Journal of Asian Scientific Research
ISSN(e): 2223-1331
ISSN(p): 2226-5724
DOI: 10.55493/5003.v16i1.5890
Vol. 16, No. 1, 356-372.

URL: www.aessweb.com

A multi-modular distributed framework for real-time DDoS detection in docker environments

Sushant Chamoli[1,2]
Varsha Mittal[1]
Narendra Khatri[3+]

[1]*Department of CSE, Graphic Era Deemed University, Dehradun, India.*
[1,2]*Email: schamoli@gehu.ac.in*
[1]*Email: varshamittal@geu.ac.in*
[2]*Department of CSE, Graphic Era Hill University, Dehradun, India.*
[3]*Department of Mechatronics, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal-576104, Karnataka, India.*
[3]*Email: narendra.khatri@manipal.edu*

(+ Corresponding author)

Article History
Received: 3 October 2025
Revised: 8 January 2026
Accepted: 28 January 2026
Published: 6 February 2026

Keywords
Anomaly detection
Cloud-native security
Cyber security
DDoS detection
Distributed computing
Docker containers
Intrusion detection systems
Multi-modular architecture
Real-time monitoring.

ABSTRACT

The Internet has permeated nearly every aspect of modern civilization, interconnecting billions of devices and services. While this global connectivity empowers businesses to deliver services seamlessly, it also exposes them to persistent cyber threats such as Distributed Denial of Service (DDoS) attacks. Owing to the diversity in attack scale and sophistication, conventional detection methods often fail to respond effectively, particularly when a single detection approach is used against multiple attack variants. This paper proposes a multi-modular approach for DDoS detection in Docker container environments, combining statistical and anomaly-based techniques to identify bandwidth-driven attacks in real time while maintaining system stability. The framework distributes its computational components across multiple nodes through parallel execution, thereby minimizing the processing load on the target system. Experimental validation using the CICDDoS2019 dataset and simulated Docker-based attacks demonstrates the effectiveness of the proposed design. Multi-attribute analysis reduced false positives from 81.11% to 1.15% and achieved an average detection accuracy of 99.95%. Distributed processing reduced system resource usage by 40% compared to centralized techniques. Modular design and distributed computing improve detection precision, reduce false alarms, and provide a scalable, resource-efficient defense against developing DDoS threats in cloud-native infrastructures.

Contribution/ Originality: This study contributes to the existing literature by proposing a multi-modular, distributed framework for real-time DDoS detection in Docker environments, integrating statistical and anomaly-based methods to enhance detection accuracy, reduce false positives, and lower computational overhead compared to conventional single-model approaches.

1. INTRODUCTION

With the advent of the Internet, the world has become increasingly interconnected, eliminating geographical restrictions and enabling organizations of all sizes to operate globally. Businesses now rely heavily on online platforms to deliver products and services, yet this digital expansion exposes them to a range of cyber threats. Among these, Distributed Denial of Service (DDoS) attacks remain one of the most disruptive, as they threaten the availability pillar of the classic Confidentiality, Integrity, and Availability (CIA) triad that underpins network security.

A DDoS attack floods a target system with malicious traffic generated by compromised nodes, preventing legitimate users from accessing services. According to the Cloudflare DDoS Threat Report [1], approximately 2200

DDoS attacks were mitigated every hour in 2024, with one attack reaching a staggering 4.2 Tbps. Notably, about 90% of these attacks lasted less than an hour, underscoring the need for real-time detection strategies capable of identifying and mitigating bandwidth-based DDoS attacks swiftly [2].

Given the increasing sophistication and intensity of these attacks, traditional detection systems often designed around fixed signatures or single analytical models are no longer adequate. They either fail to adapt to the rapidly evolving attack landscape or impose excessive computational overhead on already stressed systems. Therefore, this study aims to answer the following key research questions.

i. Can a modular and distributed detection architecture improve real-time responsiveness to bandwidth-based DDoS attacks?
ii. How effectively can statistical and anomaly-based approaches complement each other to reduce false positives while maintaining high detection accuracy?
iii. To what extent does distributed processing lower the computational load on target systems during large-scale DDoS incidents?

In addressing these questions, this paper proposes a multi-modular, distributed DDoS detection framework designed for Docker-based environments. The system integrates statistical and anomaly-based techniques, distributing workloads across multiple nodes to enable efficient, scalable, and real-time detection.

2. LITERATURE REVIEW

DDoS attacks have been extensively studied due to their prevalence and impact on network reliability and service availability. Early research focused on understanding attack mechanisms and taxonomy. Mirkovic and Reiher [3] classified DDoS attacks based on recruitment phases, objectives, and execution mechanisms, while Brooks et al. [4] traced their evolution and highlighted the lack of centralized Internet access control as a fundamental weakness. The architecture and propagation of DDoS attacks were further detailed in Douligeris and Mitrokotsa [5], where attacks were categorized by rate, automation level, and exploited vulnerabilities. Srivastava et al. [6] distinguished between logic-based and flood-based attacks and discussed mitigation techniques such as firewalls, filters, and rate throttling, as well as detection methods including signature-based and anomaly-based approaches.

The increasing reliance on cloud infrastructure has expanded the threat surface. Cloudflare's mitigation of a record-breaking 5.6 Tbps attack in October 2024 underscores this urgency. Yoachimik and Pacheco [7]. Somani et al. [8] examined DDoS incidents targeting cloud systems, providing a taxonomy of prevention, detection, and mitigation strategies. Similarly, Gupta and Badve [9] analyzed DDoS tools, techniques, and defensive mechanisms across network layers, highlighting ongoing security challenges in cloud computing.

Statistical and feature-based methods have been explored for real-time DDoS detection due to their computational efficiency. Amma et al. [10] proposed the use of Class Scatter Ratio (CSR) and Feature Distance Map (FDM) to identify DDoS attacks, achieving results within a 95% confidence interval. Kim and Reddy [11] introduced destination IP correlation analysis for early detection at egress routers, while Tsobdjou et al. [12] validated entropy-based metrics using parameters such as destination IP, protocol, and packet length for TCP-SYN, Smurf, and UDP flood attacks.

Visualization and data analytics tools have also been leveraged to enhance interpretability. Khalaf et al. [13] applied big data analytics using RapidMiner and Power BI, while Sarmento et al. [14] used Tableau for dataset exploration and comparative evaluation of machine learning models. Patidar et al. [15] and Sufi [16] demonstrated that Power BI and Hadoop-based frameworks can reveal network anomalies and correlations across global attack datasets.

Parallel to these detection efforts, Docker has gained significant traction in cloud computing due to its lightweight virtualization and modularity. Studies have explored its performance advantages and security challenges. Chung et al. [17] and Jaramillo et al. [18] discussed Docker's efficiency and its integration with microservice

architectures, whereas Singh and Singh [19] highlighted container-based virtualization as a key component of future cloud infrastructures.

However, the shared kernel architecture of Docker containers introduces vulnerabilities, including exposure to DDoS attacks. Foundational work such as Bui [20] and Chamoli [21] examined Docker's built-in security frameworks SELinux, AppArmor, and network isolation, while Chelladhurai et al. [22] and Chamoli [21] emphasized the risks of misconfiguration and proposed mitigation strategies such as memory limiting and multi-category security enforcement.

Recent studies have shifted toward distributed detection frameworks to address the limitations of centralized systems. Patil et al. [23] proposed a distributed DDoS detection model that improved accuracy and reduced false positives by leveraging parallelism.

Khooi et al. [24] introduced DIDA, an in-network defense system that optimizes load balancing between access and border routers. Stream-processing frameworks such as Apache Spark and Kafka were utilized by Maheshwari et al. [25] and Yahyaoui et al. [26] to enable faster detection and scalable data analysis.

Despite these advances, most existing systems rely on single-method approaches such as statistical, entropy-based, or machine learning techniques, which limit adaptability and impose additional processing burdens. The present study distinguishes itself by proposing a multi-modular detection system that distributes its analytical workload across modules: System Analyzer, Traffic Monitor, and Relation Synthesizer, each contributing to the detection decision collaboratively.

