A Lightweight Snapshot-Based DDoS Detector

Gilles Roudière, Philippe Owezarski
LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{gilles.roudiere, philippe.owezarski}@laas.fr

***Abstract*—Despite the efforts made from both the research community and the industry in inventing new methods to deal with distributed denial of service attacks, they stay a major threat in the Internet network. Those attacks are numerous, and can prevent, in most serious cases, the targeted system from answering any request from its clients.**

Detecting such attacks means dealing with several difficulties, such as their distributed nature or the several evasions techniques available to the attackers. The detection process has also a cost, which includes both the resources needed to perform the detection and the work of the network administrator.

In this paper we introduce AATAC (Autonomous Algorithm for Traffic Anomaly Detection), an unsupervised DDoS detector that focuses on reducing the computational resources needed to process the traffic. It models the traffic using a set of regularly created snapshots. Each new snapshot is compared to this model using a *k-NN* based measure to detect significant deviations toward the usual traffic profile. Those snapshots are also used to provide the network administrator with an explicit and dynamic view of the traffic when an anomaly occurs.

Our evaluation shows that AATAC is able to efficiently process real traces with low computational resources requirements, while achieving an efficient detection producing a low number of false-positives.

I. Introduction

When considering network anomalies, one of the most concerning problem stays Distributed Denial of Service (DDoS) attacks. Using a set of compromised hosts all over the Internet, these attacks try to prevent the victim servers from delivering a proper service by sending illegitimate requests to the victim's network. This exhausts some of the victim's routers or servers resources causing from small disturbance of the victim's operations to its total inability to handle legitimate clients' requests. With the multiplication of connected devices and the emergence of IoT, these kind of attacks is an increasing threat. Therefore, being able to prevent DDoS attacks is a priority for numerous Internet stakeholders.

Dealing with anomalies in general is a costly process requiring, inter alia, the expertise of a network administrator. As his time is a valuable resource, he needs specific tools that enable a relevant and fast decision-making, reducing his work as much as possible. Thus, setting up, configuring and keeping up to date an anomaly detector should be an easy task. On every-day operation, it should be able to handle traffic anomalies as autonomously as possible, while providing pertinent and accurate information to the network administrator for the diagnosis phase. Finally, such detector needs to be scalable towards the traffic bandwidth, enabling a real time operation with a reasonable amount of allocated resources.

Adding to those problems, the DDoS detection needs specific approaches able to tackle the following challenges:

- DDoS can harm a whole system really fast, including the DDoS detection and mitigation services. Thus DDoS should be detected as soon as possible to avoid irreversible damages.
- The detector should be able to process traffic in real-time even under network saturation.
- Isolating the malicious traffic from the legitimate one is a difficult task: first because the malicious packets overwhelms the traffic, and secondly because the DDoS traffic —as it consists in fake requests towards the victim network— resembles the legitimate one.
- If the attacker uses IP spoofing, tracking the source of the attack is difficult.

Despite the constant interest of the research community, no specific detector emerged as a perfect or consensual solution to the DDoS detection problem.

Our work was conducted in the context of the AATAC (Autonomous Algorithm for Traffic Anomaly Detection) project that aims at finding practical solutions to the DDoS detection problem. This project is carried out in collaboration with Border 6, a software editor that offers a BGP routing optimization solution. As their clients regularly suffer from such attacks, Border 6 is naturally interested in investigating new solutions to the DDoS detection and mitigation problem.

In this paper, we introduce a new unsupervised anomaly detection algorithm eponymously named AATAC. This detector specifically focuses on DDoS attacks. It uses regularly taken traffic snapshots to detect unexpected changes in the whole network traffic. AATAC is an autonomous detector, able to detect anomalies as soon as they occur while processing the traffic in real time with a low amount of resources needed. It also produces a low number of false-positives while having a good detection rate. More than that, the last created snapshots can be plotted when an anomaly occurs. This set of graphs provides a dynamic view of the traffic that helps the network administrator decision making regarding the anomaly.

The rest of the paper is as follows: related works are presented in section II. The AATAC algorithm is presented in section III while its performances are evaluated in section IV. Section V concludes the paper and considers future works.

II. Related works

In an industrial context, autonomy is probably among the most important properties of a detector. For this reason, we present here several related works from the least to the most autonomous approaches, grouped into three categories. We first introduce several knowledge-based approaches, which heavily rely on expert knowledge of existing attacks. Other approaches based on the traffic properties statistical analysis are then presented, followed by most recent approaches based on machines learning techniques. We present here detectors that are representative of approaches that are either well known, or from recent state-of-the-art techniques. We focus on detectors that have practical developments, i.e. that could be used in an industrial context.

A. Knowledge-based

Also known as *Misuse-based*, the knowledge-based techniques rely on knowledge gathered about seen anomalies. They thus provide a detection focused on known attacks. Expert systems, signature analysis and state transition analysis are common knowledge-based approaches.

