DoS Detection System Based on Dynamic Thresholding Algorithm using Netflow and Elasticsearch

Adian Fatchur Rochim
Departement of Computer Engineering,
Faculty of Engineering
Diponegoro University
Semarang 50275, Indonesia
adian@ce.undip.ac.id

Muhammad Sayyidus Shaleh Yofa
Departement of Computer Engineering,
Faculty of Engineering
Diponegoro University
Semarang 50275, Indonesia
muhammadssy@student.ce.undip.ac.id

Adnan Fauzi
Departement of Computer Engineering,
Faculty of Engineering
Diponegoro University
Semarang 50275, Indonesia
adnan@ce.undip.ac.id

***Abstract*— Network infrastructure requires constant monitoring over time, including identifying problems. Generally, an experienced network administrator responsible for the entire network is doing this task, which is inefficient because an administrator does not have full 24-hour availability in providing a fast and proper response at the time of the ongoing attack. Network attack identification systems are built to help solve this problem. This study focuses on identifying network attacks, namely Denial of Service attacks, by implementing the Dynamic Thresholding method. The data used for this research are the DARPA 2000 dataset and the self-generated dataset in a controlled laboratory environment, with the Netflow protocol and the Elasticsearch search engine. The results of the implementation show a lower False-Positive Rate in the DARPA 2000 dataset (33.33%) and the generated dataset (50%) in comparison with the False-Positive Rate value on the original paper (98%).**

Keywords— Computer Network, Network Security, Denial of Service.

I. INTRODUCTION

The use of Internet technology continues to increase in this information age. Along with developing technology supporting the network on the Internet, various kinds of risk of attacks on network infrastructure also develop [4]. In the CIA triad (Confidentiality, Integrity, Availability), many attack variants target the 3rd aspect (Availability), such as Denial of Service (DoS). DoS attacks are considered very dangerous because they can inflict huge losses with only a short duration of attack [1].

Many studies have proposed solutions to overcome DoS attacks, from the identification process [2] to mitigation after an attack incident [3]. The DoS identification method is divided based on the data source, and the algorithm used [4]. DoS detection can retrieve data from observation logs from Host, Network, or both (Hybrid) based on the data source. Based on the algorithm used, the DoS identification method is divided into two: Signature-based and Anomaly-based. Each approach has advantages and disadvantages. Signature-based methods have the advantages of a fast detection process and a low false-positive rate, but they cannot detect new attack patterns. In contrast, the Anomaly-based method can detect new attack patterns but with a greater risk of false-positive rates and require more computational resources.

Mukhlish et al., in 2019, proposed a log management system using ELK stack (Elasticsearch, Logstash, Kibana) to assist administrators in monitoring network problems [5]. The Log Management System classifies network devices' log types based on the importance level of the devices' logs. He classified network device logs by standard of Syslog. Implementation of the system was integrated by dashboard application. This study focuses on the processing of network device log data.

In 2016, Kaur H. conducted a comparative study about a DoS attack taxonomy. He explained tools for attacking a system to help researchers select tools as experimental tools [6]. This study classified DoS attacks based on network interfaces, dynamic rate, operating system, attack model, network protocols, attack categories, and targets. Since the DoS attack category's possibilities are extensive, there is a need for collaboration from the Internet community.

In 2019, David J. developed a DoS identification method that looks to anomalies of data packets flow passing through the network based on the Dynamic Thresholding method [7]. This algorithm compares the aggregation results of four features of packets that enters the network (Number of packets, Unique Source IP Count, Unique Destination IP Count, and Unique Protocol Count) which will be used to calculate the moving average and moving variance for each feature at a certain time interval. This is then compared to a dynamically set threshold to determine the normality of the traffic. The method's advantage is the lower consumption of computational resources than other variants of Anomaly-based methods, such as the ARIMA model and chaos theory, or the Machine Learning model [8] [9]. This method's disadvantage is that the parameters must be explicitly set for different use cases.

