenue du colonel Roche Toulouse Cedex 4, France
{gilles.roudiere, philippe.owezarski}@laas.fr

***Abstract*—As Distributed Denial of Service (DDoS) attack are still a severe threat for the Internet stakeholders, they should be detected with efficient tools meeting industrial requirements. In a previous paper, we introduced the AATAC detector, which showed its ability to accurately detect DDoS attacks in real time on full traffic, while being able to cope with the several constraints due to an industrial operation. However, in a realistic scenario, network monitoring is done using sampled traffic. Such sampling may impact the detection accuracy or the pertinence of produced results. Consequently, in this paper, we evaluate AATAC over sampled traffic. We use five different count-based or time-based sampling techniques, and show that AATAC's resources consumption is in general greatly reduced with little to no impact on the detection accuracy. Obtained results are succinctly compared with those from FastNetMon, an open-source threshold-based DDoS detector.**

I. Introduction

Distributed Denial of Service (DDoS) attacks consist in numerous hosts sending fake requests towards a victim's network. Those requests exhaust the victim's system resources, either its available computing power, networking bandwidth or even some protocol implementation specific resources. Thus, they can cause up to the system's total inability to handle legitimate clients' requests, and thus should be detected. However, appropriate detectors have to meet the industry requirements (real-time operation, accuracy, autonomy...)

As today's bandwidths are very large, most actual monitoring systems sample the traffic in order to reduce the size of the data to be processed, limiting the computational resources used for monitoring purposes. [1] Consequently, a realistic DDoS detector should be efficient even when processing sampled traffic.

In a previous paper we introduced AATAC (Autonomous Algorithm for Traffic Anomaly Characterization), a fully autonomous anomaly detector that focuses on DDoS attacks. We proved, on some full traffic traces, that it can perform an efficient detection while consuming a very limited amount of resources. However, to be used in an industrial context, AATAC should also be able to perform efficiently over sampled traces. This is why in this paper we evaluate the AATAC algorithm over sampled traffic. We use several types of sampling techniques and rates, the aim being to understand which sampling techniques would be the most appropriate for maximizing the DDoS detection using AATAC. The results are compared with those from FastNetMon, the only publicly available DDoS detector we manage to install on our testbed.

The remaining of this paper is organized as follows. In section II we introduce other detectors built to operate over sampled traces. Our detector AATAC is briefly introduced in section III, while its evaluation over sampled traces is discussed in sections IV and V. Section VI concludes the paper and discusses possible future works.

II. Related works

In this section, we present related works that study the impact of sampling on several monitoring tools, but mainly focus on the network anomaly detection.

Several related approaches rely on a flow-based sampling. Bartos et al. [1] study the impact of sampling on network anomaly detection. They propose an adaptive flow-level sampling technique that manages to limit the impact of sampling on the performance of a network behavioural anomaly detector. Other flow-based sampling techniques are presented by Jadidi et al. [2] or Andriolakis [3]. The problem with such approaches is that extracting flows from the traffic is already a complex task that has to deal with already packet sampled traffic. The main bottleneck for monitoring system is definitely situated at the packet level, even for systems that provide flow based information. Thus, we instead focus on packet sampling techniques as they are the most likely to significantly impact the computational cost reduction.

A framework to evaluate the impact of packet sampling over several various tools is presented in [4]. The author discusses the very generic performances of each sampling algorithm. They propose a set of metrics that allow the evaluation of each technique's ability to produce a sampled traffic that efficiently represents the original one. Their results are really generic and do not target the network anomaly detection problem specifically, they thus might not apply in our situation. An extension of this study is presented in [5].

Jun et al. [6] propose an adaptive sampling technique that intends to keep the amount of sampled traffic below a maximum inspection capability. They use the SDN technology to distribute the sampling over several switches. They perform their evaluation using Snort [7] and Suricata [8], two detectors based on Deep Packet Inspection. As such techniques focus on the packets content, instead of more encompassing statistical

[1] This processing is often performed on the switches/routers whose main task is to switch/route packets, and are then largely loaded by this main task. Other tasks must then consume resources parsimoniously.

features of the traffic, the technique might not suit all detection algorithms (including AATAC).