Gil and Poletto [1] presented the well known MULTOPS (MUlti-Level Tree for Online Packet Statistics) detector. This approach monitors several traffic characteristics in a tree-based structure, where each node stores the packet rate statistics of a given subnet. MULTOPS processing assumes that DDoS attacks produce unusual imbalanced traffic flow. It thus detects bandwidth attacks by spotting significant imbalance between incoming and outgoing packet rates. MULTOPS needs few resources to operate, but outputs only the per-subnet packets rate, a non-comprehensive information complicating the diagnostic phase. Wang et al. [2] present a new way to model DDoS attacks by using Augmented Attack Tree (AAT). This approach models the attack goals (tree nodes) along with attack means (tree branches), and detects a set of known types of DDoS. However, it does not detect unknown attacks. FastNetMon [3] is an open-source threshold-based detector that provides a practical solution to the DDoS detection and mitigation. FastNetMon uses a set of counters to detect abnormally high bandwidths or packet rates towards given subnets. With few computational resources, FastNetMon generates alarms with information on detected attacks, and can automatically run a banning script. However it might fail in correctly segregating the malicious traffic from the legitimate one.

In general, Knowledge-based techniques, despite producing a low number of false-positives, need a lot of work from the network administrator to keep the detection up to date with state-of-the-art attacks. Indeed, either building new signatures or setting up a new specialized detector for each new attack is a tedious task. Moreover, having a detector (or a rule) dedicated to each attack might imply high computational requirements when the number of attacks to be detected increases.

B. Statistical

To overcome knowledge-based detectors limitations, researchers proposed techniques based on anomaly detection. These detectors autonomously monitor the patterns of the traffic and detect events (flows, packets, etc...) that deviate from those usual patterns. The most common approaches rely on statistical analysis.

Udhayan and Hamsapriya [4] introduce the statistical segregation method (SMM). This method samples the flows in consecutive intervals, compares the samples towards the attack state condition, and sorts them according to the mean as a parameter. A final correlation analysis is then performed to separate attack flows from the legitimate ones. A flow-based detector is introduced in [5] which analyses the fast entropy (modified version of the entropy) of requests per flow. An adaptive threshold is finally computed to detect anomalies within the traffic. The AFEA detector [6] is also based on similar techniques. Özçelik and Brooks [7] present a detector using the traffic headers entropy post-processed with a wavelet filter and CUSUM to improve the detection accuracy. The entropy, as a measure of the predictability of the traffic features, as been commonly used to detect DDoS attacks.

C. Machine learning based

Various DDoS detectors proposed in the literature take advantage of the recent emergence of machine learning techniques. Such techniques are meant to autonomously extract metrics that encompass the traffic characteristics, either from a labelled (semi-supervised techniques) or an unlabelled traffic dataset (unsupervised techniques). They usually provide more information than statistical techniques.

Supervised machine learning autonomously learn the traffic characteristics using a hand-crafted labelled dataset, then detect deviation toward the produced model while in operation. These approaches are not truly autonomous, as they need to be re-trained every time the traffic evolves. Such approaches can be based on Artificial Neural Network [8], SVM [9] or decisions trees [10].

Most recent approaches use unsupervised learning techniques. Based on the assumption that anomalies are rare events, they autonomously build a model of the usual traffic and detect significant deviations from this model. Consequently, these approaches do not need any previous knowledge on the traffic characteristics, they are thus flexible need few work from the network administrator. However, they usually need a lot of computing power, as characterizing the traffic without any previous knowledge is a difficult task.

For example, unsupervised Artificial Neural Network were widely used for DDoS detection, as with [11], [12] or more recently [13]. Other unsupervised approaches might rely on Nearest-Neighbours based techniques [14] or clustering [15]. The recent framework STONE [16] provides both the DDoS detection and mitigation. To model the traffic, STONE clusters arriving flows using their source prefix. Some clusters properties are then registered and will be used for the detection. STONE performs an efficient detection but relies only over three traffic features, which is few to manually verify that the detected behaviour is truly anomalous. Our previous detector UNADA [17] or its more recent version ORUNADA [18] are based on clustering. They achieve good detection results and provide the network administrator with automatically created signatures of the malicious traffic. Their autonomy makes them a good detector, but their high computational power requirements does not make them applicable in all situations.

III. AATAC ALGORITHM

In this section we describe a new DDoS detector algorithm called AATAC. AATAC intends to tackle the DDoS detection problem by providing a solution balancing the detection cost with the comprehensiveness of the produced results.

AATAC was first built to limit the work of the network administrator. It is an unsupervised detector assuming that DDoS significantly impact the traffic statistical distributions when occurring. It inherently needs no training data, requires few configuration and autonomously adapts to the traffic shape changes. Moreover, when it raises an alarm, AATAC also provides the network administrator with a set of dynamic 2D plots representing multiple traffic feature distributions or global values. This helps the anomaly diagnostic phase. Finally, and as proved by our evaluation in section IV, AATAC performs an efficient detection producing a low number of false-positives. This avoids wasting the network administrator's time with false alarms.