From the research conducted by David J., the author considers the need to implement the Dynamic Thresholding algorithm in a real system, whose results will be evaluated and compared with the original algorithm's results to be analyzed and explored for other possible implementations. The author tries to implement the algorithm as close as possible to real use cases using software and tools commonly used by computer network administrators.

This research aims to create an implementation of identification and classification of DoS attacks from log data of network system from Cisco CSR1000v virtual router [10] based on Dynamic Thresholding method.

The difference between this research and [7] is that this research aims to test whether the Dynamic Thresholding algorithm can be implemented to be used with current data processing and network monitoring technologies (e.g.

Elasticsearch and Netflow) while the former research proposed the algorithm and tested it in mathematical simulation software (MATLAB).

The following order describes the research work: the first section is the introduction and related works, the following section explains research methodology, the third section will explain the result and discussion, and the final section is the conclusion.

II. Research Methodology

This section will explain detailed information about this research process, from setting up the experiment environment to the steps of experiments involved.

Design Science Research (DSR) is a research methodology that focuses on developing and evaluating designed system artifacts. Hevner et al. state that the purpose of DSR methodology is understanding a problem domain by building a system of designed artifacts [11]. This methodology is commonly applied to Engineering and Computer Science field [12].

DSR methodology is used to design the implementation of the Dynamic Thresholding algorithm. The DSR methodology flow consists of system definition, system specification, system configuration, and experiments, evaluation, and results.

The first step is system definition. This stage describes the system to be created, the system's benefits and objectives, the requirements of the system, the topology, and how the system works.

The second step is the system specification. The requirements specification process will be explained in the initial system design by determining the specifications of requirements that conform to the system definition.

The third step is system configuration. In this step, the specified requirements will be designed according to the network design and applied as a system. According to the original paper, a series of experiments will be conducted on the system [7].

The fourth step is testing the system performance and results. The results of the research can be concluded from the system test output.

A. Requirements Identification

The Denial of Service identification system's functional requirement is a system that can receive aggregated network data from outside at each specific time interval and process that input, which with the output can determine whether an attack is occurring in that time interval (positive) or otherwise (negative).

This research's hardware requirement consists of one personal computer unit and one Cisco CSR1000v virtual router instance. DoS identification systems require supporting software to work. The system uses Ubuntu Server 20.04 OS. The Filebeat log collector is used to receive Netflow protocol flows and pass parsed data to Elasticsearch for storage. Here Elasticsearch act as a database and a search engine for querying. The Dynamic Thresholding algorithm's implementation is made using Python programming language.

B. Topology Design

System topology design is done locally using GNS3 software for computer network simulation. In the experimental scenario that has been carried out, three PCs act as hosts that will send normal traffic to the Web Server through the router. Meanwhile, one PC acts as an attacker who will send flood traffic to the Web Server through the router. All incoming traffic from normal hosts and attackers will be recorded and sent to Filebeat via the Netflow protocol. The topology design itself is shown in Figure 1.

Host "AlpineLinux" 1-3 simulating a user visits a webpage on the Web Server on host "Nginx" by way of HTTP requests at random intervals from 1 to 5 seconds. In the event of an attack, the "KaliLinux" host sends a TCP Flood with spoofed source IP address to the "Nginx" host IP address for 10 seconds. This DoS attack is carried out with the Hping3 tool.