In an older paper, Brauckhoff et al. [9] evaluate the impact of sampling over anomaly detection metrics. Their evaluation uses traces containing the Blaster worm to evaluate several detection techniques at various sampling rates. With a detector they propose, and thanks to an entropy-based summarization technique, authors achieve a good detection independently from how high is the sampling rate. The paper also shows that flow-based detection is more impacted by the packet sampling than packet-based detection.

III. AATAC ALGORITHM OVERVIEW

AATAC is a balanced solution to the DDoS detection problem. It provides a real-time detection with low computing resources while still producing pertinent and eloquent results. It is a fully autonomous detector relying on unsupervised machine learning techniques, requiring very few configuration or updates. In this section we present only a brief overview of the algorithm. For more technical details, please refer to the original publication [10].

AATAC's processing is illustrated on Figure 1. It is split into two distinct parts: a continuous and a discrete one. The first one quickly handles instances and maintain a data structure representing the traffic in real time. The second one uses this data structure to store, at a regular interval, a snapshot of the traffic. Those snapshots are then stored and used to detect anomalies in the traffic. As those snapshots can be plotted into a set of two-dimensional graphs, they provide the network administrator with a dynamic and pertinent view of the traffic properties when the anomaly occurs.

A. Continuous processing

To build a representation of the traffic at any time, the continuous processing relies on a statistical analysis on the traffic. Its processing is inspired from D-Stream [11], a grid-based clustering algorithm.

To characterize the traffic properties, AATAC uses grids. Those grids are either grouped as histograms, to characterize a traffic feature distribution, or alone, to characterize a global feature of the traffic. Whenever AATAC processes a traffic instance (here a traffic flow and its associated characteristics), AATAC determines a set of grids it falls in. It then increases a density value assigned each of those grids (any instance might be weighted depending on the monitored feature). This processing is done in an incremental manner, which speeds up the processing and limits the computational resources required.

Over time, the density of each grid exponentially decays towards 0. How fast the densities decays is determined by the *decay factor* (λ). For reasons made explicit in [10], λ is set using the R parameter, defined as $R = \lambda^{\Delta T}$, where ΔT is the parameter defining the time interval between two snapshot creations.

This incremental weighting technique inherently gives much more weight to recent instances in the model than to the older ones. Thus, at a given instant, all grids together form a short-term complete characterization of the traffic.

B. Discrete processing

The discrete processing is executed at a regular interval ΔT, it can be split into three steps. First, it updates the continuous processing data structure, it then creates a traffic snapshot and finally detects anomalous behaviours.

As the continuous processing is incremental, the whole set of grids' densities might not create a consistent characterization at a given instant. The grids' densities are thus updated to be consistent with each other. Also, to avoid the number of stored grids to overgrow in the memory, several grids having a density lower than a really low value D_l are removed from the data structure.

For performance reasons, AATAC's data structure cannot be stored as is. Grids organized as histograms are thus simplified, and transformed into histogram prototypes, a lighter and faster to process data structure. The traffic snapshot is created by storing all histogram prototypes along with the up-to-date densities characterizing global traffic features. This set constitutes the *snapshot features*.

Finally in the anomaly detection phase, AATAC compares the lastly created snapshot to the set of N last snapshots. This is done using the k Nearest-Neighbour (kNN) algorithm applied for each analyzed snapshot feature. kNN produces an outlierness score that AATAC uses to determine how different is the lastly created snapshot from the previously created ones. Note that the scores are normalized.

The final anomaly detection is performed by detecting if the kNN score goes over a given threshold for any snapshot feature. If so, an alarm is raised. As the histogram prototypes and the global densities can be plotted, this alarm is given to the network administrator along with a set of graphs allowing him the have a pertinent and dynamic view of the traffic features when the anomaly occurs.