A. Overview

As illustrated by Figure 1, AATAC processing is split into two components. A first part, the *continuous processing* quickly handles network instances, it updates a set of automatically decreasing *densities*, most of them beign organized as histograms characterizing some traffic feature distribution (IP addresses, ports...). The second part, the *discrete processing*, builds at a regular interval a relevant short-time characterization of the traffic called **traffic snapshot**. Each traffic snapshot is composed of a set F of **snapshot features** that can be divided into two subsets:

$F_{distributions}$ whose features are histogram prototypes, built from the densities organized as histograms and characterizing a traffic feature distribution,

F_{global} whose features use a single counter value, and characterize a global property of the traffic.

The AATAC separation into two parts makes it able to process the traffic in an almost linear time. It makes it robust to sudden

traffic increases and reduces its computational needs when dealing with larger bandwidths.

B. Online processing

The continuous part of the algorithm uses as input a per-flow aggregated data. A flow is usually defined as a 5-tuple: IP source address, IP destination address, source port, destination port, protocol. Each flow is associated with several characteristics such as its average packet size, its number of packets or even its number of SYN packets. Those characteristics are called **flow features**. Each flow should be associated with a timestamp set to either the beginning or the end of the flow.

AATAC uses histograms to model a flow feature distribution. Given a flow feature i, an histogram assigns a probability to each value that i may take within its set of possible values X_i. Each histogram is used to produce a single snapshot feature $f \in F_{distributions}$.

The histogram construction, and its processing, was originally inspired by D-Stream [19], a grid-based stream clustering algorithm. Each instance x added to the model is assigned a **density coefficient** $D(x,t)$ that decreases when x ages. If the instance arrived at a time t_c, its current density coefficient at a time t is:

$$D(x,t) = w_x \lambda^{t-t_c} \tag{1}$$

where $\lambda \in (0,1)$. The w_x variable is a weight assigned to x. λ is a parameter of the algorithm called **decay factor**, which illustrates how fast the density coefficient of an instance decreases over time.

The input space is then divided into a set of equally-sized partitions, also called grids, that can be assimilated to an histogram bins. For each grid g, let us consider the set $E(g,t)$ of all instances that fell into this grid at a given time t. Each grid g is assigned a density value $D(g,t)$ calculated as follow:

$$D(g,t) = \sum\nolimits_{x \in E(g,t)} D(x,t) \tag{2}$$

As the number of instances constantly increases over time, it is not conceivable to store all records to calculate the value of $D(g,t)$. Luckily, this value can be calculated in an incremental fashion: Let us consider a grid g that received a last instance at a time t_l with a corresponding density $D(g,t_l)$. Whenever g receives a next instance at a time $t_n, t_n \geq t_l$, the new density of g can be calculated as follows:

$$D(g,t_n) = \lambda^{t_n - t_l} D(g,t_l) + w_n \tag{3}$$

with w_n the weight of the new instance. Therefore, keeping up-to-date the density of a grid needs only to store two values: a *last update* timestamp t_l and the corresponding density $D(g,t_l)$. The instance weight w_n parameter is set to 1 when considering a per-flow distribution, while multiplied by the number of packets in the flow for a per-packets distribution.

Considering a time $t > t_l$, if g did no received any instance between t and t_l, its density at t is:

$$D(g,t) = \lambda^{t-t_l} D(g,t_l) \tag{4}$$

Similarly, the traffic-wide features (in F_{global}) are processed in a similar manner. A unique density is assigned to each feature which is updated as a grid that would receive all instances. The weight parameter w_n depends here on the feature. For example, the density of the *total SYN packets* feature is weighted using the number of SYN packets in an arriving flow.

C. Offline processing

The discrete processing is applied every ΔT seconds. It is split into three parts: the continuous processing data structure update, the snapshot creation and the anomaly detection.

1) Online data structure update: As the continuous processing data structure is required to be up-to-date when the discrete processing is performed, the grids' densities must be updated. This is done applying equation 4 with $t = t_{snapshot}$, the snapshot creation date. The densities of the traffic-wide features are updated in a similar manner.

To avoid the number of grids in a given feature space to overgrow, AATAC performs a dynamic resource allocation. This implies that grids having a density lower than a given value D_l should be removed from the model. Indeed, these **sparse grids** (having a low density) haven't received instances recently, and are thus no more representative in a picture of the current traffic.

2) Snapshot creation: To reduce the complexity of the anomaly detection phase and keep light the model of the traffic, AATAC records a simplified version of each histogram. Those are called *histogram prototype*, and are made only from grids having a density over a threshold D_m. These grids, said **dense**, are selected because they provide much more information on the traffic than the low-density grids, despite needing as much memory to be characterized. These histogram prototypes are created by gathering adjacent dense grids into clusters. To produce the histogram prototype, our algorithm uses for each cluster: the average density of its grids (avg_c) and its boundaries (min_c and max_c). The histogram prototype, which is basically a piecewise linear curve, includes, for each cluster, the following set of points: $(min_c, 0)$, (min_c, avg_c), (max_c, avg_c) and $(max_c, 0)$. Grids that have a density between D_l and D_m are called **transitional**.