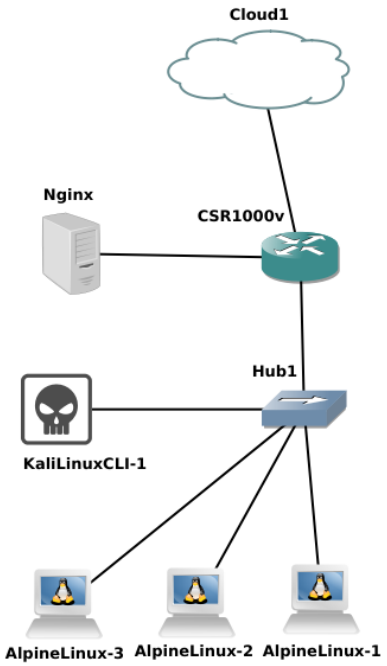

Figure 1 Network Topology Design of DoS Attack Identification System

C. System Design and Workflow

The workflow for implementing the DoS identification system with Dynamic Thresholding, as shown in Figure 2, is described as follows:

1. Incoming traffic from other networks passes through the CSR1000v router.
2. CSR1000v router performs constant traffic flow header export with Netflow protocol.
3. Filebeat receives the Netflow export data, parses it, and then sends the formatted data to Elasticsearch. Elasticsearch stores Netflow parsing data.
4. Every T seconds, Detector will query data to Elasticsearch to retrieve the traffic header aggregation as algorithm input.
5. Detector receives the traffic header aggregation metrics. With this input, the algorithm analyzes whether it is positive (attack) or negative (normal).

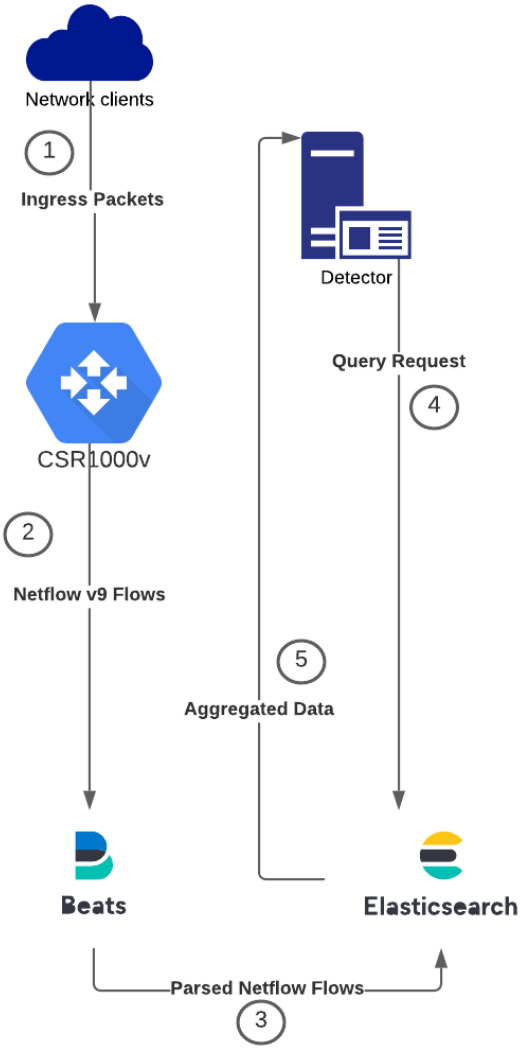

Figure 2 System Design and Workflow of the system

D. Experiment Settings and Metrics

Experiments in this study were carried out in 4 stages:

1. Implementation of the Dynamic Thresholding algorithm

The implementation step is carried out in the Python programming language with the help of the Jupyter Notebook library for the development environment, Pandas for processing the dataset, Numpy for statistical data processing, and Matplotlib for data visualization in graphical form plot.

The algorithm itself requires two parameters whose values are assigned before the first run, namely Window Size (K) and Sampling Interval (T). The first parameter (K) acts as the size of the moving average and moving variance for each iteration of the algorithm process. The second parameter (T) acts as sampling interval for the network traffic header. TABLE I lists algorithm parameter values at experiments conducted for DARPA 2000 Dataset and Generated Dataset.

The author has difficulty at this stage because David J.'s paper does not provide complete experimental parameters. This stage has experienced many failures due to the missing value of the Window Size (K).

However, there has been a success in approaching the results of the data plot in David J.'s paper, namely with Window Size (K) = 1.