IV. EVALUATING THE IMPACT OF SAMPLING ON THE DDOS DETECTION

In this section, we describe the testbed we used to evaluate AATAC's performances over sampled traffic.

A. Sampling techniques

Basically, a packet sampling algorithm aims at selecting whether or not a packet should be sampled for further analysis. To evaluate AATAC's performances in various situations, we used five different sampling techniques with several sampling rate: *systematic count-based*, *probabilistic*, *1-out-of-N*, *systematic time-based* and *random time-based*. Those techniques can be split into two categories: count-based or time-based. They cover the full range of common techniques for traffic sampling.

1) Count-based selection techniques: Count-based selection techniques give all packets the same probability to get sampled, independently from their content or arrival time. Statistically, this implies selecting one packet out of N.

The first approach we use is the systematic approach. It consists in systematically selecting a packet every N packets. This approach is pretty straightforward to implement and gives an accurate representation of the traffic in most situations.

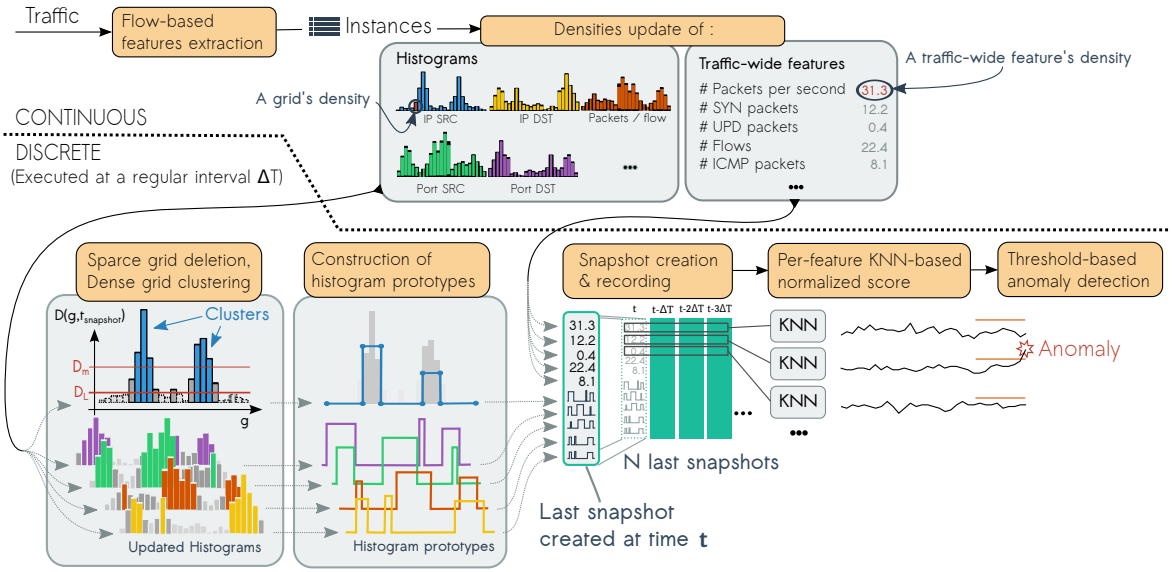


Fig. 1: An overview of the AATAC algorithm

However, it might suffer from a bias if the monitored features exhibit a periodic behaviour.

The random probabilistic sampling runs a random test whenever a packet arrives and selects it with a given probability p. As it is does not require an analysis of the packet's content, our approach uses a uniform probabilistic sampling that selects each packet with the same probability.

The last random count-based technique we use is the *n-out-of-N* sampling. Every N packets, this technique chooses a random set of n indexes between 0 and $N-1$. Those indexes correspond to the indexes of the n packets that will be selected among the next N packets. In most cases, n is set to 1. The *1-out-of-N* sampling is easier to implement while showing similar performances as any other n values.

2) Time-based techniques: Time-based techniques operate by defining a set of dates when a packet selection should be triggered. They generally do that by computing a waiting time interval between each selection. Unlike count-based sampling, time-based sampling does not produce an output that is proportional to the input in terms of number of packets. They instead smooth the variations of the traffic, which might not suit all detection techniques.