The final snapshot is created storing the produced histogram prototypes (in $F_{distributions}$) along with the current values (updated) of the traffic-wide features densities (in F_{global}). They constitute the set of snapshot features. A snapshot is noted S, and its value for feature f is noted $S(f)$.

3) Anomaly detection: The anomaly detection phase uses the set L_N of the last N snapshots created by the previous phase. AATAC applies a per-snapshot feature k-nearest neighbour (*k-NN*) based technique [20]. S_{last} being the last snapshot added to the model, *k-NN* searches for S_k, its k nearest neighbour in the set L_N considering a feature f. The distance between $S_k(f)$ and $S_{last}(f)$ is an estimation of the local density around S_{last} considering the snapshot feature f. This value is used as an outlier score, illustrating how unlikely is the $S_{last}(f)$ value.

The distance function used for features from F_{global} is the absolute value of the numerical difference (the one-dimensional Euclidean distance). For features into $F_{distributions}$, and because histogram prototypes are basically piecewise linear curves, AATAC uses the area between the two curves as a distance function. The distances between the last N snapshots are stored in an incrementally updated distance matrix.

Finally, AATAC applies a per-feature standard normalization considering the set L_N. The output value is then used as a score, which is compared towards a threshold $\tau_{anomaly}$. If the score goes over this threshold, AATAC considers that the last produced snapshot is anomalous and raises an alarm.

D. Selecting appropriate parameters for AATAC

The decay factor λ parameter characterizes how fast the grid densities decreases over time, it thus impacts the similarity between

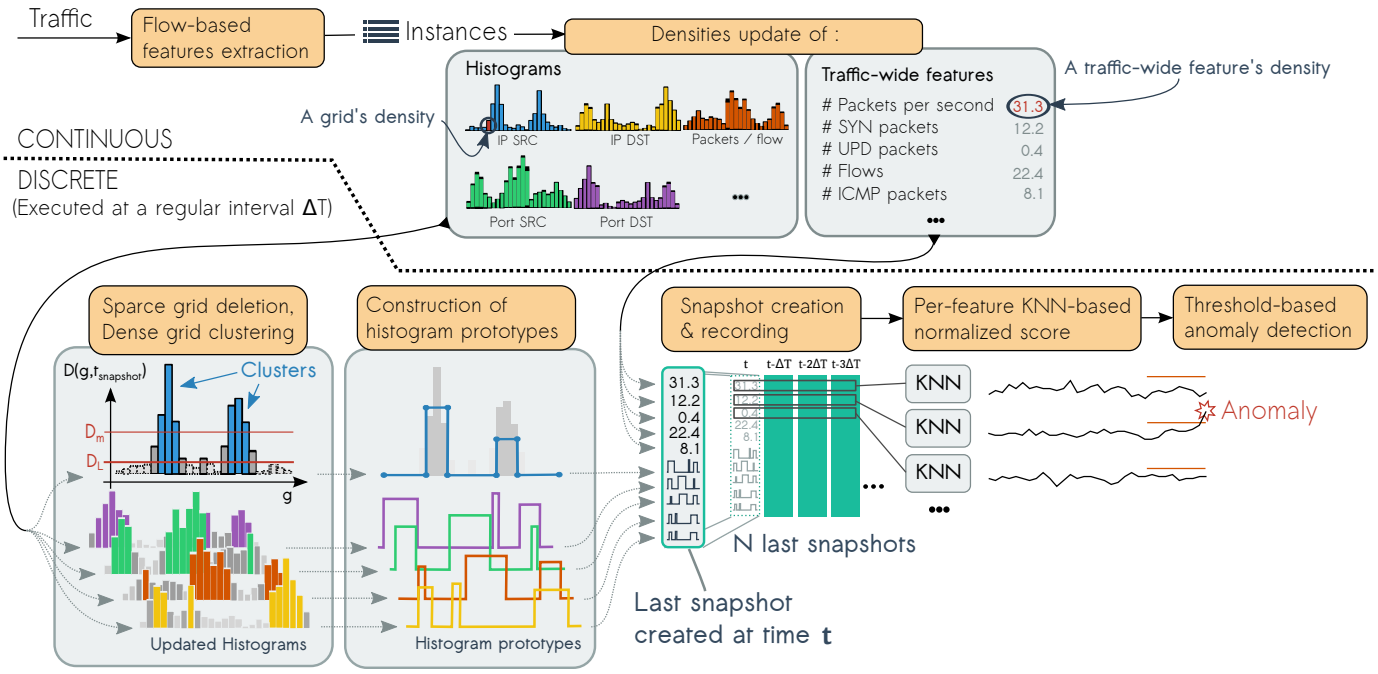

Fig. 1: AATAC architecture

two snapshots: the density of g in a given snapshot is always under the influence of previously stored densities. High values of λ can produce strongly correlated snapshots (reducing the information provided by each snapshot) but low values may also provide a too short-term view of the traffic. We call $R = \lambda^{\Delta T}$ the ratio corresponding to how much the densities of a given snapshot weight in those of the following one. As this ratio is more meaningful than the lambda parameter, it can thus be used to fix lambda with $\lambda = \sqrt[\Delta T]{R}$.

