2023 IEEE Latin-American Conference on Communications (LATINCOM) | 979-8-3503-2687-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/LATINCOM59467.2023.10361881

A Runtime DDoS Attack Detection Technique Based on Stochastic Mathematical Model

Euclides Peres Farias Jr.[*], **Allainn Christiam Jacinto Tavares**[†], **Michele Nogueira**[*†]
[*]Department of Informatics - Federal University of Paraná, Brazil
[†]Department of Computer Science - Federal University of Minas Gerais, Brazil
Emails: epfjunior@inf.ufpr.br, allainncjt@gmail.com, michele@dcc.ufmg.br

***Abstract*—Distributed Denial of Service (DDoS) attacks are increasingly prevalent, targeting various entities. Detecting DDoS attacks is still an evolving and open challenge, despite considerable efforts. Existing solutions, including those employing artificial intelligence techniques, require significant computational resources and present limitations in handling real-time data. Hence, this paper presents a novel technique founded on a stochastic model to detect DDoS attacks during runtime. For evaluation, the technique focuses on SYN flood DDoS attack, and it has been implemented in a software-defined network given its programmability feature. Results have compared the proposed technique to representative ones from the literature, as Fuzzy Logic, MLP Neural Network, and Shannon Entropy. The new technique outperforms the other methods, opening up possibilities for application in different scenarios.**

***Index Terms*—DDoS attacks, Network Security, AI**

I. Introduction

There has been a 38% increase in global cyberattacks in recent years [1]. Distributed Denial of Service (DDoS) attacks have been particularly prominent. According to the DDoS threat intelligence report, the first half of 2022 has witnessed a staggering 6 million DDoS attacks worldwide [2]. Unfortunately, DDoS attacks have progressively grown in severity over the years, with records of attacks surpassing the 10 million mark per month. Hence, the efficient and runtime detection of these attacks is of utmost importance to safeguard networks and ensure uninterrupted service.

Research focusing on the development of real-time or runtime DDoS attack detection is essential for detecting current DDoS attacks and mitigating their impacts [3]. Despite the efforts and work on detecting attacks at runtime, designing techniques, models and methods for effectively addressing the runtime aspect is a challenging task [3] given the large amount of data generated by DDoS attacks in a very small-time unit (*e.g.*, second) [12]. However, approaches to detect DDoS attacks on runtime are of utmost relevance and needed in the face of the advent of distributed computation, Internet of Things IoT, and cloud environments [4].

The literature is rich in detecting DDoS attacks mechanisms, particularly, SYN flooding attacks [3], [4]. Around 46% of the DDoS attacks exploit the TCP traffic features, as revealed by the IS2022 report [5]. SYN Flood attacks stand out as one of the most aggressive and extensively studied [3]. However, considerable challenges are posed by the implementation of detection mechanisms, as there is a need to collect and process large volumes of real-time network traffic data, the complexity of identifying and classifying malicious traffic patterns amidst significant benign traffic, and the necessity to ensure that attack detection does not negatively impact network performance [7], [8]. Nugraha and Murthy [6] highlighted the use of several approaches for detecting DDoS attacks in real-time or runtime, such as Machine Learning. Other approaches involve signature-based intrusion detection techniques, which compare real-time network traffic with a set of known attack signatures. In [4], a complex event processing technique analyzes real-time network traffic and detects anomalies. Bidi-Directional Long Short-Term Memory Recurrent Neural Networks have been employed to detect botnet formation in real-time on IoT devices. These techniques require preprocessing efforts, as they need training models to identify traffic patterns of an ongoing DDoS attack.

This work presents DetectAMath, a simple, fast, efficient, and real-time technique for DDoS attack detection. This technique was initially created and developed inspired by the use of annotated paraconsistent logic (LPA) [16]. This way, DetectAMath focuses on detecting SYN Flood attacks, extensible to other DDoS attacks. DetectAMath preprocesses TCP/IP packets, taking as input the existing flags to determine a score within the range of 0 to 1, called the degree of belief, necessary to identify a potential attack. DetectAMath compares the average degree of belief to a threshold to indicate or not the attack detection. The technique provides real-time attack detection, effectively identifying SYN Flood DDoS attacks. The strengths of this approach lie in its simplicity, speed, and efficiency, making it highly practical for rapid implementation and effective attack detection in execution-time scenarios.