If the time between two selections is constant, such approach is said systematic. Like the count-based systematic approach, this technique might suffer from a bias if the traffic exhibits a periodic behaviour.

Other time-based techniques might randomly select the time to wait between two triggers. The technique we use generates an inter-selection interval following an exponential distribution.

3) Other techniques: Various other techniques have been proposed in the literature. For example, adaptive sampling techniques adapt the sampling rate depending on the load of the system (CPU, memory, bandwidth...). Other techniques, said hybrid, use a combination of two other techniques. While being interesting, those techniques do not suit our requirements as they introduce several other parameters that would have made difficult a generalization of the results. Also, such techniques are in fact not common in an industrial context. They require either a knowledge database, or setting up a feedback loop from a performance monitoring system to the sampling hardware, which is not easy to set up and maintain.

B. The testbed

To evaluate the performances of AATAC, we implemented it using C++. The experiments are run on a standard computer, featuring a 3.00GHz Intel Xeon CPU (E5-2623 v3). It features 8 cores but our implementation does not handle multi-threading.

In the context of the ONTIC project, we built the SynthONTS labelled dataset to evaluate our algorithm. It contains a one hour long, payload-free and anonymized real traces captured at the entrance of a large cloud service provider. In an emulated network, we generated 12 realistic attacks and inserted them into the dataset. This set mainly includes DDoS attacks, such as fraggle attacks, smurf attacks, Syn flooding and UDP flooding attacks. The dataset is publicly available on the website of the ONTIC project [12].

We sampled the traces thanks to a custom tool called PCAPsampler [13]. For count-based algorithms, we used the sampling rate of one packet out of 500, 1000, 2000, 5000 or 10000 packets. For time-based algorithms, we picked one packet every 1, 0.1, 0.01 or 0.001 seconds. As the average packet rate is around 67k packets per-seconds in the dataset, this corresponds to values between picking one packet out of 67000, to one out of 67. According to our industrial partners, a sampling rate of one out of 2000 is a common value used on operation. Those value ranges were thus chosen considering this information, but still wide enough for significantly covering different sampling rates.

Regarding the parameters of AATAC, we used the parameters exhibiting the better performances in our previous evaluation [10]. Thus, we set $N = 500$, $\Delta T = 1\ s$ and $R = 0.9$. To adapt to the sampling rate, the computed instances weights are multiplied according to the sampling rate.

While our implementation of AATAC is able to directly operate from pcap files, this is not possible with FastNetMon. We thus had to replay the whole set of traces (the 1h long traces multiplied by the 12 attacks to evaluate) to evaluate its accuracy. As this requires some time, we only evaluated FastNetMon's performances with the 1-out-of-N sampling algorithm considering a sampling rate of one packet out of 2000.

C. Evaluation methodology

To measure the accuracy of the detector, we run both implementations over the sampled traces. We consider an alarm as a true positive if it is raised while the anomaly is occurring, a false positive if it is not. To analyse the results, we use the well known Receiver Operating Characteristic (ROC) curve. However, as they suffer from the *base-rate fallacy* [14], we complete our evaluation by plotting the IDS operation curve of the detector. This method proposed by Nasr et al. [15], consists in plotting the *positive predictive value* (PPV) along with the false positive rate. Each plotted curve is compared to the *zero reference curve (ZRC)*, the operation curve of an ideal detector that should detect all anomalies while producing an increasing number of false positives.

The *intrusion detection effectiveness* ($E_{ID} \in [0, 1]$), is then extracted from this curve. It corresponds to the normalized area between the actual IDS operation curve and the ZRC for a FPR between 0 and a maximum acceptable false positive rate T_{FP}. The lower the E_{ID}, the more effective the detector.

To evaluate how sampling impacts the computational resources required to run the algorithm, we measure how much time our implementation spends into either the continuous or the discrete parts of the algorithm. For the continuous part, we plot the time required to process one second of traffic, and for the discrete part, we measure the time required to create a single snapshot.