2. Testing implementation on DARPA 2000 dataset

The DARPA 2000 dataset [13] was used for the first test because David J.'s research used this dataset for testing. The results of this test will be used as a comparison between the original paper and the implementation. DARPA 2000 is a dataset produced by Lincoln Laboratory of MIT with multiple scenarios. The dataset scenario used in this research is LLDOS 1.0. This scenario is created in the year 2000 by Lincoln Laboratory of MIT [14]. It includes a DoS attack run by a novice attacker. This attack scenario is carried out over multiple network and audit sessions. These sessions have been grouped into the following attack phases: The attacker probes the network, breaks into a host by exploiting a vulnerability found inside the system, installs trojan software, and launches a DoS attack at an off-site server from the compromised host. The grouped attack phases are included as one timeline in the original algorithm's implementation testing.

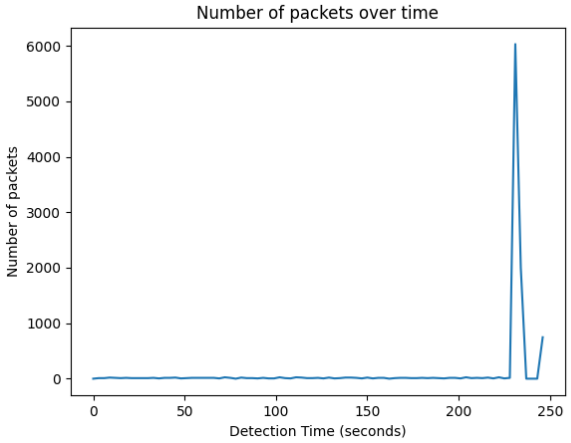

Figure 3 Number of packets over time

TABLE I. ALGORITHM PARAMETERS

Parameter Name	Value
Window Size (K)	1
Sampling Interval (T)	3

3. Generating own dataset

Collection of generated dataset starts from creating a virtual network scenario using GNS3 to simulate attack case. Netflow traffic is recorded for 4 minutes 20 seconds to Elasticsearch via the "Cloud1" node, with attacks launched from the 4th minute for 10 seconds. Figure 1 shows the topology of the system used in generating the dataset.

4. Testing implementation on the generated dataset

After the dataset is collected and stored in Elasticsearch, testing is carried out with the input of the data aggregation query from Elasticsearch for the dataset collection duration.

The data stored in elasticsearch contain the flow of traffic headers from Netflow, which does not have a form that

can be used as an input for the algorithm. Therefore, these data rows must be aggregated with the Elasticsearch aggregation query feature. The query syntax can be seen in Figure 7.

The aggregated data is the number of data packets per T seconds (Figure 3), the number of unique source IP addresses per T seconds (Figure 4), the number of unique destination IP addresses per T seconds (Figure 5), and the number of unique protocols per T seconds (Figure 6). The value of T can be seen in TABLE I.