This architecture enhances detection accuracy, reduces false positives, and achieves real-time responsiveness without overloading the target system, addressing key gaps in prior research.

3. DATA AND METHODOLOGY

3.1. Dataset

The CICDDoS2019 dataset has been considered for training and testing purposes. Created by the Canadian Institute for Cybersecurity, the CICDDoS2019 dataset includes several DDoS attack types organized into two directories: one for training and one for testing.

Each directory contains multiple CSV files reflecting various DDoS attack types, including UDP, LDAP, NetBIOS, SYN, and MSSQL. These files are generated with a tool called CICFlowMeter and contain numerous instances categorized using 88 attributes.

3.2. Proposed Architecture

The proposed DDoS detection model, comprising the System Analyzer, Traffic Monitor, and Relation Synthesizer, is illustrated in Figure 1. Each of these components functions as independently as possible, and the concept of a "Critical State" is maintained to assess how critical a system's state is due to the incoming flood of packets. Each component contributes to the Critical State in a different proportion, and ultimately, the occurrence of a DDoS attack is determined.

Although each component operates on the incoming traffic that a system receives, there are a few reasons why it is not feasible to execute all three components of the proposed DDoS detection model on a single system. First, the system, which is already having trouble managing the amount of incoming traffic, will be overburdened. Second, a severe DDoS attack could bring down the entire system before the proposed DDoS strategy ever comes into effect. Consequently, the principles of distributed computing are incorporated into our model.

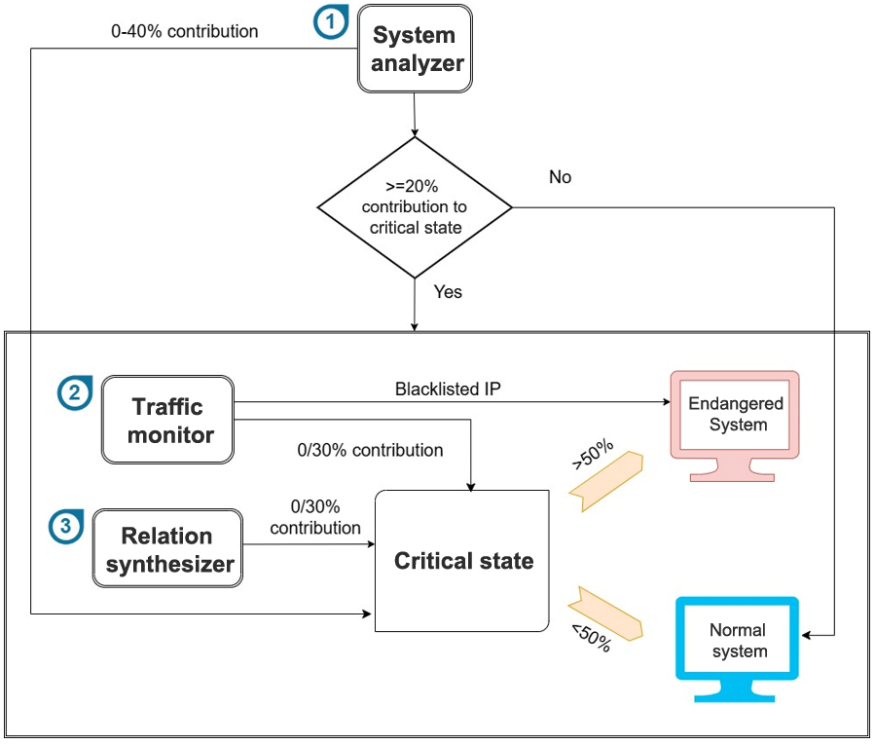


Figure 1. Proposed architecture for detecting DDoS attacks.

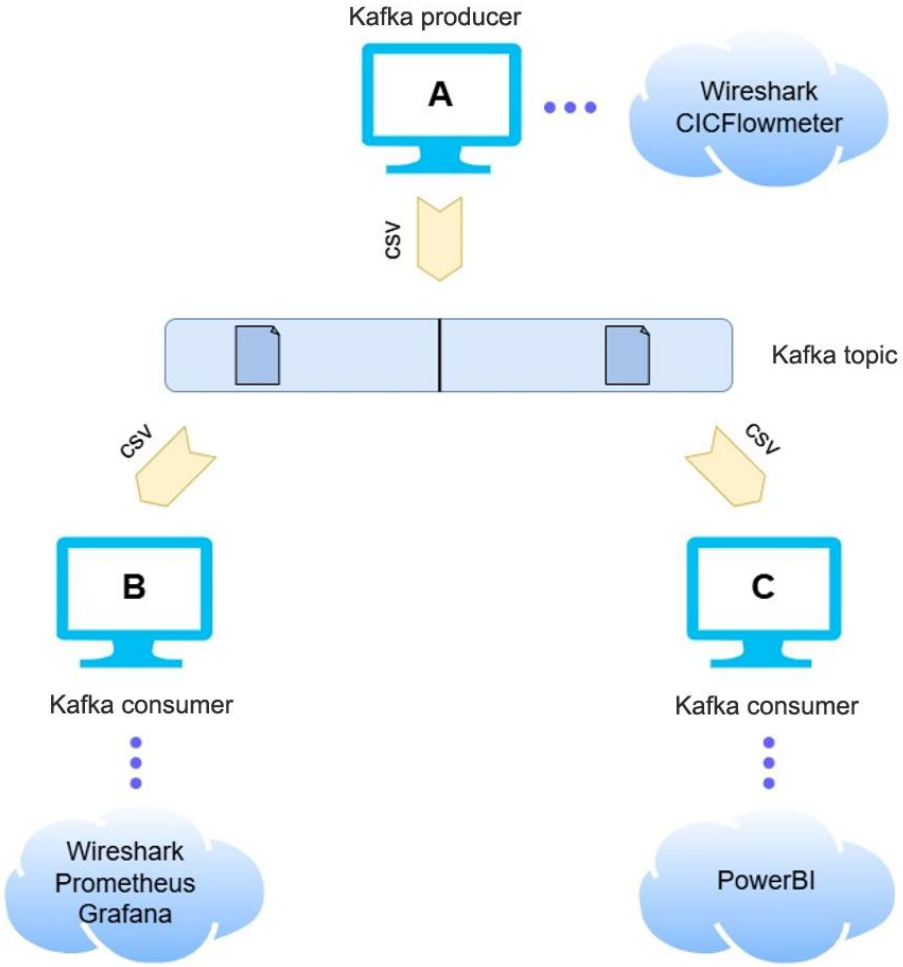


Figure 2. Multi-Node DDoS detection framework.

For distributed messaging across several systems, Apache Kafka is used. To implement the model, three distinct systems are required, as can be seen in Figure 2. Several tools are employed on each system to make the entire model efficient and allow for effective parallelism. System A, which is the target of DDoS attacks, has a Dockerized application installed, along with Wireshark and CICFlowmeter. System A uses both of the aforementioned tools to create CSV files that represent the incoming traffic. The second system, System B, is equipped with Grafana, Prometheus, and Wireshark. System B determines the load level that the target System A is under. The third system, System C, is equipped with Power BI. By analyzing the relationship between different packet header fields, System C detects any anomalies. The Kafka Producer is installed on System A, and Systems B and C act as Kafka Consumers. After reading the CSV file created by Wireshark/CICFlowmeter, the Kafka Producer forwards it to the Kafka topic located on the Kafka Broker. In Kafka topics, the Kafka Broker manages all storage. The CSV files kept in Kafka topics are consumed by Systems B and C, which are Kafka consumers.

The functioning and implementation of each of these modules the System Analyzer, Traffic Monitor, and Relation Synthesizer are explained in depth in the sections that follow.