V. Results

A. Detection accuracy

Figure 2 shows the ROC curves we obtain for the five sampling algorithms. All count-based sampling techniques achieve good results, but the 1-out-of-N technique seems to produce better and more stable results than other ones. From the time-based sampling techniques, the systematic sampling achieves the best results. For high sampling rates, the systematic time-based approach seems to produce better results than the unsampled traffic while for lower sampling rates, with one packet every $100ms$ or $1s$, the results are not as good. Indeed, those sampling rates correspond to one packet every 6700 or 67000 packets. The count-based sampling algorithms produce better results for similar sampling rates, as they detect more attacks for equivalent false-positive rate.

	Unsampled	Systematic count-based				
T_{FP}		1/500	1/1000	1/2000	1/5000	1/10000
0.01	0.1543	0.0853	0.1023	0.1197	0.1051	0.1568
0.001	0.0680	0.1026	0.0819	0.0789	0.0958	0.1053
0.0001	0.0155	0.2602	0.0208	0.0163	0.0263	0.0256

	1-out-of-N					Probabilistic				
T_{FP}	1/500	1/1000	1/2000	1/5000	1/10000	1/500	1/1000	1/2000	1/5000	1/10000
0.01	0.0804	0.0903	0.0968	0.1371	0.1204	0.0758	0.0822	0.0909	0.1285	0.2127
0.001	0.0910	0.0804	0.0851	0.0902	0.0899	0.0661	0.0649	0.1014	0.1092	0.2328
0.0001	0.0286	0.0197	0.0274	0.0351	0.0654	0.0232	0.0136	0.0276	0.0390	0.0610

	Systematic time-based				Random time-based			
T_{FP}	1ms	10ms	100ms	1s	1ms	10ms	100ms	1s
0.01	0.0631	0.0394	0.2959	0.8424	0.0851	0.0939	0.2954	0.2402
0.001	0.2308	0.0242	0.2113	1.0000	0.1364	0.1127	0.1580	0.1607
0.0001	0.8993	0.0101	0.1238	1.0000	0.2495	0.0207	0.0274	0.0927

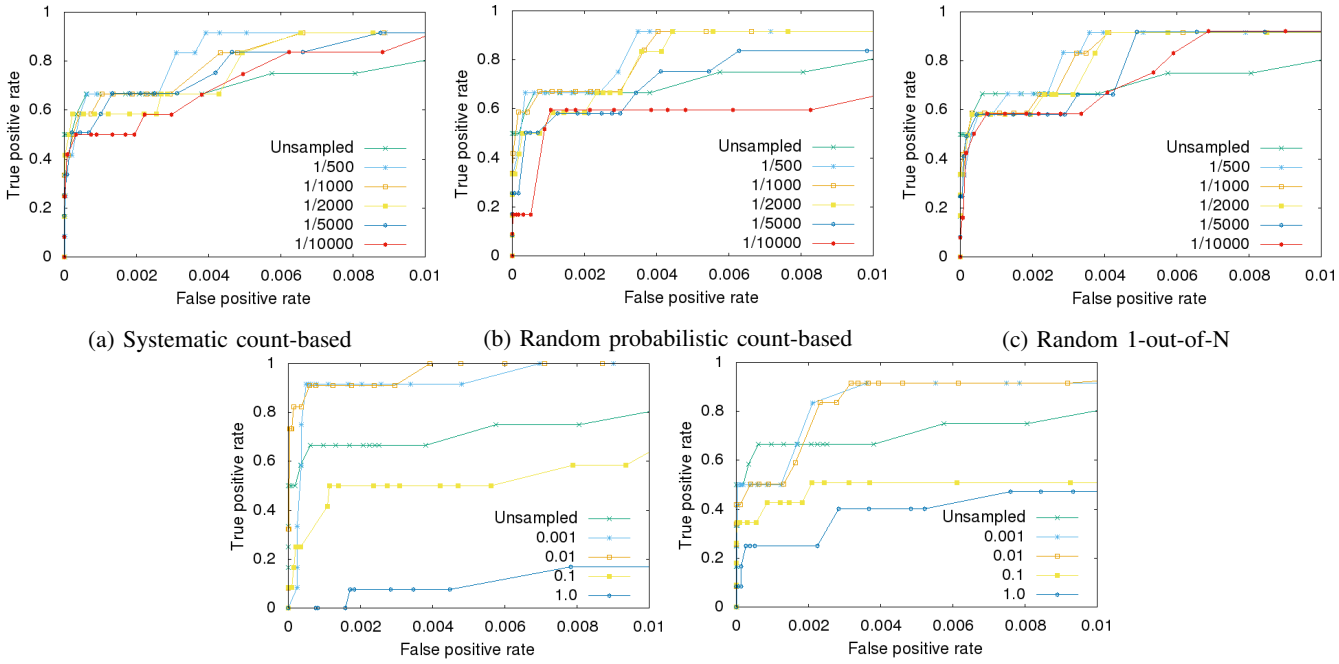
TABLE I: Intrusion Detection Effectiveness for all [2]*sampling techniques* and several values of T_{FP}.