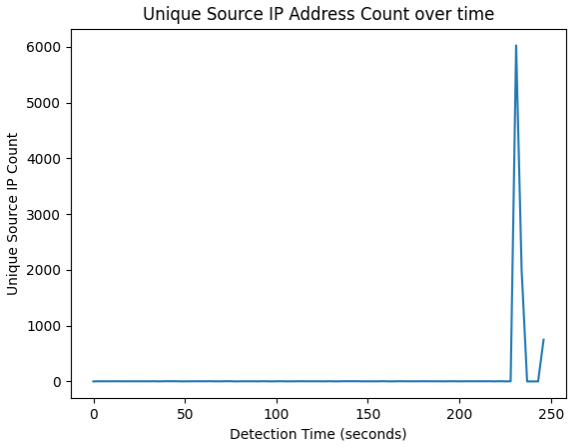

Figure 4 Unique Source IP Count over time

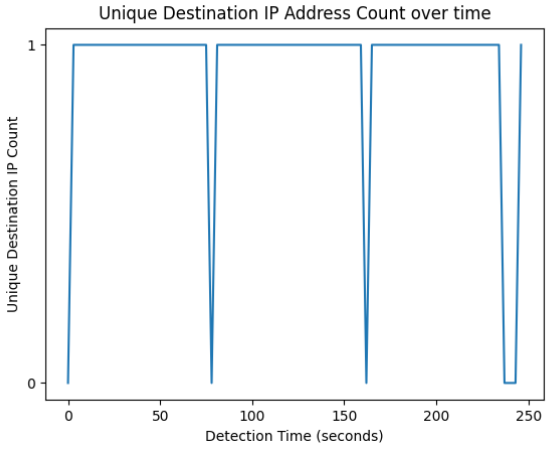

Figure 5 Unique Destination IP Count over time

III. Result And Discussion

The results of experiments in this research are gathered from locally collected data. The Confusion Matrix method is used to measure the performance of the algorithm implementation made [15], namely the accuracy, precision, sensitivity. These values are calculated from the count of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), where TP refers to correct predictions of attacks, TN indicates the normal data classified correctly as normal. FP refers to incorrect predictions of attacks, and FN indicates the attack classified incorrectly as normal data.

Sensitivity/False Positive Rate (TPR) measures the percentage of correctly identified attacks over the actual attacks in sampled traffic and is computed using Equation (1).

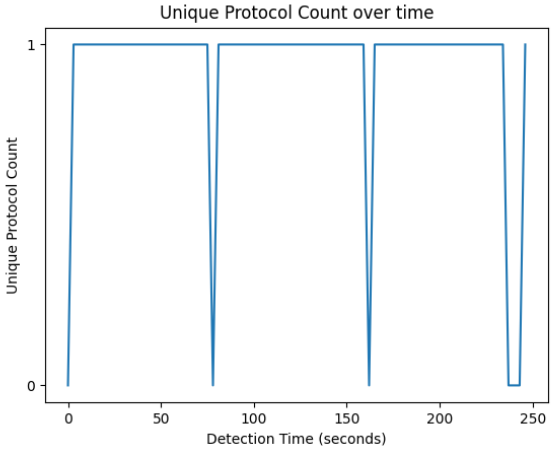

Figure 6 Unique Protocol Count over time

```
{
  {
  "size": "0",
  "query": {
    "range": {
      "@timestamp": {
        "gte": "now-3s",
        "lte": "now"
      }
    }
  },
  "aggs": {
    "packets": {
      "sum": {
        "field": "netflow.packet_delta_count"
      }
    },
    "USIP": {
      "cardinality": {
        "field": "netflow.destination_ipv4_address"
      }
    },
    "UPR": {
      "cardinality": {
        "field": "netflow.protocol_identifier"
      }
    }
  }
}
```

Figure 7 Elasticsearch Aggregation Query

$$Sensitivity = \frac{TP}{TP + FN} \quad (1)$$

Accuracy measures the percentage of accurate detection over the sampled traffic and is computed using Equation (2).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

Specificity/True Negative Rate(TNR) relates to the system's ability to detect sampled traffic without attack correctly. The specificity is computed by Equation (3).

$$Specificity = \frac{TN}{TN + FP} \quad (3)$$

TABLE II. DARPA 2000 Dataset Confusion Matrix

Total: 3885	**True Positive**	**True Negative**
Predicted Positive	1 (TP)	0 (FP)
Predicted Negative	2 (FN)	3882 (TN)

TABLE III. GENERATED DATASET CONFUSION MATRIX

Total: 83	True Positive	True Negative
Predicted Positive	1 (TP)	8 (FP)
Predicted Negative	1 (FN)	73 (TN)

TABLE IV. IMPLEMENTED SYSTEM RESULT

Datasets	TPR	Accuracy	TNR
DARPA 2000	33.33%	99.94%	100%
Generated Dataset	50%	89.15%	90.12%

The results of the algorithm implementation test are displayed in the form of a score table for False Positive Rate (FPR), Accuracy, and True Negative Rate (TNR) and the visualization of 4 quadrants of detection results. TABLE II and

TABLE III shows the Confusion Matrix of both results from DARPA 2000 dataset and Generated Dataset. TABLE IV shows measurements from this research's evaluation result. The clustered bar chart shown in Figure 8 visualizes evaluation result comparison based on values at TABLE IV and the original paper's result [7]. Figure 9 chronologically shows detection results, categorized in four quadrants of the confusion matrix.