Both R, N and ΔT parameters have a deep impact on the temporal representation of the traffic. While larger values for any of those parameters imply a less sensitive detection to short-term changes into the traffic, those parameters should be balanced considering other criteria. Reducing ΔT reduces the time needed to detect an anomaly, but triggers the discrete treatment more often. Increasing N provides a longer term view of the traffic but increases the computational time needed for the discrete processing. Finally, R acts as an adjustment parameter, impacting how the traffic representation of a snapshot is instantaneous. The best parameters are highly dependant on the monitored traffic, and should be balanced according to the required sensitivity, specificity and computational resources available. These three parameters should be experimentally set before $\tau_{anomaly}$, which is used as a final adjustment to the sensitivity of AATAC.

The D_l and D_m values should by fixed independently from one snapshot feature to another, as they may depend on the weight assigned to added instances. To set them, we use the average density in an histogram while operating on a sample traffic. Indeed, we consider that a density is pertinent enough if it represents, at least, a significant part of the summed densities. As D_l is only used to avoid memory leaks from the multiplication of unused grids, and because densities decreases in an exponential manner, D_l can be chosen very small. Thus, a value around 0.01% of the average density is a good choice. The D_m threshold is more difficult to fix. It is used to select dense enough grids to be stored into the snapshot. We empirically fix this value around 5% of the average density.

As the k parameter (for the *k-NN*-based score), corresponds to the number of nearest neighbours used to estimate a local density, it is thus to be considered along with N, the total number of snapshots kept for the *k-NN*-based analysis. k also impacts the algorithm sensitivity. Thus fixing its value should be done considering previously selected parameters. There is no strict method to fix k but it can be fixed from experimentations on a training dataset. However a common rule of thumb is to fix $k = \frac{1}{2}\sqrt{N}$, which empirically achieved good results.

E. Alarm analysis

An advantage of AATAC is its capability to produce a graphical representation of each snapshot. Indeed, histogram prototypes can be plotted as curves, and global densities as a simple dot. Thus, whenever AATAC generates an alarm, the last N snapshots can be plotted as a sequence of graphs, providing a dynamic view of the traffic properties when an anomaly occurs. They can be plotted along with the outlier score to give a better understanding of the occurring anomaly. During our evaluation, we were able to produce videos of the traffic anomalies [21].

IV. Performance analysis

To evaluate its performances, we implemented AATAC in C. It uses the *libpcap* [22] capture format as input and performs a flow extraction using tumbling window. The timestamp associated to each flow is set to the date of each window end. Experimentations are run over a single machine powered by a 3.00GHz Intel Xeon CPU (E5-2623 v3). It features 8 cores (16 with hyper-threading), but our implementation does not fully benefit from this feature. Indeed, only the discrete part of AATAC is per-dimension parallelized.

Our evaluation discusses AATAC properties in terms of detection accuracy and computational resource consumption, depending on its parameters. Obtained results are then compared with FastNetMon and ORUNADA.

A. Detection accuracy

1) Evaluation methodology: To evaluate our algorithm ability to efficiently detect DDoS attacks, we needed a labelled dataset

containing various types of DDoS attacks within realistic and up-to-date traces. Unfortunately, we were not able to find a publicly available dataset meeting those criteria. Thus, in the context of the ONTIC project, we created a set of 13 synthetic attacks in an emulated network. Each of these attacks was inserted in a 1 hour long subset of the ONTS dataset, which consists in five months of anonymized and payload-free traces, captured at the entrance of a large cloud service provider. These traces are publicly available on the ONTIC project website [23]. The dataset including the generated attacks is called synthONTS, it is still a work in progress as it should be completed with other attacks.

To perform the evaluation, we run AATAC over each trace included into the synthONTS dataset. This evaluation over labelled traces allows an estimation of several characteristics of the detector, in terms of accuracy, such as the *true-positive rate* (TPR, the probability that there is a real anomaly when the detector raises an alarm) and the *false-positive rate* (FPR, the probability that the detector raises an alarm while there is no anomaly).

The most common tool to evaluate an intrusion detection system accuracy is the Receiver Operating Characteristic (ROC) curve. This curve plots the TPR against the corresponding FPR for a given IDS. Thus, an ideal detector should be plotted at $(0, 1)$. AATAC's ROC curves are plotted varying the detection threshold $\tau_{anomaly}$. To evaluate a detector operation point independently from the threshold value, the Area Under the ROC Curve (AUC) is used. A perfect detector has $AUC = 1$ while a random one has $AUC = 0.5$.