As we can see from the curves, once the false-positive value reaches a given value, AATAC seems to perform better with the sampled traffic than with the unsampled one. As stated before, we multiplied the weight of instances depending on the sampling rate. This has increased the detector's sensitivity.

That being said, the results are more significant with the lowest values of the FPR. Indeed, as we run a test every second, even a FPR of 0.001 means raising a false positive every 17 minutes, which is not acceptable in a real situation. With very low false positive rates, such as 10^{-4}, AATAC performs better with unsampled traffic or high sampling rates. The IDS curves in Figure 3 confirm this analysis. Considering the PPV values for acceptable false-positive rate (below $T_{FP} = 10^{-3}$), AATAC's with unsampled traffic IDS operation curves stays above the other ones. However, regardless of the sampling algorithm, the efficiency decrease stays very low. This is shown in Table I: the Intrusion Detection Effectiveness only increases by a very low amount when the sampling rate reduces. This table also confirms that the *1-out-of-N* technique has the most stable results, especially for $T_{FP} = 10^{-3}$.

From those results, time-based sampling stands out. With high sampling rates, AATAC achieves a detection that is exceptionally good, looking better than with unsampled traffic. This may be explained by the fact that time-based sampling tends to smooth out the short term variations of the traffic, reducing the FPR. Also, AATAC relies on densities that depend on the number of packet received per second. Making this rate constant makes the detection less dependent on the packet rate itself, and more on other properties of the traffic. This has probably enhanced the detection.

Regarding the impact of sampling over FastNetMon accuracy, our evaluation showed that FastNetMon was completely unable to perform its detection over sampled traffic. Indeed, with standard sampling rate of one packet out of 2000, FastNetMon generated 7 false positives in our evaluation, while detecting no attacks out of 12. Decreasing the threshold only increased the number of false positives generated by FastNetMon. The detector's inability to detect DDoS is probably due to the fact it operates at an IP address level. Indeed, at



(a) Systematic count-based (b) Random probabilistic count-based (c) Random 1-out-of-N

(d) Systematic time-based (e) Random time-based

Fig. 2: ROC curves for the several sampling algorithms and multiple sampling rates.

such sampling rates, the amount of data corresponding to a specific IP address is too low for the detector to take a pertinent decision. This implies raising more false positive.

As a conclusion, we can see that AATAC performs an accurate detection independently for the sampling technique used. It even performs better at high sampling rates with time-based sampling. The sampling rate has a very limited impact on the quality of AATAC's detection, which is not the case for FastNetMon, which was totally unable to operate with sampled traffic at a standard sampling rate.

B. Computing power requirements

The calculated processing time of AATAC for each sampling algorithm are depicted on Figure 4. To process the traffic in real time, the continuous processing was designed to process the traffic with a linear complexity regarding the number of instances. This is confirmed on Figures 4a and 4b. As the X axis corresponds to the inverse of how many packets are processed per second, the processing time follows a $x \mapsto \frac{1}{x}$ pattern.