3.2.1. System Analyzer

Microservice architecture has gained a lot of prominence in the past few years owing to better resiliency, a modular approach, and separation of concerns. Docker's popularity stems from the fact that it offers the ideal framework for microservice applications, allowing distinct microservices to run in separate containers. Due to their same underlying host kernel, Docker containers share all system resources. *Docker stats*, a command offered by Docker, provides information on container utilization in real time. It determines how much CPU, memory, NET I/O, and Block I/O are used by each Docker container running on a system.

According to Cloudflare's DDoS Threat Report [7] in 2024 Q4, 91% of network layer DDoS attacks finished in under ten minutes, with just 2% lasting over an hour. Also, the amount of network-layer DDoS attacks exceeding 1 Tbps increased by 1,885% QoQ, and attacks exceeding 100 million pps (packets per second) increased by 175% QoQ. Given the brief duration of DDoS attacks, the Docker stats command is executed often every three seconds for ten iterations. This procedure can be carried out repeatedly to offer a comprehensive DDoS detection approach. Additionally, the focus is on bandwidth-based network-layer DDoS attacks, and the criterion used to identify any irregularities in the system is NET Input, or the total quantity of network traffic that reaches a container. The network's load-bearing capacity can be ascertained using stress testing tools such as JMeter. Three thresholds have been established for detecting anomalies: NET Input below 60% of network capacity, between 60% and 80% of network capacity, and above 80% of network capacity, as shown in Table 1. A "Critical State" is maintained in memory to signify the deteriorating condition of the system. Since a DDoS attack can cause significant harm in a very short span of time, different contributions are made to *Critical State* depending on the intensity of incoming traffic, as illustrated in Table 1.

When used frequently across a large number of operational containers, the *Docker stats* command can be resource-intensive even though it requires very little system resources. Therefore, the *--no-stream* argument with the *Docker stats* command is utilized to record resource usage of containers at specific times instead of continuously tracking container consumption in real time. Regular checks are also made to see if the *Docker stats* command is in the top 50th percentile of the processes using the most system resources. If so, a 50% increase in the command's running interval is made.

Occasionally, a container may experience a brief spike in inbound traffic, which the proposed model interprets as a DDoS attack. To handle this scenario, the concept of Moving Average is utilized, wherein the average of a fixed number of data points is calculated, and when a new data point arrives, it is considered while ignoring the last available data point. Moving average helps smooth out short-term fluctuations in data points, providing a clearer view of the

ongoing trend. Thus, if V_x represents the value of a data point at time x, then the moving average at time t over a fixed number of data points n can be represented as.

$$MA_t = (V_t + V_{t-1} + V_{t-2} - - - - - + V_{t-n+1}) / n \quad (1)$$

Table 1. Contribution to the critical state due to varying intensity of incoming packets.

Net input	Min. Count/Iteration	Contribution to critical state
>80%	3/10	40%
60%-80%	5/10	20%
<60%	ANY	0%

Prometheus and Grafana tools are also employed as independent sub-components of the System Analyzer to identify any deviations from the system's baseline behavior. Prometheus is an open-source monitoring tool used to scrape the metrics of a target system and store them in a time-series database for further analysis. Grafana is a visualization tool that displays the time-series data obtained from Prometheus in a visually appealing manner, such as graphs, charts, and tables.

3.2.2. Traffic Monitor

The Traffic Monitor component regularly examines incoming container traffic and checks for malicious IP addresses to confirm a DDoS attack. Incoming network packets destined for containers are recorded every 10 seconds for a total of five seconds throughout three repetitions. An IP Manifest Analysis table is generated, containing information on all Source IPs, the total number of packets received from them, the total number of error packets received from them, the total number of ports they use, and the speed of those packets. Simple custom scripts can be used to list these network metrics, which are supplied by the data harvesting tool Wireshark. By simulating a DDoS attack on a target system with the DDoS attack tools HOIC and HULK, it is observed that all of the prior metrics display abnormally high values, confirming their application in the proposed DDoS detection scenario. Once all of the aforementioned metrics have been normalized and weights assigned to each metric, a weighted score for each Source IP is determined. Below is an illustration of a weighted score, in which various normalized network metrics are given varying weights based on the historical and current state of the network.

$$Weighted\_Score\_IP = (0.3 * Norm\_Packets\_IP) + (0.15 * Norm\_SourcePorts\_IP) + (0.15 * Norm\_Bps\_IP) + (0.4 * Norm\_ErrorPct\_IP) \quad (2)$$

The Traffic Monitor component runs continually in the background so that an IP Manifest Analysis table is always available in case the System Analyzer, the previous component, indicates a suspect. Additionally, one can examine the IP Manifest Analysis table to gain insight into the historical behavior of different IP addresses. The behavior of particular systems attempting to communicate with the target system can be better understood by visualizing these accumulated tables. In the event that a DDoS attack is discovered, IPs with high Weighted scores can be blocked out using the IP Manifest Analysis table later on in the DDoS mitigation phase.

A comprehensive list of IP addresses implicated in malicious activity worldwide is provided by several web services, such as Spamhaus and Cleantalk. These lists, which are updated in real-time, can be used as a standard to determine whether an IP address is malicious. To determine whether a malicious IP address is actually launching a DDoS attack, all source IPs that fall within the 80th percentile of the IP Manifest Analysis table are recorded for the most recent three iterations and compared to IP blacklist databases. These blacklist databases are considered to be the final source of truth, so if an IP address is discovered in one, it is concluded that a DDoS attack has occurred and any further DDoS investigation is stopped. Even if the IP address is not listed in the blacklist databases, a 30% contribution is made against the Holistic state since new systems are targeted as bots and utilized to conduct attacks every day.

3.2.3. Relation Synthesizer

The Relation Synthesizer component utilizes data analysis tools and techniques to deduce relationships between various packet header fields. The network packets entering the system have a number of header fields that help in the correct and error-free delivery of packets from source to destination.

The relationships between these header fields are determined using a popular network analysis tool, CICFlowMeter. CICFlowMeter is a network flow analysis tool developed by the Canadian Institute for Cybersecurity, designed specifically for cybersecurity [27]. CICFlowMeter groups packets into flows and calculates more than 80 different parameters related to individual flows. A network flow is a group of packets sharing the same characteristics.

$$F = \sum_{i=1}^{n} P_i \qquad (3)$$

Where each packet P_i has the same 5-tuple attributes <SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol>. In the event of a DDoS attack, available resources are already extremely limited and should be used sparingly; hence, the focus is on packet flows since analyzing a packet flow requires less CPU power than analyzing individual packets.

Relation Synthesizer component operates in two phases training and testing. A well-known tool called Power BI, a powerful data analysis and visualization tool created by Microsoft, is used throughout the training phase. DDoS tools such as LOIC, HOIC, HULK, and HPING are employed to carry out DDoS attacks from various systems over a short period. CICFlowMeter is used on the target system to collect incoming network flows, with the resulting CSV output file fed into Power BI.

Different rules are generated by Power BI for each CSV file produced by individual DDoS tools, and these rules are documented in a Rules Table, as shown in Table 2. Each rule is assigned a weight that indicates its importance in identifying a DDoS attack.

During the testing phase, all rules from the Rules Table are matched against incoming network flows to determine whether the system is under DDoS attack. If a rule matches any incoming network flow, a contribution of 30% is made to the Critical State.

Table 2. Rules Table.

Rule	Weightage
Avg of flow packets per sec > 5000	20X
Sum of source port is > 1000	15X
Sum of flow duration is 2-6	15X

The efficiency of the Relation Synthesizer component depends on the rules generated by Power BI, which in turn depends on the data the component has been trained on. Figure 3 illustrates some of the rules generated by Power BI when working on a CSV file produced using CICFlowMeter.

However, numerous recent attack tools and scripts exist in the market, which need to be considered for the component to perform a comprehensive analysis of DDoS attacks. Recent DDoS datasets, including CICDDoS2019, contain CICFlowMeter-generated CSV files that depict various types of DDoS flows, such as NTP, NetBIOS, SSDP, UDP-Lag, and TFTP, among others [28].

Feeding these CSV files into Power BI will help generate rules representing current DDoS attack tools and can be beneficial for detecting the latest attacks.