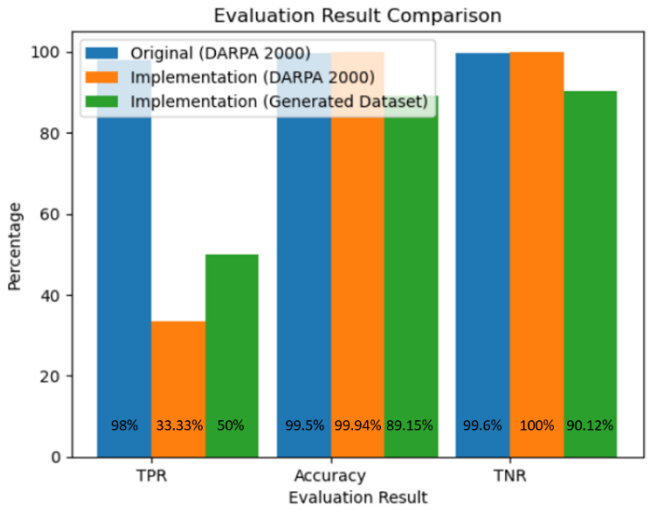

Figure 8 Evaluation Result Comparison

While the experiments are conducted locally, they can be replicated in a real-world environment such as cloud-based Virtual Private Clouds and virtual network devices that supports NetFlow and can be connected to an Elasticsearch instance.

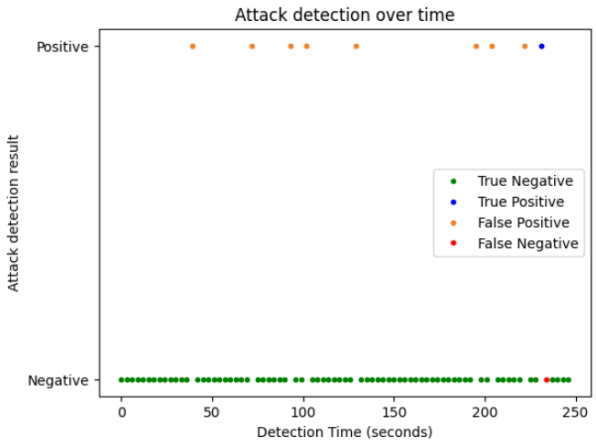

Figure 9 Attack Detection over time

IV. CONCLUSION

The system evaluation results show a lower True Positive Rate than the original paper. In the DARPA 2000 dataset, the evaluation results show a sensitivity (TPR) of 33.33%, an accuracy of 99.94%, and a specificity (TNR) of 100%, while the results of the evaluation on the generated dataset show 50% sensitivity (TPR), 89.15% accuracy, and specificity (TNR) 90.12%.

Performance scores are affected by data input to the system. When experimenting with generated datasets, it was found that the Netflow protocol was inconsistent in reporting traffic protocol information, with accuracy only up to OSI Layer 4. In contrast, in the DARPA 2000 dataset, the traffic protocol could be differentiated up to OSI Layer 7.

Suggestions for further research are implementing the Dynamic Thresholding algorithm using tools other than Netflow, such as tcpdump, which can distinguish traffic protocols more accurately, then compare the results obtained. While this research focused on high-rate DoS attacks, another emerging types of attacks such as slow-rate DDoS, reflection attack, and amplification attack are to be considered important as a research subject as well.