The evaluation methodology employs a Software-Defined Network (SDN) infrastructure, given its programmability feature. The DetectAMath technique, as well as, Shannon Entropy, Fuzzy Logic, and KNN [9], has been implemented and tested in this environment under the evaluation metrics accuracy, precision, recall, F1-score and processing time. Through this approach, real-tine has been recorded during runtime attack detection.

This paper proceeds as follows. Section II presents the related works. Section III describes the proposed technique. Section IV details the evaluation methodology and results, and finally section V concludes the work.

II. Related Work

There are several methods of DDoS attack detection in the literature. The Shannon Entropy (SE) is used to construct feature vectors, which are subsequently used to separate DDoS and Flash Events traffic from normal network traffic in a time- and cost-efficient manner. Additionally, Shannon entropy is used to differentiate DDoS attacks from Flash Events traffic [10]. Shannon's entropy is promising compared to existing methods for DDoS attack detection, as stated by [12]. The online detection system proposed by [12] was presented as a solution to identify the presence of DDoS flood attacks and detect attack connections in real-time, based on entropy and dynamic thresholding. The proposed technique involves the calculation of normalized Shannon Entropy to detect suspicious activities, while the threshold is dynamically updated based on the evolution of legitimate traffic. However, it is observed that the detection time achieved by the proposed system was 7.89 seconds when identifying DDoS flood attacks in a client-server environment. This result does not appear to be satisfactory for a real-time application.

According to [3], SYN Flood attacks are common and can overwhelm a host with numerous SYN packets, impairing processing and memory, thus affecting the availability and integrity of services. Therefore, in [3], the authors proposed a supervised machine learning-based and multi-objective optimization approach to detect and mitigate these attacks. The approach involves real-time traffic data collection, extraction of relevant features, and generation of attack signatures. Machine learning models are trained using these signatures to classify SYN Flood attacks in real-time. Multi-objective optimization techniques are also employed to reduce the number of filtering rules and minimize the impact on network performance. Other relevant works applying machine learning have utilized various deep learning classifiers, such as CNN, LSTM, SVC-SOM, MLP, and I-Class SVM, all applied in real-time [6], [13].

Other techniques for detecting DDoS attacks include the application of Fuzzy Logic [14]. In [14], the authors discusses the use of fuzzy logic in detecting network traffic anomalies and preventing DDoS attacks. It presents a systematic literature review and comprehensive survey of recent advances in the area of anomaly-based intrusion detection systems that employ fuzzy logic to correlate network behavioral features. The article also proposes a new hierarchy for categorizing various types of DDoS attacks based on their internal mechanism and technology. The results indicate that fuzzy logic can be effectively incorporated into a wide range of network anomaly detection schemes as a highly reliable solution to increase intrusion detection accuracy while maintaining network performance.

In [15], the authors presented a new algorithm for analyzing network traffic in real time. Several variants were considered to the implementation of intelligent agents, based on the algorithm for detecting pseudogradient anomalies and on the inference of fuzzy logic. The authors mention that this technique can be applied in any computational/hardware environment. However, the computation in IP-kernels requires floating point, which is supported at the hardware level of the chip, which suggests a potentially relatively high computational complexity for this technique.

III. DetectAMath: A Technique for DDoS Attack Detection

This section presents DetectAMath, a technique funded on a stochastic mathematical model that assesses the occurrence of SYN Flood DDoS attacks. The technique computes a decision degree known as the "Degree of Belief" (DoB), which means the probability of an attack occurrence. DoB is derived from processed data regarding the number of packets with SYN, FIN, SYN_ACK, and ACK flags, resulting in a value ranging between 0 and 1, which represents how acceptable the condition of occurrence of an attack is. As a guide to the model, Table (I) presents the description of each element of the equations.