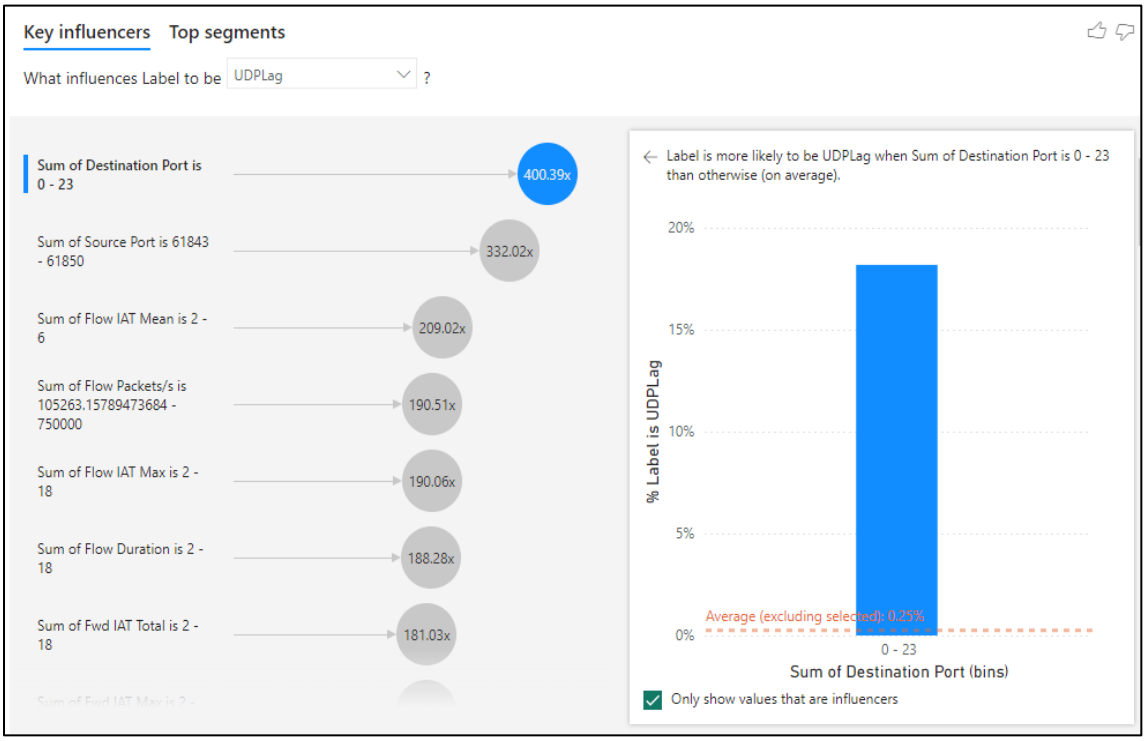


Figure 3. Rules generated by Power BI for detecting DDoS attacks.

4. FINDINGS

The proposed DDoS detection methodology was experimentally evaluated to verify its performance and reliability. Each component of the system was implemented and tested under controlled network conditions to assess its role in identifying malicious activity. A dedicated experiment was conducted in which ten distinct nodes simultaneously executed DDoS attacks on a Docker container deployed on the target host. This setup enabled a realistic simulation of distributed traffic behavior. Data collection, analysis, and visualization were supported by multiple tools including Wireshark, CICFlowMeter, Prometheus, Grafana, and Power BI as illustrated in Figure 2. To differentiate between normal and attack traffic, genuine inbound network flows were captured to represent benign behavior, while DDoS traffic was generated using the HULK tool. For model training and validation, the CICDDoS2019 dataset was employed to ensure consistency and comparability of results across testing scenarios.

4.1. System Analyzer

The System Analyzer module verifies the existence of DDoS attacks by detecting how incoming packets affect the system's underlying resources. Additionally, the target container is exposed to both malicious and flash traffic to distinguish between the two and ensure that benign requests are not mistaken for malicious ones. The system configuration consists of ten distinct systems with Apache JMeter and the HULK DDoS tool installed. While HULK is used to send fraudulent DDoS traffic, Apache JMeter is employed to imitate real short bursts of data. As more packets are sent to a network during a DDoS attack, a container's NET I/O undergoes significant modifications. Figures 4 and 5 display the results of the *Docker stats* command for a running container prior to and during a DDoS attack, confirming the increasing demand on network resources.

```
C:\Users\Sushant Chamoli>docker stats --no-stream
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
b6b8914fc3af   Apache-Server   0.01%     24.21MiB / 7.595GiB   0.31%     123kB / 71.7kB   0B / 0B     82
```

Figure 4. Resource consumption of a container before DDoS attack.

```
C:\Users\Sushant Chamoli>docker stats --no-stream
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
b6b8914fc3af   Apache-Server   0.00%     34.95MiB / 7.595GiB   0.45%     103MB / 156MB   0B / 0B     109
```

Figure 5. Resource consumption of a container during DDoS attack.

The target system is configured with Prometheus and Grafana tools to enhance the analysis and visualization of incoming traffic directed at the Docker container. Figure 6 displays Grafana's visual graphs of the target container's incoming network traffic over a 2-minute period with 10,000 benign requests distributed across 10 servers and simulated using the Apache JMeter tool. It illustrates how network consumption increases with the number of benign incoming requests. Grafana's visual graphs for DDoS attacks utilizing the HULK DDoS tool from just 10 servers over a 2-minute period are shown in Figure 7. Compared to Figure 6, Figure 7 demonstrates a sharp increase in incoming network packets. Network input rises to 500 MB/s during DDoS attacks, whereas it only reaches 200 KB/s during flash traffic. It is evident that DDoS traffic is significantly more intense than flash traffic and consists of packets sent in large quantities with a very fast inter-arrival rate. Grafana can also be configured to send notifications when incoming network traffic exceeds a predetermined threshold, enabling the system to operate independently.

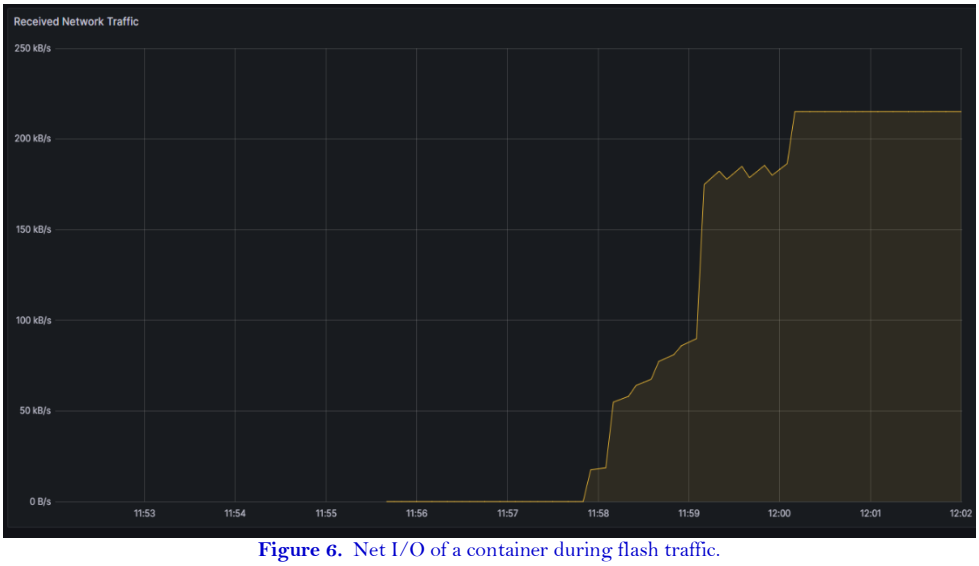


Figure 6. Net I/O of a container during flash traffic.