Despite providing interesting and readable results, ROC curves applied to synthONTS suffer from the *base-rate fallacy* [24]. This is a common bias that happens when considering such rates without considering the base-rate, i.e. the probability that an anomalous event occurs. To overcome this problem, we complete our ROC-based evaluation with another evaluation method from Nasr et al. [25]. This method uses the *positive predictive value* (also known as *Bayesian Detection rate*) which corresponds to the probability of an intrusion given that the IDS raised an alarms. From the result of an experimentation, it can be computed as follows:

$$PPV = \frac{Number\ of\ true\text{-}positives}{Number\ of\ positives} \tag{5}$$

This value is plotted against the corresponding false-positive rate, constituting the *actual IDS operation curve*. This curve is compared to the *zero reference curve (ZRC)* which corresponds to the trade-off between PPV and the false-positive rate FPR. Concretely, the various PPV values for the ZRC are calculated considering a detector that detects all anomalies ($TPR = 1$), but that generates an increasing number of false-positives. As the PPV calculation encompasses the base-rate, this method does not suffer from the base-rate fallacy.

The accuracy of the detector is finally estimated using the detector *intrusion detection effectiveness* ($E_{ID} \in [0, 1]$), which is the normalized variance between the actual IDS operation curve and the ZRC. It is calculated as follows:

$$E_{ID} = \frac{1}{\int_0^{T_{FP}} PPV_{ZRC}\,\mathrm{d}\alpha}\left(\int_0^{T_{FP}} PPV_{ZRC}\,\mathrm{d}\alpha - \int_0^{T_{FP}} PPV_{ID}\,\mathrm{d}\alpha\right) \tag{6}$$

Where T_{FP} is the maximum acceptable false-positive rate exhibited by the IDS. The lower E_{ID}, the more effective the detector.

2) Evaluation results: In our experimentations, we tested several values for R: 1%, 2%, 5%, 10%, 50% and 90%, and several values for N: 100, 500, 1000. As the traces are quite short, ΔT is set to 1 second to have enough recorded snapshots to train AATAC. The different ROC curves obtained are pictured in Figure 2. Despite that it seems that we obtain better results with lower values of R, they tend to produce a lot of false-positives. However, as we can see from the different operation points, the best point appears to be with $R = 0.90$, $N = 500$ and a threshold $\tau_{anomaly} = 3.0$. For this operation point, we have $TPR = 0.83$ and $FPR = 0.0013$. While the TPR is relatively low, due to the fact that the dataset contains a low number of anomalies, the FPR in this situation is very good.

R \ N	100	500	1000
0.01	0.9211	0.9668	0.9722
0.02	0.9202	0.9666	0.9721
0.05	0.9197	0.9598	0.9716
0.10	0.9181	0.9595	0.9717
0.50	0.8860	0.9487	0.9639
0.90	0.9083	0.9445	0.9443

TABLE I: AUC for multiple values for R and N

The AUC values, as depicted in table I, confirms those preliminary results: the higher values for AUC correspond to the lowest values for R and the highest for N. However, the IDS operation curve (for $N = 500$) depicted by Figure 3 shows different results than the ROC curves. Indeed, the base-rate fallacy is removed from this evaluation, showing that, considering a low enough false-positive rate, a higher value of R corresponds to more efficient detection. The calculated instruction detection efficiency is depicted in Table II (values where $E_{ID} = 1.0$ stands for conditions for which there were no valid PPV for any $FPR < T_{FP}$). For any $T_{FP} \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, AATAC appears more efficient with higher values of R. Increasing N still improves the detection efficiency.

Needing $R = 90\%$ is quite high (as a reminder, this means that a snapshot includes 90% from the previous one in its densities). However, this can be explained by the fact that we used a short snapshot interval time ΔT. Indeed, this large value of R makes the detector less sensitive to short-term variations, which reduces the number of generated false-positives.

B. Real time

To estimate the capability of AATAC to process the traffic in real-time, we ran another evaluation. In the context of the Border 6/LAAS-CNRS project, we had the opportunity to capture the traffic of a company whose main activity consists in hosting online sales websites. The traffic is 3 Gbit/s and 440.000 pkt/s average. This evaluation was ran over one day of network traces containing the beginning of a DDoS attack (the rest was removed by the mitigation system of the company). As they are the only parameters that may significantly impact the processing time, we chose to vary only $\Delta T \in \{1s, 5s, 10s, 30s, 1min\}$ and $N \in \{100, 1000, 5000\}$ for our evaluation.

As we run our experimentations from recorded traffic, we use the processing time as our evaluation measure. The continuous and discrete part of the algorithm are distinctly considered:

- For the continuous part we measure the ratio between the processing time over the corresponding traces interval treated,
- For the discrete time we measure the time needed to produce a single snapshot, independently of how many of them were created.