TABLE I

Symbol	Description
A	Number of packages with *flags* ACK
S	Number of packages with *flags* SYN
F	Number of packages with *flags* FIN
SA	Number of packages with *flags* SYN_ACK
R	Number of packages with *flags* RST
P	Number of packages with *flags* PSH
$d-2$	Delimiter of decimal place amount

The mathematical model supporting the proposed technique follows five phases. Each phase comprises a set of equations. The partial models for each phase follow.

Phase I

In Eq. (1), the condition '0 if $S+SA=0$' intends to block division by zero. In '$1-\frac{F}{S+SA}$ if $S+SA\neq 0$' it has the purpose of verifying the balance between the number of packets with the FIN flag and the number of packets with SYN flag in a window time of 10 seconds. The division of F by the sum of $S+AS$ results in the belief that there is an attack. Eq. (2) normalizes the value in the interval between [0 and 1].

$$y_a = \begin{cases} 0 & \text{if } S+SA=0 \\ 1-\frac{F}{S+SA} & \text{if } S+SA\neq 0 \end{cases} \tag{1}$$

$$\mu_a = \begin{cases} 0 & \text{if } y_a<0 \\ y_a & \text{if } 0\leq y_a\leq 1 \end{cases} \tag{2}$$

Phase II

In this phase, Eq. (3) '0 if $S=0$' has the function to check if the number of packets with only the SYN flag active is different from zero. As for the non-zero condition, y_b receives the ratio of the number of packets that have the SYN and ACK flag active with packets that only have the SYN active. This ratio validates whether there was an equivalent amount between the client's connection opening requests and the server's confirmation responses. Eq. (4) normalizes the values in the interval between [0 and 1].

$$y_b = \begin{cases} 0 & \text{if } S = 0 \\ 1 - \frac{SA}{S} & \text{if } S \neq 0 \end{cases} \quad (3)$$

$$\mu_b = \begin{cases} 0 & \text{if } y_b < 0 \\ y_b & \text{if } 0 \leq y_b \leq 1 \end{cases} \quad (4)$$

Phase III

In Eq. (5), x represents the number of digits of the value of the number of TCP packets. In Eq. (6) "d" is the variable employed in Eq. (7) to ensure that the value is greater than or equal to the value 2. This verification is done by the condition that if the value obtained by Eq. (5) is less than or equal to 2, then d receives 2. If the value is greater than or equal to 2, then "d" receives the value obtained without changes. In Eq. (7), the condition '0 if $S = 0$' ensures that the value will be equal to zero, as negative values are not acceptable. In the condition '$1 - \frac{\frac{A}{S}}{10^{d-2}}$ if $S \neq 0$', when using the factor $d - 2$ in its composition, a limit is thus created. Hence, Eq. (7) guarantees that the value of subtraction is not negative. The term $\frac{A}{S}$ validates if the value of the number of SYN packets is different from zero to avoid division by zero. If SYN he is different from zero (0), an association is made between the number of packets that have the ACK flag active and the number of packets that have only the SYN flag active. This relationship is associated with requests for opening connections, exchanging messages and already established connections. In order to the value to behave close to the interval [0, 1], the result of the division A/S is divided by 10^{d-2} is performed. This value is validated in Eq. (8) to ensure that the received value is within the interval between [0 and 1], which will be the attack state or not.

$$x = \text{Number of value digits } TCP \quad (5)$$

$$d = \begin{cases} 2 & \text{if } x < 2 \\ x & \text{if } x \geq 2 \end{cases} \quad (6)$$

$$y_c = \begin{cases} 0 & \text{if } S = 0 \\ 1 - \frac{\frac{A}{S}}{10^{d-2}} & \text{if } S \neq 0 \end{cases} \quad (7)$$

$$\mu_c = \begin{cases} 0 & \text{if } y_c < 0 \\ y_c & \text{if } 0 \leq y_c \leq 1 \end{cases} \quad (8)$$