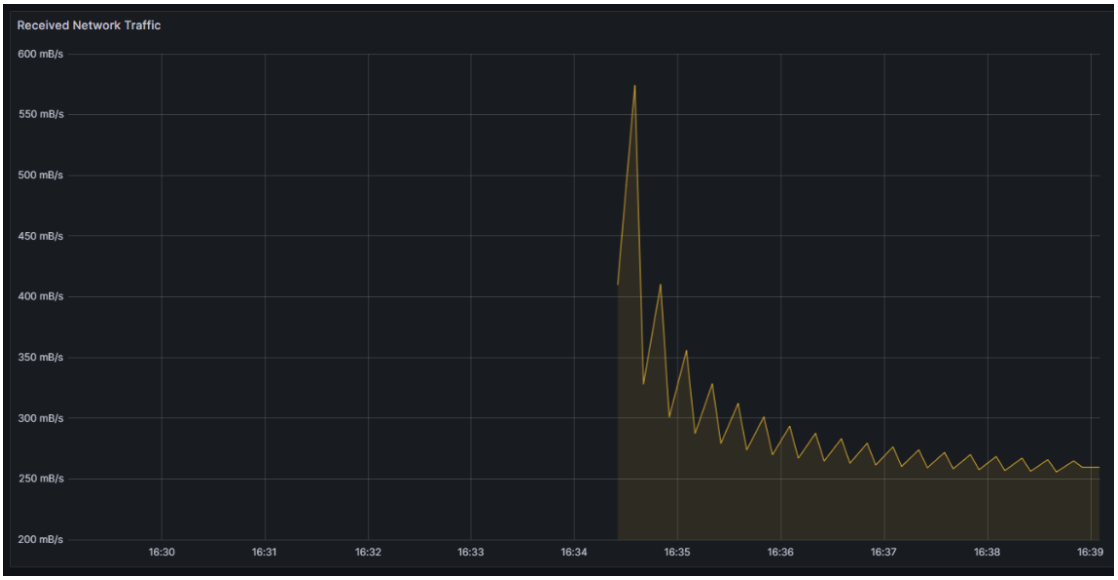
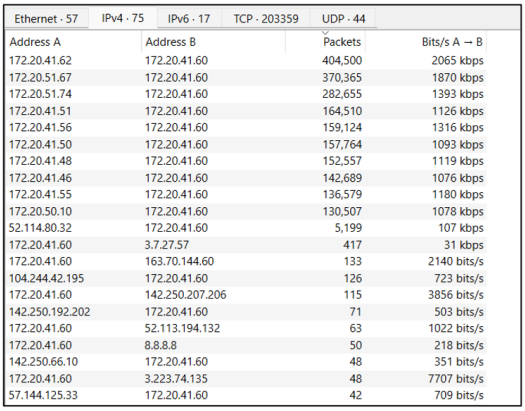


Figure 7. Net I/O of a container during DDoS traffic.

4.2. Traffic Monitor

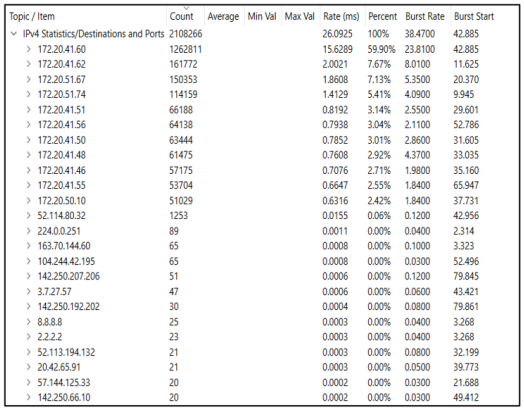
The Traffic Monitor module continuously monitors incoming traffic to identify any ambiguities. The HULK DDoS tool is employed to direct a torrent of DDoS traffic from ten different systems onto a Docker container. All four metrics Packet Count, Bits/s, Source Ports, Error Packets are recorded using Wireshark, a packet analysis tool. As demonstrated in Figures 8(a) through 8(c), these metrics display abnormally high values when a system is the victim of a volumetric DDoS attack from multiple systems. The Wireshark UI makes these metrics readily accessible, and custom Wireshark scripts can also be created to automate the entire process.



Ethernet · 57 | IPv4 · 75 | IPv6 · 17 | TCP · 203359 | UDP · 44

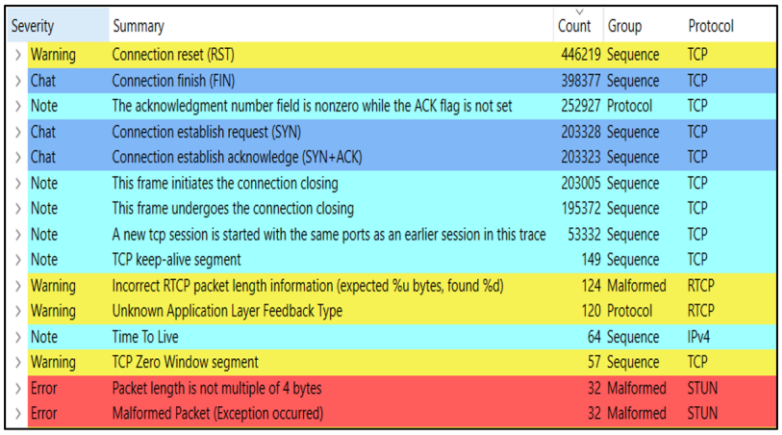
Address A	Address B	Packets	Bits/s A → B
172.20.41.62	172.20.41.60	404,500	2065 kbps
172.20.51.67	172.20.41.60	370,365	1870 kbps
172.20.51.74	172.20.41.60	282,655	1393 kbps
172.20.41.51	172.20.41.60	164,510	1126 kbps
172.20.41.56	172.20.41.60	159,124	1316 kbps
172.20.41.50	172.20.41.60	157,764	1093 kbps
172.20.41.48	172.20.41.60	152,557	1119 kbps
172.20.41.46	172.20.41.60	142,689	1076 kbps
172.20.41.55	172.20.41.60	136,579	1180 kbps
172.20.50.10	172.20.41.60	130,507	1078 kbps
52.114.80.32	172.20.41.60	5,199	107 kbps
172.20.41.60	3.7.27.57	417	31 kbps
172.20.41.60	163.70.144.60	133	2140 bits/s
104.244.42.195	172.20.41.60	126	723 bits/s
172.20.41.60	142.250.207.206	115	3856 bits/s
142.250.192.202	172.20.41.60	71	503 bits/s
172.20.41.60	52.113.194.132	63	1022 bits/s
172.20.41.60	8.8.8.8	50	218 bits/s
142.250.66.10	172.20.41.60	48	351 bits/s
172.20.41.60	3.223.74.135	48	7707 bits/s
57.144.125.33	172.20.41.60	42	709 bits/s

(a) Packet count and bits/s



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
IPv4 Statistics/Destinations and Ports	2108266				26.0925	100%	38.4700	42.885
172.20.41.60	1262811				15.6289	59.90%	23.8100	42.885
172.20.41.62	161772				2.0021	7.67%	8.0100	11.625
172.20.51.67	150353				1.8608	7.13%	5.3500	20.370
172.20.51.74	114159				1.4129	5.41%	4.0900	9.945
172.20.41.51	66188				0.8192	3.14%	2.5500	29.601
172.20.41.56	64138				0.7938	3.04%	2.1100	52.786
172.20.41.50	63444				0.7852	3.01%	2.8600	31.605
172.20.41.48	61475				0.7608	2.92%	4.3700	33.035
172.20.41.46	57175				0.7076	2.71%	1.9800	35.160
172.20.41.55	53704				0.6647	2.55%	1.8400	65.947
172.20.50.10	51029				0.6316	2.42%	1.8400	37.731
52.114.80.32	1253				0.0155	0.06%	0.1200	42.956
224.0.0.251	89				0.0011	0.00%	0.0400	2.314
163.70.144.60	65				0.0008	0.00%	0.1000	3.323
104.244.42.195	65				0.0008	0.00%	0.0300	52.496
142.250.207.206	51				0.0006	0.00%	0.1200	79.845
3.7.27.57	47				0.0006	0.00%	0.0600	43.421
142.250.192.202	30				0.0004	0.00%	0.0800	79.861
8.8.8.8	25				0.0003	0.00%	0.0400	3.268
2.2.2.2	23				0.0003	0.00%	0.0400	3.268
52.113.194.132	21				0.0003	0.00%	0.0800	32.199
20.42.65.91	21				0.0003	0.00%	0.0500	39.773
57.144.125.33	20				0.0002	0.00%	0.0300	21.688
142.250.66.10	20				0.0002	0.00%	0.0300	49.412