ACKNOWLEDGMENT

This research was financially supported by The Faculty of Engineering, Diponegoro University, Semarang, Indonesia through Strategic Research Grant 2021 number: 3178/S/komputer/2/UN7.5.3.2/PP/2021.

REFERENCES

[1] Ashu, R. Mahajan, and S. Zafar, “DDoS Attacks Impact on Data Transfer in IOT-MANET-Based E-Healthcare for Tackling COVID-19,” in *Data Analytics and Management*, A. Khanna, D. Gupta, Z. Pólkowski, S. Bhattacharyya, and O. Castillo, Eds. Singapore: Springer Singapore, 2021, pp. 301–309.

[2] J. David and C. Thomas, “Detection of distributed denial of service attacks based on information theoretic approach in time series models,” *J. Inf. Secur. Appl.*, vol. 55, no. October, p. 102621, 2020, doi: 10.1016/j.jisa.2020.102621.

[3] N. Z. Bawany, J. A. Shamsi, and K. Salah, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions,” *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017, doi: 10.1007/s13369-017-2414-5.

[4] M. Alenezi and M. J. Reed, “Methodologies for detecting DoS / DDoS attacks against network servers,” in *The Seventh International Conference on Systems and Networks Communications (ICSNC)*, 2012, no. c, pp. 92–98.

[5] A. F. Rochim, M. A. Aziz, and A. Fauzi, “Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack,” in *ICECOS 2019 - 3rd International Conference on Electrical Engineering and Computer Science, Proceeding*, 2019, pp. 338–342, doi: 10.1109/ICECOS47637.2019.8984494.

[6] H. Kaur, S. Behal, and K. Kumar, “Characterization and comparison of Distributed Denial of Service attack tools,” in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, Oct. 2015, pp. 1139–1145, doi: 10.1109/ICGCIoT.2015.7380634.

[7] J. David and C. Thomas, “Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic,” *Comput. Secur.*, vol. 82, pp. 284–295, 2019, doi: 10.1016/j.cose.2019.01.002.

[8] S. M. Tabatabaie Nezhad, M. Nazari, and E. A. Gharavol, “A Novel DoS and DDoS Attacks Detection Algorithm Using ARIMA Time Series Model and Chaotic System in Computer Networks,” *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 700–703, Apr. 2016, doi: 10.1109/LCOMM.2016.2517622.

[9] R. Rajendran, S. V. N. Santhosh Kumar, Y. Palanichamy, and K. Arputharaj, “Detection of DoS attacks in cloud networks using intelligent rule based classification system,” *Cluster Comput.*, vol. 22, no. S1, pp. 423–434, Jan. 2019, doi: 10.1007/s10586-018-2181-4.

[10] R. Hofstede *et al.*, “Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014, doi: 10.1109/COMST.2014.2321898.

[11] S. Laumer and A. Eckhardt, *Integrated Series in Information Systems*, vol. 28. 2012.

[12] A. Shahin, T. van Gurp, S. A. Peters, R. G. Visser, J. M. van Tuyl, and P. Arens, “SNP markers retrieval for a non-model species: a practical approach,” *BMC Res. Notes*, vol. 5, no. 1, p. 79, Dec. 2012, doi: 10.1186/1756-0500-5-79.

[13] C. Thomas, V. Sharma, and N. Balakrishnan, “Usefulness of DARPA dataset for intrusion detection system evaluation,” *Data Mining, Intrusion Detect. Inf. Assur. Data Networks Secur. 2008*, vol. 6973, p. 69730G, 2008, doi: 10.1117/12.777341.

[14] M. L. Laboratory, “Lincoln Laboratory Scenario (DDoS) 1.0,” 2000. https://archive.ll.mit.edu/ideval/data/2000/LLS_DDOS_1.0.html.

[15] M. E. Elhamahmy, H. N. Elmahdy, and I. A. Saroit, “A New Approach for Evaluating Intrusion Detection System,” *CiiT Int.*, vol. 2, no. 11, pp. 290–298, 2010.