Phase IV

Eq. (9) helps to calculate DoB, as it verifies is the divisor (which calculates information within the TCP protocol through *flags*) is different from zero. This avoids division by zero. If the divisor value is zero, the result is directly assigned as zero. Otherwise, the equation performs the mathematical operation that relates the number of packets that have the *flags* ACK, FIN, RST and PSH active with the number of packets TCP. This relation is necessary to verify if there was reduced traffic of packets that have the *flag* SYN active in relation to the amount of TCP packets.

$$\mu_d = \begin{cases} 0 & \text{if } TCP = 0 \\ 1 - \frac{A+F+R+P}{TCP} & \text{if } TCP \neq 0 \end{cases} \quad (9)$$

Phase V

Eq. (10) calculates the average x among the DoBs (μ_a, μ_b, μ_c, μ_d). Then, Eq. (11) validates whether the result of Eq. (10) exceeds the limit of a given value that is the reference for the greater or equal attack condition to 0.65 (balance point regarding the quality of accuracy compared to FP and FN). If you exceed the limit, the system considers the occurrence of an attack; otherwise, there is no attack. This happens because the conditional framework establishes a decision criterion based on average values and a specific threshold to determine whether something is considered an attack or not.

$$x = average(\mu_a, \mu_b, \mu_c, \mu_d) \quad (10)$$

$$predict = \begin{cases} 0 & \text{if } x \leq threshold \\ 1 & \text{if } x > threshold \end{cases} \quad (11)$$

The mathematical model supporting DetectAMath was developed exclusively for the detection of DDoS attacks of the *SYN Flood* type. Through the three-way TCP *handshake*, it performs a complete verification in the collection and analysis of synchronization packets (SYN) to the server, as well as analysis of synchronization confirmation packets (SYN_ACK) to the client, analysis of resets that occurred during transmission (RST). Thus, this is the complete analysis of checking these flags within the TCP/IP protocol.

In this way, the algorithmic analysis of the proposed technique it divided into two complexities: preprocessing, which involves counting the flags, has a complexity of O(n). In this context, (n) refers to the number of packets in the window. In phases, I, II, III, IV, and V, all have a complexity of O(n log(n)). In this case, (n) represents the number of equations applied in the technique.

IV. Performance Evaluation

This section describes the applied evaluation methodology, results and discussion of results achieved through the use of the proposed DDoS detection technique.

A. Methodology

The proposed technique was constructed and evaluated by the following items: virtualized environment (VirtualBox); Linux Ubuntu 20.04 LTS operating system; Anaconda environment with Python version 3.9; SDN network emulator (Mininet) with RYU controller; packet manipulation tool for networks (Scapy); penetration testing tools T50 and HPING3. The network topology comprises an SDN controller (C0) RYU, an intermediate switch (SW1) and five edge switches (s1, ..., s5), each one containing ten (10) hosts. The first host (h11) of the first switch (s1) serves as the HTTP server, the attack target. The controller (C0) is directly connected to SW1, which in turn is connected to all five edge switches, which thus connect their respective hosts.

Three more algorithms capable of detecting SYN Flood DDoS attacks were implemented. The first implemented algorithm was the Fuzzy Logic algorithm, used to construct the Fuzzy system. It received four input sets for analysis, represented by Eqs. (2), (4), (8), and (9), as Fuzzy Logic requires evaluating the DoB condition and ensuring the threshold is between [0, 1]. These input sets were also divided into three terms: low, medium, and high. Therefore, membership functions were employed within each set, defined as trapezoidal and triangular. The trapezoidal functions represent the low and high terms, while the medium term is represented by the triangular function. The points on the Cartesian plane for the low membership function are: $A = (0, 0), B = (0, 1), C = (1 - N_exig, 1)$, and $D = (0.5, 0)$. The points for the medium function are: $A = (1 - N_exig, 0), B = (0.5, 1)$, and $C = (N_exig, 0)$. Finally, the points for the high function are: $A = (0.5, 0), B = (N_exig, 1), C = (1, 1), andD = (1, 0)$. The points are connected by straight lines within each factor, following the presented order.