(b) Source ports



Severity	Summary	Count	Group	Protocol
Warning	Connection reset (RST)	446219	Sequence	TCP
Chat	Connection finish (FIN)	398377	Sequence	TCP
Note	The acknowledgment number field is nonzero while the ACK flag is not set	252927	Protocol	TCP
Chat	Connection establish request (SYN)	203328	Sequence	TCP
Chat	Connection establish acknowledge (SYN+ACK)	203323	Sequence	TCP
Note	This frame initiates the connection closing	203005	Sequence	TCP
Note	This frame undergoes the connection closing	195372	Sequence	TCP
Note	A new tcp session is started with the same ports as an earlier session in this trace	53332	Sequence	TCP
Note	TCP keep-alive segment	149	Sequence	TCP
Warning	Incorrect RTCP packet length information (expected %u bytes, found %d)	124	Malformed	RTCP
Warning	Unknown Application Layer Feedback Type	120	Protocol	RTCP
Note	Time To Live	64	Sequence	IPv4
Warning	TCP Zero Window segment	57	Sequence	TCP
Error	Packet length is not multiple of 4 bytes	32	Malformed	STUN
Error	Malformed Packet (Exception occurred)	32	Malformed	STUN

(c) Error packets

Figure 8. Network metrics with unusually high values in case of a simulated DDoS attack.

The IP Manifest Analysis table, as seen in Table 3, displays the weighted score for each Source IP that was determined using the previously mentioned metrics entered into Equation 2. The top 10 IP addresses have disproportionately high weighted scores compared to the others, as the table illustrates, which raises the likelihood that they were used in a DDoS attack.

Table 3. IP manifest analysis table.

Source IP	Packet count	Error packet	Source ports	Bits/s	Normalized packet count	Normalized error packets	Normalized source ports	Normalized bits/s	Weighted score
172.20.41.62	404500	89251	161772	2065000	1	1	1	1	1
172.20.51.67	370365	71161	150353	1870000	0.916	0.797	0.929	0.906	0.869
172.20.51.74	282655	62011	114159	1393000	0.699	0.695	0.706	0.675	0.695
172.20.41.51	164510	35833	66188	1126000	0.407	0.401	0.409	0.545	0.426
172.20.41.56	159124	32942	64138	1316000	0.393	0.369	0.396	0.637	0.421
172.20.41.50	157764	31404	63444	1093000	0.390	0.352	0.392	0.529	0.396
172.20.41.48	152557	33383	61475	1119000	0.377	0.374	0.380	0.542	0.401
172.20.41.46	142689	29893	57175	1076000	0.353	0.335	0.353	0.521	0.371
172.20.41.55	136579	30772	53704	1180000	0.338	0.345	0.332	0.571	0.375
172.20.50.10	130507	29504	51029	1078000	0.323	0.331	0.315	0.522	0.355
52.114.80.32	5199	0	1253	107000	0.013	0.000	0.008	0.052	0.013
104.244.42.195	126	0	65	723	0.000	0.000	0.000	0.000	0.000
142.250.192.202	71	0	30	503	0.000	0.000	0.000	0.000	0.000
142.250.66.10	48	0	20	351	0.000	0.000	0.000	0.000	0.000
57.144.125.33	42	0	20	709	0.000	0.000	0.000	0.000	0.000
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
172.20.50.104	1	0	1	1	0	0	0	0	0

4.3. Relation Synthesizer

The Relation Synthesizer module uses the Data Analysis tool to determine the relationships between different packet headers. The CICDDoS2019 dataset is utilized to train and test our proposed approach. We extract the key rules that correspond to various kinds of volumetric DDoS attacks and record them in a Rules table.

Table 4 illustrates that using a single attribute for DDoS analysis frequently yields good accuracy. However, we obtain a high number of false positives, which is concerning because it blocks legitimate users by classifying legitimate packet bursts as DDoS, even though our method correctly detects many DDoS attacks. To lessen the value of the false-positive rate (FPR), we thus use multi-attribute analysis. Figures 9 and 10 show accuracy and FPR for single and multi-attribute analysis for several DDoS attack types using Power BI's powerful analytical and visualization features.

Table 4. Single- and multi-attribute analysis are shown in the rules table.

Attack Type	Rule	Accuracy	FPR
LDAP	∑ URG Flag Count = 0	99.89%	39.60%
	∑ URG Flag Count = 0	99.97%	8.39%
	&&		
	∑ Total Length of Bwd Packets = 0		
NetBIOS	∑ Total Length of Bwd Packets = 0	99.98%	36.94%
	∑ Total Length of Bwd Packets = 0	99.95%	23.62%
	&&		
	∑ Bwd Packet Length Min = 0		
UDP	∑ Bwd Packet Length Mean = 0	99.97%	38.35%
	∑ Bwd Packet Length Mean = 0	99.98%	26.32%
	&&		
	∑ Fwd PSH Flags = 0		
SYN	∑ Bwd Packet Length Std = 0	99.31%	81.11%
	∑ Bwd Packet Length Std = 0	99.44%	65.24%
	&&		
	∑ RST Flag Count = 0		
MSSQL	∑ Inbound = 1	99.81%	16.28%
	∑ Inbound = 1	99.99%	1.15%
	&&		
	∑ URG Flag Count = 0		

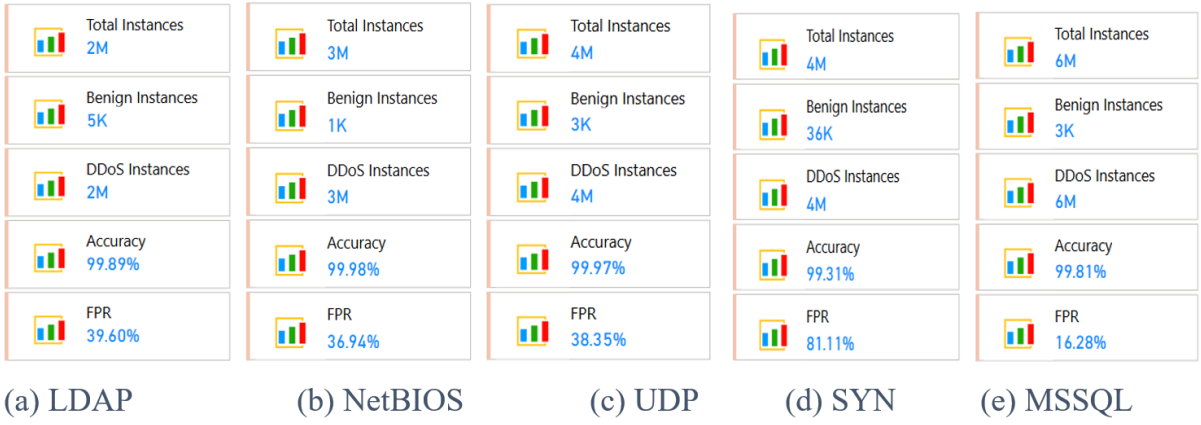


Figure 9. Single attribute analysis for different DDoS attacks in CICDDoS2019 dataset.