The results with 18 features are depicted in table III. The values are averaged within the different value of ΔT. From those results we can see that even with relatively low computer resources, our

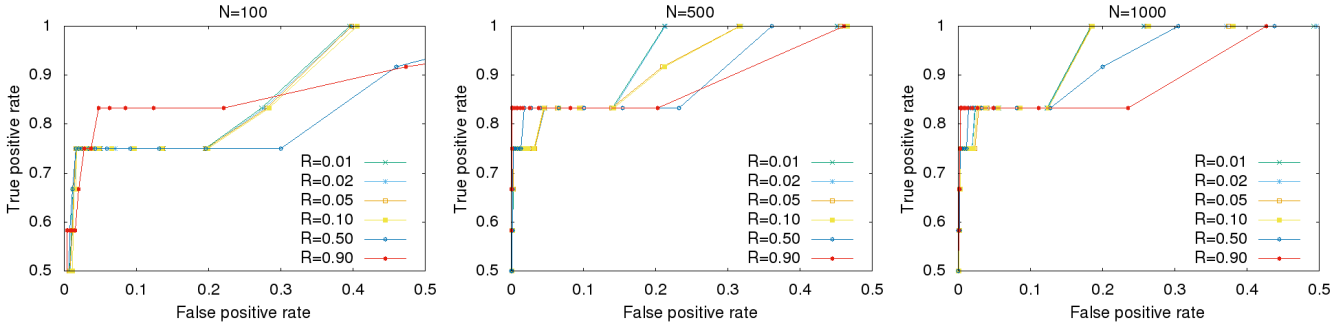

Fig. 2: ROC curves for multiple values of N and R

T_{FP}	10^{-2}			10^{-3}			10^{-4}		
R \ N	100	500	1000	100	500	1000	100	500	1000
0.01	0.4286	0.1627	0.1579	0.5520	0.1221	0.1054	1.0	0.0200	0.0163
0.02	0.4794	0.1628	0.1570	0.7772	0.1217	0.1011	1.0	0.0201	0.0183
0.05	0.4278	0.1616	0.1587	0.5602	0.1174	0.1088	1.0	0.0201	0.0185
0.10	0.4414	0.1599	0.1593	0.5602	0.1140	0.1073	1.0	0.0209	0.0178
0.50	0.4372	0.1537	0.1465	0.5930	0.0989	0.0796	1.0	0.0218	0.0205
0.90	0.4636	0.0790	0.1114	0.6973	0.0586	0.0843	1.0	0.0158	0.0216

TABLE II: Intrusion Detection Effectiveness for multiple values of N, T_{FP} and R

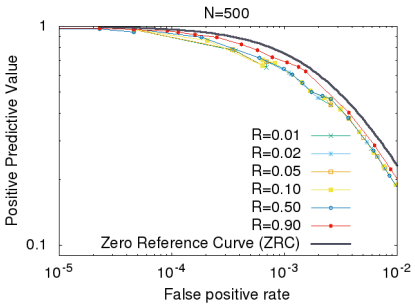

Fig. 3: IDS operation curves for $N = 500$ and several R values

N	100	1000	5000
Continuous (maximum for 1s of trafic)	0.80	0.79	0.89
Continuous (average for 1s of trafic)	0.21	0.25	0.26
Discrete	0.028	0.027	0.033

TABLE III: Processing time in seconds

implementation of AATAC is able to handle the traffic in real time. Even in stress situations, the detector still takes less than one second to treat one second of traffic. Thanks to the simplicity of the discrete processing, the time needed to produce a snapshot and compare it to the N last produced ones appears negligible (around $30\mu s$), independently of N.

C. Comparative evaluation with other detectors

For a fair evaluation of AATAC, we compared the obtained results over synthONTS with the best performances obtained by two other detectors: FastNetMon [3] and ORUNADA [18]. While those detectors are respectively representative of knowledge-based and unsupervised detectors, they were also available to us.

In terms of detection accuracy, ORUNADA in its best operation point is able to detect all DDoS attacks included into the synthONTS dataset, but also detects other kinds of anomalies that were already present in the original traces. However, in this configuration, ORUNADA needs a lot of resources to perform the detection: it needs 220% of one CPU computing power to operate in real time.

FastNetMon manages to operate a real time detection on the traffic with two cores at 60% load. In its best operating point, it detects 6 out of 13 DDoS attacks, and produces one false-positive.

In its best operation point, mentioned earlier, AATAC can operate at 21% load on a single core. Those results confirms that AATAC can operate with few resources, while still providing an accurate detection.

V. Conclusion and Future Works

In this paper we introduced AATAC, a new anomaly detector that focuses on DDoS attacks. We proposed a detector divided into two components, one that processes the traffic in linear time while a second one performs the traffic analysis and anomaly detection. As showed in our evaluation, the several algorithmic optimizations make AATAC able to process actual traffic in real time with few computational resources. That being said, AATAC still performs an efficient detection, producing a low number of false-positives. Also, AATAC provides the administrator with pertinent information on the detected anomalies. Its unsupervised nature makes it an autonomous detector, needing little configuration and maintenance to be operational.