The implementation of Shannon's Entropy algorithm involved collecting all standard flags. This collection was carried out by capturing the flags, followed by the calculation of the probability separated by flag. The entropy equations (Eqs. 12, 13 and 14) were applied to the data in TCP protocol windows of 10 seconds, returning the value after applying threshold. The results were recorded in logs, both in CSV files and console output, depending on the options selected by the user through the corresponding switch. The code implements a real-time detection system for SYN Flooding DDoS attacks by capturing network packets, performing preprocessing, analyzing the data, and displaying the results.

$$H = -\sum_{i=1}^{n} p_i \; log \; p_i \tag{12}$$

$$W = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\} \tag{13}$$

$$p_i = \frac{x_i}{n} \tag{14}$$

The Multilayer Perceptron (MLP) Neural Network, also known as Multilayer Perceptron [16], was used in this study as another reference technique to validate the DetectAMath technique. Thus, the neural network was configured with 8 nodes in the input layer, 16 nodes in the first hidden layer, 32 nodes in the second hidden layer, and 2 nodes in the output layer. The activation function used in the hidden layers was the Relu, while the activation function in the output layer was Softmax, the optimizer was Adam, and the loss function used was categorical crossentropy. Due to the collection of packets for all DetectAMath, Fuzzy, Entropy and MLP Network techniques, it was decided to carry out the training in 10-second windows, enough time so that all techniques are not compromised in the analysis of data collection.

Hence, the input data collection for the MLP was performed from the TCP protocol transmission packets with 10-second windows. During this collection, the following flags were considered: SYN, SYN_ACK, FIN, ACK, RST and PSH. Then, metrics were applied as the total number of TCP packets, the number of packets with only the flag SYN active, the number of packets with the flags SYN and ACK enabled, the number of packets with just the flag ACK enabled, the number of packets with only the flag FIN enabled, the number of packets with just the flag RST active, the number of packets with only the flag PSH active. This task was necessary to obtain the results from the MLP technique.

For the entire experimental environment, shell scripts have facilitated the management of the overall infrastructure, following a specific initialization order. First, the SDN network controller, RYU, was launched as a prerequisite for the network controller. Next, the network emulation environment, Mininet, was started, ensuring proper connectivity and control within the proposed topology. In order to enhance operational control, all actions were continuously monitored through dedicated scripts. Also, an attack script was developed to provide the option of launching attacks using T50 or hping3 against the HTTP web server on port 80, referred to as "h11". From this particular host, other hosts were accessed using xterm, enabling the execution of the attack script and facilitating the collection of data extracted by the algorithms.

In order to create a network monitoring environment in the application of DetectAMath, Fuzzy, MLP and Entropia techniques, a shell script (named monitora_ataques.sh) was created. This script is made up of functions that allow you to choose the port on the distribution switch, which connects the 05 switches of the topology and checks which ones are active ($sw\_int-eth1$, $sw\_int-eth2$, $sw\_int-eth3$, $sw\_int-eth4$, $sw\_int-eth5$). From the chosen switch interface, it is possible to select which technique you want to monitor in the SDN network (1 - DetectAMath, 2 - Fuzzy, 3 - MLP and 4 - Entropy). From this action, the chosen technique is executed and all of them have an area to store logs so that the analysis of the data resulting from the execution can be performed.

For the sake of consistency and minimize interference, the practice of restarting the Virtual Machine (VM) prior to executing each algorithm (DetectAMath, Fuzzy, MLP, and Entropy) was adopted for data collection. This approach aims to establish similar execution conditions for each technique by monitoring all running processes and mitigating potential interference during execution and data collection. The process measurements, along with the load average reported by the operating system and the number of processes running in the last 1, 5, and 15 minutes, should be below zero during the runtime. The monitoring script guarantees that, for each monitored technique, the attack process and data collection are carried out on each identified interface during the execution of the SDN network ($sw_int-eth1$, $sw_int-eth2$, $sw_int-eth3$, $sw_int-eth4$, and $sw_int-eth5$).