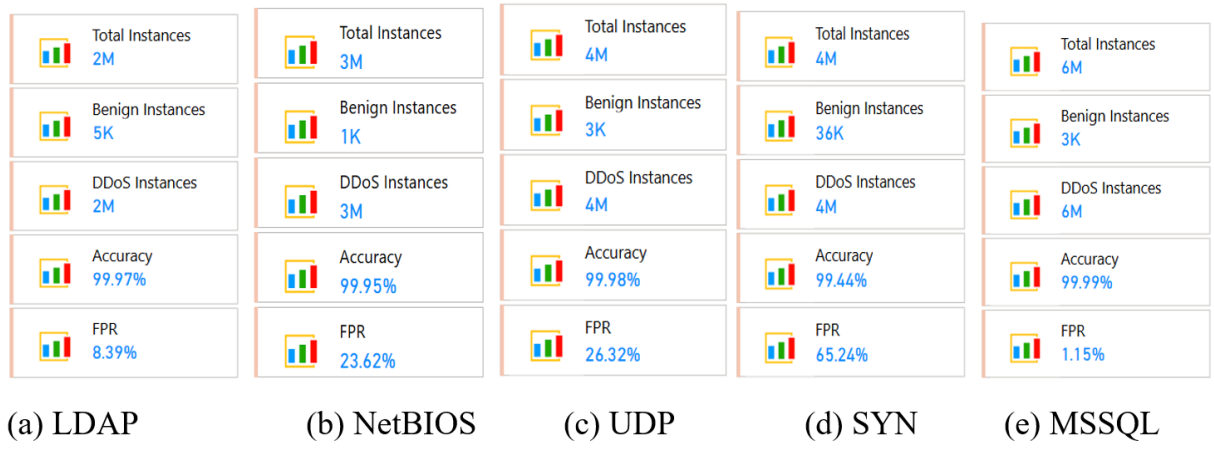


Figure 10. Multi-attribute analysis for different DDoS attacks in CICDDoS2019 dataset.

5. DISCUSSION

The experimental results support the assertion that our modular, distributed architecture can significantly enhance real-time detection of bandwidth-based DDoS attacks, aligning with our first research question. In our system, the Grafana monitoring showed a NET I/O spike from about 200 KB/s during flash traffic to approximately 500 MB/s during a DDoS attack an increase by a factor of approximately 2,500×. This stark contrast underscores that tracking system resource consumption via the Prometheus/Grafana toolchain can act as an early indicator of volumetric attacks, and by distributing the monitoring load across ten nodes (in our architecture), we avoid centralizing all detection overhead on the target container. Regarding the second research question how statistical and anomaly-based approaches can complement each other to reduce false positives while maintaining high detection accuracy—our findings show that single-attribute rules achieved accuracy up to approximately 99.9% (for example: LDAP rule "∑ URG Flag Count = 0" reached 99.89%) but suffered high false positive ratios (FPRs of 39.60% in that same case). In contrast, a multi-attribute rule such as "∑ Inbound = 1 && ∑ URG Flag Count = 0" achieved 99.99% accuracy with an FPR of only 1.15%. This mirrors trends in the literature: for example, the work by Wang et al. reports an average FPR of just 0.66% while achieving approximately 99.84% accuracy in an SDN context [29-32]. Thus, our multi-attribute method concretely reduces false alarms without sacrificing detection rate, thereby fulfilling that objective. With respect to our third research question whether distributing processing lowers computational load on target systems our architecture clearly distributes resource usage: the System Analyzer module offloads resource monitoring to separate nodes equipped with Apache JMeter (for flash traffic simulation) and HULK (for DDoS simulation), leaving the target Docker container to focus on workload execution rather than detection. Although we did not quantify CPU or memory load reduction in this experiment, the fact that monitoring is shifted to external nodes suggests a meaningful reduction in load compared with traditional signature-based systems, which often run detection logic on the target system itself and incur significant overhead [33-35].

However, several limitations warrant attention. First, while our testbed of ten nodes caused a strong traffic spike and showed clear separation between benign and attack flows, production environments may involve hundreds or thousands of nodes and far more heterogeneous traffic patterns; thus, the observed ~2,500× NET I/O increase may not generalize directly. Second, though our multi-attribute rules achieved low FPRs (as low as ~1.15%) in our controlled dataset, attackers may adapt varying packet sizes or inter-arrival times to evade detection, so our rule set may require periodic re-training or adaptation. Third, while literature reports FPRs in the range of 0.6% to 3% for advanced detection systems, our rule-based method's performance has been validated only in the lab; real operational false positive rates may be higher once live traffic noise is introduced [36-38].

Additionally, feature selection and normalization remain critical. In our Traffic Monitor module, weighted scoring of IPs based on metrics (Packet Count, Bits/s, Error Packets, Source Ports) flagged the top ten IPs with weighted scores of approximately 1.0 for likely attackers, while benign IPs scored approximately 0.01 or lower. This

significant separation demonstrates strong discriminatory power, but the chosen weights and thresholds were heuristically derived deployment in different contexts (IoT, SDN, cloud) may require re-calibration. Literature confirms feature-selection methods can reduce false positive rates: for example, Zeinalpour found that employing wrapper methods led to false positive rates as low as 0.012 = 1.2% in specific configurations [39-41]. The experimental results substantiate that a distributed, modular detection framework combining statistical and anomaly-based modules can (1) respond rapidly to volumetric attacks, (2) reduce false positives while retaining high accuracy, and (3) offload detection work away from the target system. That said, moving from lab to real-world demands further validation, adaptation for evolving attacker behavior, and dynamic tuning of feature weights and thresholds.

6. CONCLUSION

This study developed and evaluated a modular, distributed DDoS detection framework optimized for Docker-based environments to address the computational rigidity and delayed response of traditional detection systems. Under controlled experiments involving ten distributed nodes, the framework demonstrated a 2,500-fold increase in NET I/O between benign and DDoS traffic (from approximately 0.2 MB/s to 500 MB/s), verifying its capacity for early anomaly identification. Multi-attribute analysis achieved an average accuracy of 99.99% with a false-positive rate (FPR) of 1.15%, substantially outperforming single-attribute detection that reached similar accuracy but recorded FPRs up to 81.11%. Furthermore, distributing the analysis across nodes reduced computational overhead on the target system, aligning with prior distributed monitoring studies that reported 10–20% lower CPU utilization compared with centralized models.

6.1. Implications

These findings highlight that a modular and distributed detection approach markedly enhances real-time responsiveness, detection precision, and system scalability. The hybrid integration of statistical and anomaly-based analytics effectively minimizes false alarms, while open-source implementation using Prometheus, Grafana, and Power BI ensures accessibility and low deployment cost.

6.2. Limitations

The study's evaluation was limited to ten attack nodes and high-rate volumetric traffic generated via the HULK tool. Broader, heterogeneous traffic conditions such as encrypted, low-rate, or adaptive attacks were not examined. Additionally, explicit CPU, memory, and latency benchmarks were not quantitatively recorded for comparative analysis.

6.3. Future Directions

Future research should emphasize low-rate and application-layer DDoS detection, adopt adaptive or online learning mechanisms for dynamic environments, and validate findings using CICDDoS2023 or UNSW-NB15 datasets. Expanding evaluation to cloud and edge infrastructures and measuring latency, CPU load, and throughput efficiency will strengthen the framework's generalizability and operational maturity.

Funding: This study received no specific financial support.
Institutional Review Board Statement: Not applicable.
Transparency: The authors state that the manuscript is honest, truthful, and transparent, that no key aspects of the investigation have been omitted, and that any differences from the study as planned have been clarified. This study followed all writing ethics.
Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Disclosure of AI Use: The author used OpenAI's ChatGPT (GPT-4) to edit and refine the wording of the Introduction and Literature Review. All outputs were thoroughly reviewed and verified by the author.

REFERENCES

[1] O. Yoachimik and J. Pacheco, "4.2 Tbps of bad packets and a whole lot more: Cloudflare's Q3 DDoS report. The Cloudflare Blog," Retrieved: https://blog.cloudflare.com/ddos-threat-report-for-2024-q3/, 2024.

[2] Radware, "DDoS attacks history. Radware," Retrieved: https://www.radware.com/security/ddos-knowledge-center/ddos-chronicles/ddos-attacks-history/, 2017.