Regarding the discrete processing, we can see on Figures 4c and 4d that the sampling has little impact on the time required to create a snapshot. Indeed there is no specific reason why the treatment should be more complex with a snapshot built from a sampled traffic than with unsampled ones. However, we can see a slight increase of the processing time when few packets per second are sampled. This is probably due to the fact that a stronger sampling implies that the impact of each instance has to be higher on the model. This thus produces a more varying characterization of the traffic which might create more complex histogram prototypes. This impact is however minimal.

We can also see from the several curves that only the sampling rate has an impact on the processing time, while the sampling technique has none.

This evaluation proved that using AATAC with sampled traffic is possible and allows a significant resource consumption reduction, proportional to the sampling rate.

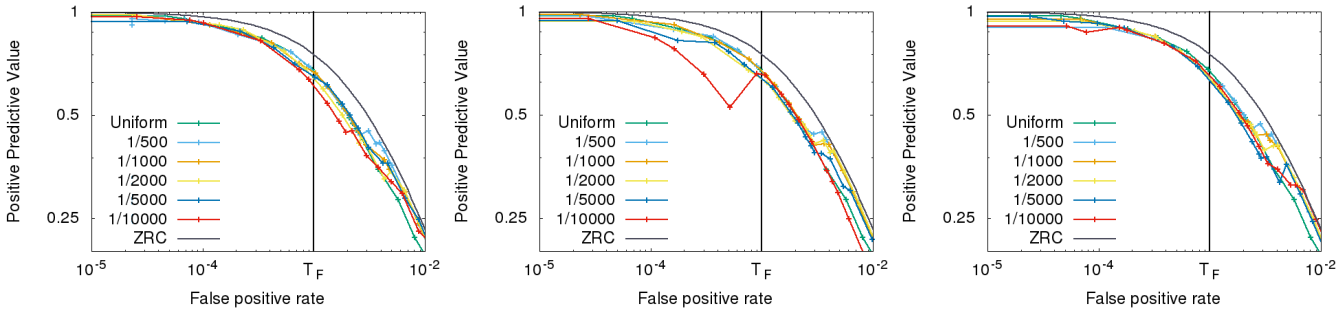
VI. Conclusion

In this paper, we evaluated the AATAC DDoS detector's performances over sampled traffic. For sampling rates going up to 1 packet out 2000, AATAC has shown an almost unchanged detection accuracy while benefiting from a substantial reduction of the required computational resources. From the several sampling techniques used, count-based sampling techniques produce results less dependent on the sampling rate (especially for the *1-out-of-N* technique). However, at very high sampling rates, time-based sampling techniques tend to produce better results, as they smooth the short-term variations of the traffic. For further studies, it thus would be interesting to study a new sampling technique. A ideal one would probably use an adaptive mechanism, combining both time-based sampling for high sampling rates and count-based for lower ones.

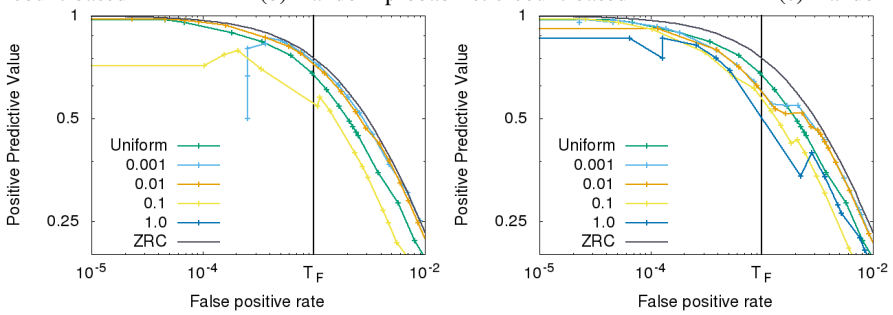
References

[1] K. Bartos, M. Rehak, and V. Krmicek, "Optimizing flow sampling for network anomaly detection," in *IWCMC 2011 - 7th Int. Wirel. Commun. Mob. Comput. Conf.*, pp. 1304–1309, 2011.