Then, a time period of up to 10 minutes was defined for the application of each technique in each interface. This strat-

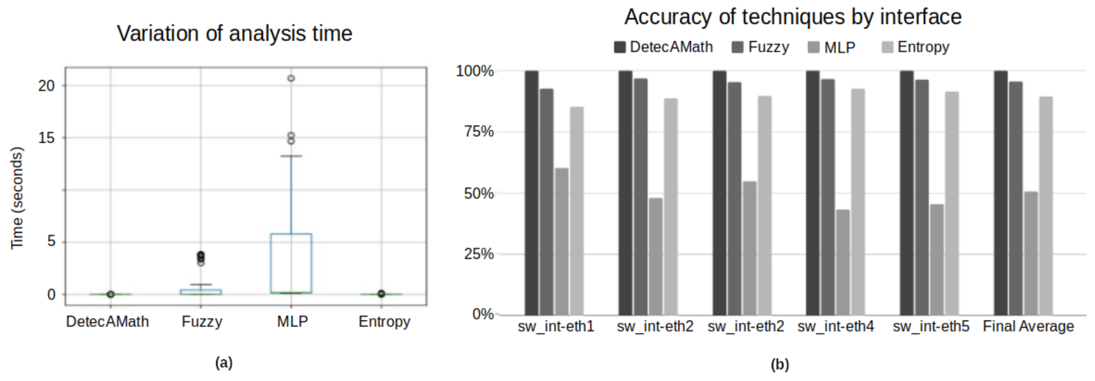

Fig. 1. **Analysis a)** Acuracy of the thechniques by interfaces and **Analysis b)** Variation of analysis time

egy allowed establishing attacks based on the hosts directly connected to each switch, considering their IP address ranges. At the end of each 10-minute period, during the execution of a technique on an interface, it was necessary to wait for all operating system indicators to reach significant values, such as running processes, ensuring that the environment had been the target of attacks and that monitored mean load values below zero are observed. This action was repeated for all interfaces, ensuring that all were covered by the chosen technique. When moving to the next technique, the operating system needed to be restarted (rebooted), ensuring that the measures of execution time and resource availability were the same for all techniques.

The infrastructure allows the management of the topology, allowing the detection and control of all flows from an IP address, since it is a topology in SDN, the dynamics took place through the infrastructure of switches, which defined an intermediate switch ($sw1$) and five edge switches ($s1$, $s2$, $s3$, $s4$ and $s5$) containing 10 (ten) hosts each switch. The topology was conceived with the objective of building the technique for attack detection from the intermediate switch ($sw1$), which manages the other switches. Data streams from these switches, when connecting to controller and/or interacting with other networks, must be directed by data streams in $sw1$. All techniques able to detect attacks of the SYN Flood type were executed, mainly the proposed DetectAMath technique, which is the subject of this work.

B. Results

In the process of analyzing and displaying the results, the average execution times of the DetectAMath, Fuzzy, MLP, and Entropy techniques were 0.00002 seconds, 0.30397 seconds, 2.33535 seconds, and 0.00066 seconds, respectively. The maximum execution times for each technique were 0.0048 seconds, 3.8387 seconds, 20.6825 seconds, and 0.1364 seconds, respectively. DetectAMath has demonstrated superior efficiency in the analysis, with the analysis taking only 0.0048 seconds in the worst case. Fig. 1 **(a)** represents the variation in the analysis time, as observed through the statistical box plot, which demonstrates the efficiency of the techniques. The DetectAMath and Entropy techniques are positioned at the zero axis, indicating their equal performance. However, they differ in the evaluation of metrics presented in Table II.