The short duration of the discrete treatment makes us consider running a more complex analysis over the snapshots to provide the administrator with more information over detected anomalies. This treatment could include a correlation analysis of the several features scores. A final goal should be to suggest the network administrator with automatically generated filtering rules.

As a request from our collaborators from Border 6 and regarding implementation concerns, we also consider evaluating AATAC over sampled traces. We plan to evaluate the impact of sampling on both the computational resource consumption and the detection accuracy.

Acknowledgements

We would like to thank Border 6 for their collaboration, and Kering for allowing us capturing their traces for the purpose of our evaluation.

References

[1] T. Gil and M. Poletto, "MULTOPS: a data-structure for bandwidth attack detection," in *Proc. 10th USENIX Secur. Symp.*, 2001.

[2] J. Wang, R. C. Phan, J. N. Whitley, and D. J. Parish, "Augmented Attack Tree Modeling of Distributed Denial of Services and Tree Based Attack Detection Method," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol. (CIT 2010)*, pp. 1009–1014, 2010.

[3] "Fastnetmon." https://fastnetmon.com/.

[4] J. Udhayan and T. Hamsapriya, "Statistical Segregation Method to Minimize the False Detections During DDoS Attacks," *Int. J. Netw. Secur.*, vol. 13, no. 3, pp. 152–160, 2011.

[5] J. David and C. Thomas, "DDoS Attack Detection Using Fast Entropy Approach on Flow-Based Network Traffic," in *2nd Int. Symp. Big Data Cloud Comput.*, vol. 50, pp. 30–36, Elsevier Masson SAS, 2015.

[6] G. No and I. Ra, "Adaptive DDoS detector design using fast entropy computation method," *2011 5th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput.*, pp. 86–93, 2011.

[7] I. Ozçelik and R. R. Brooks, "Cusum - Entropy: An efficient method for DDoS attack detection," in *2016 4th Int. Istanbul Smart Grid Congr. Fair*, pp. 1–5, 2016.

[8] B. B. Gupta, R. C. Joshi, and M. Misra, "ANN Based Scheme to Predict Number of Zombies in a DDoS Attack," *Int. J. Netw. Secur.*, vol. 14, no. 2, pp. 61–70, 2012.

[9] J. Cheng, J. Yin, Y. Liu, Z. Cai, and C. Wu, "DDoS Attack Detection Using IP Address Feature Interaction," in *2009 Int. Conf. Intell. Netw. Collab. Syst.*, pp. 113–118, 2009.

[10] Y.-c. Wu, H.-R. Tseng, W. Yang, and R.-H. Jan, "DDoS detection and traceback with decision tree and grey relational analysis," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 7, no. 2, 2011.

[11] R. Jalili, F. Imani-mehr, M. Amini, and H. R. Shahriari, "Detection of Distributed Denial of Service Attacks Using Statistical Pre-processor and Unsupervised Neural Networks," in *Inf. Secur. Pract. Exp. First Int. Conf. ISPEC 2005*, pp. 192–203, 2005.

[12] R. Karimazad and A. Faraahi, "An Anomaly-Based Method for DDoS Attacks Detection using RBF Neural Networks," in *2011 Int. Conf. Netw. Electron. Eng.*, vol. 11, pp. 44–48, 2011.

[13] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016.

[14] A. O. Adetunmbi, S. O. Falaki, O. S. Adewale, and B. K. Alese, "Network Intrusion Detection Based on Rough Set and K-Nearest Neighbour," *Int. J. Comput. ICT Res.*, vol. 2, no. 1, pp. 60–66, 2008.

[15] R. Zhong and G. Yue, "DDoS Detection System Based on Data Mining," in *Proc. Second Int. Symp. Netw. Netw. Secur. (ISNNS '10)*, vol. 1, pp. 62–65, 2010.

[16] V. Gulisano, M. Callau-zori, Z. Fu, R. Jiménez-Peris, M. Papatriantafilou, and M. Patiño-Martínez, "STONE: A streaming DDoS defense framework," *Expert Syst. Appl.*, vol. 42, no. 24, pp. 9620–9633, 2015.

[17] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge," *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, 2012.

[18] J. Dromard and P. Owezarski, "Online and Scalable Unsupervised Network Anomaly Detection Method," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 1, pp. 34–47, 2017.

[19] 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.

[20] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 427–438, 2000.

[21] "Aatac visualization tool video." https://youtu.be/unqQmaBnlHc.

[22] "Libpcap file format." http://wiki.wireshark.org/Development/LibpcapFileFormat. Accessed: 2017-05-10.

[23] "Ontic." http://ict-ontic.eu/. Accessed: 2017-05-12.

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

[25] 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.