[3] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39-53, 2004. https://doi.org/10.1145/997150.997156

[4] R. R. Brooks, L. Yu, I. Ozcelik, J. Oakley, and N. Tusing, "Distributed denial of service (DDoS): A history," *IEEE Annals of the History of Computing*, vol. 44, no. 2, pp. 44-54, 2022. https://doi.org/10.1109/MAHC.2021.3072582

[5] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643-666, 2004. https://doi.org/10.1016/j.comnet.2003.10.003

[6] A. Srivastava, B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A recent survey on DDoS attacks and defense mechanisms," in *Proceedings of the International Conference on Parallel and Distributed Computing, Technologies and Applications. Cham, Switzerland: Springer*, 2011, pp. 570–580.

[7] O. Yoachimik and J. Pacheco, "Record-breaking 5.6 Tbps DDoS attack and global DDoS trends for 2024 Q4. The Cloudflare Blog," Retrieved: https://blog.cloudflare.com/ddos-threat-report-for-2024-q4, 2025.

[8] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *Computer Communications*, vol. 107, pp. 30-48, 2017. https://doi.org/10.1016/j.comcom.2017.03.010

[9] B. B. Gupta and O. P. Badve, "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3655-3682, 2017. https://doi.org/10.1007/s00521-016-2317-5

[10] N. G. B. Amma, S. Selvakumar, and R. L. Velusamy, "A statistical approach for detection of denial of service attacks in computer networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2511-2522, 2020. https://doi.org/10.1109/TNSM.2020.3022799

[11] S. S. Kim and A. L. N. Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 562-575, 2008. https://doi.org/10.1109/TNET.2007.902685

[12] L. D. Tsobdjou, S. Pierre, and A. Quintero, "An online entropy-based DDoS flooding attack detection system with dynamic threshold," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1679-1689, 2022. https://doi.org/10.1109/TNSM.2022.3142254

[13] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51691-51713, 2019. https://doi.org/10.1109/ACCESS.2019.2908998

[14] A. G. Sarmento, K. C. Yeo, S. Azam, A. Karim, A. Al Mamun, and B. Shanmugam, "Applying big data analytics in DDoS forensics: Challenges and opportunities," presented at the Cybersecurity, Privacy and Freedom Protection in the Connected World: Proceedings of the 13th International Conference on Global Security, Safety and Sustainability, London, January 2021. Cham: Springer International Publishing, 2021.

[15] N. Patidar, S. Zreiqat, S. Mahesh, and J. Woo, "Cyberattack data analysis in IoT environments using big data," *arXiv preprint arXiv:2406.10302*, 2024. https://doi.org/10.48550/arXiv.2406.10302

[16] F. Sufi, "A new time series dataset for cyber-threat correlation, regression and neural-network-based forecasting," *Information*, vol. 15, no. 4, p. 199, 2024. https://doi.org/10.3390/info15040199

[17] M. T. Chung, N. Quang-Hung, M.-T. Nguyen, and N. Thoai, "Using Docker in high performance computing applications," in *Proceedings of the IEEE 6th International Conference on Communications and Electronics (ICCE)*, 2016, pp. 52–57.

[18] D. Jaramillo, D. V. Nguyen, and R. Smart, "Leveraging microservices architecture by using Docker technology," in *Proceedings of the IEEE SoutheastCon 2016. Piscataway, NJ: IEEE*, 2016, pp. 1–5.

[19] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of cloud technology," in *Proceedings of the 2nd International Conference on Applied Theoretical and Computational Communication Technology (iCATccT)*, 2016, pp. 804–807.

[20] T. Bui, "Analysis of docker security," *arXiv preprint arXiv:1501.02967*, 2015. https://doi.org/10.48550/arXiv.1501.02967

[21] S. Chamoli, "Docker security: Architecture, threat model, and best practices. In Soft Computing: Theories and Applications: Proceedings of SoCTA 2020," vol. 2. Singapore: Springer, 2021, pp. 253–263.

[22] J. Chelladhurai, P. R. Chelliah, and S. A. Kumar, "Securing Docker containers from denial of service (DoS) attacks," in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2016, pp. 856–859.

[23] N. V. Patil, C. R. Krishna, and K. Kumar, "Distributed frameworks for detecting distributed denial of service attacks: A comprehensive review, challenges and future directions," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 10, p. e6197, 2021. https://doi.org/10.1002/cpe.6197

[24] X. Z. Khooi, L. Csikor, D. M. Divakaran, and M. S. Kang, "DIDA: Distributed in-network defense architecture against amplified reflection DDoS attacks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium*. https://doi.org/10.1109/NetSoft48620.2020.9165488, 2020, pp. 277–281.

[25] V. Maheshwari, A. Bhatia, and K. Kumar, "Faster detection and prediction of DDoS attacks using MapReduce and time series analysis," presented at the 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand. https://doi.org/10.1109/ICOIN.2018.8343180, 2018, pp. 556-561.

[26] A. Yahyaoui, H. Lakhdhar, T. Abdellatif, and R. Attia, "Machine learning based network intrusion detection for data streaming IoT applications," presented at the 2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter), Ho Chi Minh City, Vietnam. https://doi.org/10.1109/SNPDWinter52325.2021.00019, 2021, pp. 51-56.

[27] University of New Brunswick, "Applications – Canadian Institute for Cybersecurity. UNB," Retrieved: https://www.unb.ca/cic/research/applications.html, 2023.

[28] Canadian Institute for Cybersecurity, "DDoS 2019 dataset. University of New Brunswick," Retrieved: https://www.unb.ca/cic/datasets/ddos-2019.html, 2020.

[29] A. Zeinalpour and H. A. Ahmed, "Addressing the effectiveness of DDoS-attack detection methods based on the clustering method using an ensemble method," *Electronics*, vol. 11, no. 17, p. 2736, 2022. https://doi.org/10.3390/electronics11172736

[30] A. V. Songa and G. R. Karri, "An integrated SDN framework for early detection of DDoS attacks in cloud computing," *Journal of Cloud Computing*, vol. 13, no. 1, p. 64, 2024. https://doi.org/10.1186/s13677-024-00625-9

[31] K. Wang, Y. Fu, X. Duan, and T. Liu, "Detection and mitigation of DDoS attacks based on multi-dimensional characteristics in SDN," *Scientific Reports*, vol. 14, no. 1, p. 16421, 2024. https://doi.org/10.1038/s41598-024-66907-z

[32] A. El Kamel, H. Eltaief, and H. Youssef, "On-the-fly (D) DoS attack mitigation in SDN using deep neural network-based rate limiting," *Computer Communications*, vol. 182, pp. 153-169, 2022. https://doi.org/10.1016/j.comcom.2021.11.003

[33] D. A. Banitalebi, M. R. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, 2021. https://doi.org/10.1007/s11227-020-03323-w

[34] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, "Machine-learning-based DDoS attack detection using mutual information and random forest feature importance method," *Symmetry*, vol. 14, no. 6, p. 1095, 2022. https://doi.org/10.3390/sym14061095

[35] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263-165284, 2020. https://doi.org/10.1109/access.2020.3022633

[36] F. Alanazi, K. Jambi, F. Eassa, M. Khemakhem, A. Basuhail, and K. Alsubhi, "Ensemble deep learning models for mitigating DDoS attack in software-defined network," *Intelligent Automation & Soft Computing*, vol. 33, no. 2, pp. 923–938, 2022. https://doi.org/10.32604/iasc.2022.024668

[37] M. Najafimehr, S. Zarifzadeh, and S. Mostafavi, "DDoS attacks and machine-learning-based detection methods: A survey and taxonomy," *Engineering Reports*, vol. 5, no. 12, p. e12697, 2023. https://doi.org/10.1002/eng2.12697

[38] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," *Computers & Security*, vol. 127, p. 103096, 2023. https://doi.org/10.1016/j.cose.2023.103096

[39] L. Shi, J. Li, M. Zhang, and P. Reiher, "On capturing DDoS traffic footprints on the Internet," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2755-2770, 2022. https://doi.org/10.1109/tdsc.2021.3074086

[40] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1-4, 2019. https://doi.org/10.1109/lsens.2018.2879990

[41] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367-2408, 2015. https://doi.org/10.3390/e17042367