For the analysis of results, validation metrics were generated including Accuracy, Precision, Recall and F1-Score. DetectAMath has resulted in precision of 100%, Fuzzy reached 95.62%, MLP reached 50.72% and Entropy achieved 89.53%. It can be seen that the proposed DetectAMath technique obtained the best result among the other techniques, reaching an accuracy of 100% both for attack cases and for normal traffic. Table II presents the results for Technique (Tec), which represents which technique was applied, Status (St.) (Normal, Attacks and Not Conclusive), True Quantity (n (Tr.)), Classified Quantity (n (Cl.)), Accuracy (Ac.), Precision (Pr.), Recall (Re.) and F1-Score (F1) for all tested techniques. It is important to emphasize that the tests carried out in a real scenario are non-deterministic, and variations in the results may occur with each simulation. In this case, there was a variation in the number of windows between the tests of each technique, with the number of windows being 295, 297, 278 and 276 for the DetectAMath, Fuzzy, MLP and Entropy techniques, respectively.

TABLE II

Tec	**St.**	**n (Tr.)**	**n (Cl.)**	***Ac.**	***Pr.**	****Re.***	****F1***
01	Normal	196	196	100	100	100	100
	Attack	99	99	100	100	100	100
	Not Conc	–	–	–	–	–	–
02	Normal	168	158	96	99	93	96
	Attack	129	127	99	100	98	99
	Not Conc	0	12	96	–	–	–
03	Normal	179	130	51	66	48	56
	Attack	99	148	51	37	56	45
	Not Conc	–	–	–	–	–	–
04	Normal	155	126	89	100	81	89
	Attack	122	151	89	81	100	89
	Not Conc	–	–	–	–	–	–

$*$ All values in percents (%).

Finally, Fig. 1 **(b)** presents the accuracy of the systems based on the applied techniques and interfaces, where the suggested technique's detection of attacks shows superior performance compared to the other techniques, thus proving its efficiency and effectiveness in DDoS attack detection. The DetectAMath and Fuzzy techniques achieved an overall accuracy above 90%. However, the Fuzzy technique showed errors in the majority of normal traffic analysis cases, classifying them as inconclusive. The Recall exhibited a higher error rate for normal traffic, with the Fuzzy system test showing a Recall of 98% for attacks and 93% for normal traffic. The worst-performing technique analyzed was MLP, with an accuracy of only 50.72%, highlighting the need for a larger volume of data for training and learning in order to make better decisions. Regarding the proposed DetectAMath technique, the obtained result can be considered satisfactory and efficient as it achieved 100% precision for both normal traffic and attacks. This technique stood out as the best among those evaluated in this study, demonstrating its potential and feasibility for use in other scenarios.

C. Discussion

The results showed a performance for the DetectAMath technique of 0.00002 seconds and a maximum time of 0.0048 seconds. The Fuzzy technique had a minimum time of 0.30397 seconds and a maximum time of 3.8387 seconds, while the Neural Network MLP had a minimum time of 2.33535 seconds and a maximum time of 20.6825 seconds. Finally, Shannon's Entropy obtained a minimum time of 0.00066 seconds and a maximum time of 0.1364 seconds. Results demonstrate that the DetectAMath technique performed better in the analysis and visualization of results, with a maximum time of 0.0048 seconds during the execution of the tests. The results were corroborated by the validation metrics, including accuracy, precision and recall, presented in Table II. It is important to emphasize that, although the tests were deterministic in a real scenario, it is possible to have variations in the results for each technique and in each simulation.

TABLE III

Techniques	Qty.Window	Process	Qty.Attack	Qty.Pkts
DetectAMathd	295	0.3154	99	449,394
Entropy	276	0.3467	151	875,342
Fuzzy	297	0.3312	128	600,209
MLP	278	0.2730	148	493,321

Table III presents an analysis of the performance of the techniques, which shows the number of windows found, where each window was determined with a time of 10 seconds. It also presents the average processing time to finish this total of windows, the number of attacks identified and, finally, the total number of packets sent during the attack detection process.