[2] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and K. Singh, "A Probabilistic Sampling Method for Efficient Flow-based Analysis," *J. Commun. NETWORKS*, vol. 18, no. 5, pp. 818–825, 2016.

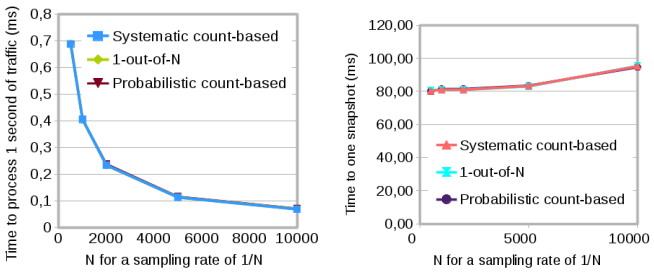


(a) Systematic count-based (b) Random probabilistic count-based (c) Random 1-out-of-N

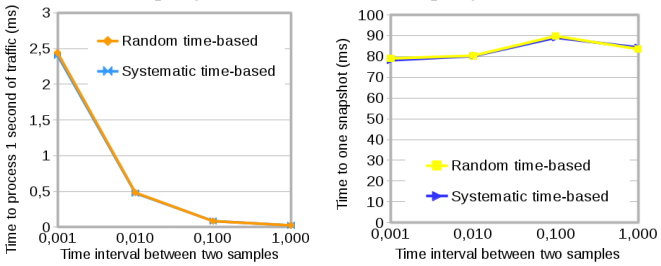


(d) Systematic time-based (e) Random time-based

Fig. 3: IDS operation curves for the several sampling algorithms and multiple sampling rates.



(a) Continuous processing for count-based sampling (b) Discrete processing for count-based sampling



(c) Continuous processing for time-based sampling (d) Discrete processing for time-based sampling

Fig. 4: AATAC's processing times for several sampling techniques and sampling rate

[3] G. Androulidakis and S. Papavassiliou, "Intelligent flow-based sampling for effective network anomaly detection," in *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 1948–1953, 2007.

[4] J. M. C. Silva, P. Carvalho, and S. R. Lima, "A Modular Sampling Framework for Flexible Traffic Analysis," 2015.

[5] J. M. C. Silva, P. Carvalho, and S. R. Lima, "Analysing traffic flows through sampling: A comparative study," in *Proc. - IEEE Symp. Comput. Commun.*, vol. 2016-Febru, pp. 341–346, 2016.

[6] J.-h. Jun, D. Lee, and S.-h. Kim, "DDoS Attack Detection Using Flow Entropy and Packet Sampling on Huge Networks," *Thirteen. Int. Conf. Networks.*, no. c, pp. 185–190, 2014.

[7] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks.," *LISA '99 13th Syst. Adm. Conf.*, pp. 229–238, 1999.

[8] suricata, "https://suricata-ids.org/,"

[9] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*, pp. 159–164, 2006.

[10] G. Roudière and P. Owezarski, "A Lightweight Snapshot-Based DDoS Detector," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017.

[11] Y. Chen and L. Tu, "Density-Based Clustering for Real-Time Stream Data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, pp. 133–142, 2007.

[12] ONTIC, "http://ict-ontic.eu/," Accessed: 2017-05-12.

[13] PCAPSampler, "https://github.com/groud/pcapsampler,"

[14] M. Bar-Hillel, "The Base-Rate Fallacy In Probability Judgments," *Acta Psychol. (Amst).*, vol. 44, no. 3, pp. 211–233, 1980.

[15] K. Nasr, A. A.-e. Kalam, and C. Fraboul, "Performance Analysis of Wireless Intrusion Detection Systems," in *Internet Distrib. Comput. Syst. 5th Int. Conf. IDCS 2012, Wuyishan,*, pp. 238–252, 2012.