V. Conclusion

This work presented DetectAMath, a technique for detecting DDoS attacks at runtime. It is a simple and efficient, which does not require training or prior knowledge. This technique is based on a stochastic mathematical model to analyze the conditions of *SYN Flood* DDoS attacks. Its evaluation followed the approach of comparing the proposed technique to applied techniques (Fuzzy, Neural Network MLP and Shannon Entropy). Evaluation measured the efficiency and effectiveness of the technique proposed in this work. The results achieved 100% accuracy in detecting attacks, thus surpassing the other applied techniques. In addition, the technique also obtained results in runtime, which makes it different from other techniques. As future work, the intention is to apply the technique in real-time environments and also to use it in other contexts, such as IoT, since it involves a small code that doesn't require intensive processing for attack detection. This technique can contribute when applied in conjunction with other techniques with machine learning, such as federated learning.

References

[1] Check Point Research Reports. "Check Point Research Reports a 38% Increase in 2022 Global Cyberattacks,".

[2] NETSCOUT. "NETSCOUT DDoS Threat Inteligence Report / 5TH Anniversary Edition." accessed in 07/22/23.

[3] M. Dimolianis, A. Pavlidis, V. Maglaris. "SYN flood attack detection and mitigation using machine learning traffic classification and programmable data plane filtering." 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). IEEE, 2021.

[4] M. M. Salim, S. Rathore, J. H. Park. "Distributed denial-of-service attacks and its defenses in IoT: a survey." The Journal of Supercomputing 76 (2020): 5320-5363.

[5] S. Anil, "5th Anniversary DDoS Threat Intelligence Report" - Unveiling the New Threat Landscape, 2023.

[6] B. Nugraha, R. N. Murthy. "Deep learning-based slow DDoS attack detection in SDN-based networks." 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2020.

[7] M. Zhan, Y. Li, H. Yang, G. Yu, B. Li and W. Wang, "Coda: Runtime Detection of Application-Layer CPU-Exhaustion DoS Attacks in Containers." in IEEE Transactions on Services Computing, vol. 16, no. 3, pp. 1686-1697, 1 May-June 2023, doi: 10.1109/TSC.2022.3194266.

[8] M.J. Awan, U. Farooq, H.M.A. Babar, A. Yasin, H. Nobanee, M. Hussain, O. Hakeem, A.M. Zain. "Real-Time DDoS Attack Detection System Using Big Data Approach." Sustainability 2021, 13, 10743. https://doi.org/10.3390/su131910743

[9] J. Singh, B. Sunny. "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions." Computer Science Review 37 (2020): 100279.

[10] Daneshgadeh, Salva, et al. "An empirical investigation of DDoS and Flash event detection using Shannon entropy, KOAD and SVM combined." 2019 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2019.

[11] S. Daneshgadeh, et al. "Detection of DDoS attacks and flash events using Shannon entropy, KOAD and Mahalanobis distance." 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). IEEE, 2019.

[12] L. D. Tsobdjou, S. Pierre, & A. Quintero, "An online entropy-based DDoS flooding attack detection system with dynamic threshold". IEEE Trans. on Network and Service Management, 2022, 19(2), 1679-1689.

[13] N. Ahuja, G. Singal, and D. Mukhopadhyay. "DLSDN: Deep learning for DDoS attack detection in software defined networking." 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2021.

[14] Javaheri, Danial, et al. "Fuzzy logic-based DDoS attacks and network traffic anomaly detection methods: Classification, overview, and future perspectives." Information Sciences (2023).

[15] I. Kotenko, I. Saenko, S. Ageev. "Applying intelligent agents for anomaly detection of network traffic in internet of things networks." 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS). IEEE, 2018.

[16] E. P. Farias Junior, A. J. Tavares, & M. Nogueira, "Lógica Paraconsistente Anotada Aplicada na Detecção de Ataques DDoS em Redes SDN." Anais do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. SBC, 2019.