İSTANBUL UNIVERSITY CERRAHPAŞA

ORIGINAL ARTICLE
Electrica 2023; 23(2): 160-176

Real-Time Transmission Control Protocol-Synchronize-Based Distributed Denial of Service Detection Framework Using Entropy Variations in Self-Coded Bot-Network Architecture

Beenish Habib, Farida Khursheed

Department of Electronics and Communication, National Institute of Technology, Srinagar, India

Cite this article as: B. Habib and F. Khursheed, "Real-time transmission control protocol-synchronize-based distributed denial of service detection framework using entropy variations in self-coded bot-network architecture," *Electrica*, 23(2), 160-176, 2023.

ABSTRACT

Among the recent network infrastructure proliferations, distributed denial of service (DDoS) attack continues to be one of the most severe security threats. It becomes difficult to detect and demarcate the high volume of data and a huge number of users in any conventional network. The network randomness, physical attacks, and DDoS attacks leave their imprint through bandwidth profile (throughput), latency (time duration), and network traffic information (metadata). These three parameters form the crux of any detection setup. In this paper, we propose a fast, cost-efficient, open-source, and effective real-time entropy-based DDoS detector that uses entropy variations of Transmission Control Protocol-Synchronize packets as a base for attack detection. The attack traffic is self-generated through a compromised Bot-System controlled by a Command and Control Server. This way, we analyze the actual representative characteristics of the attack pattern. Our DDoS detector not only detects the attack but also sends the contextual information to a registered email ID. The attack information provides required network traffic characterization for the threshold-entropy calculations and its mathematical modelling, all that in real time. We code the whole architecture in python. It provides an optimal detection sensitivity and enhances predicting an attack with less resource utilization.

***Index Terms*—**Botnet, distributed denial of service, email alert, entropy, intrusion detection system, python

I. INTRODUCTION

With most of the data and information on the internet, the world is now more prone to global data thefts, intrusions, and bandwidth bankruptcy. In the first half of 2020 itself, there was a 15% increase in the network traffic with 4.83 million distributed denial of service (DDoS) attacks globally [1]. The methodology of the DDoS attack and its area of impact got changed due to covid outbreak and the consequent lockdown. Attackers are now targeting healthcare, e-commerce, and education. More than 930 000 attacks took place in May 2020, making it the hardest-hit month of the year [2]. Distributed Denial of Service attack coefficient [3] has increased as attackers do not pay for the bandwidth, leading to its most significant theft globally. As of 2021 [4], DDoS topped the list of most frequent cybersecurity incidents.

These attacks have affected internet services and telephony, and emergency services like 911 [5]. The global botnet breaches in 2021 have even reached 4G-5G spectrum inculcating short message service (SMS) flooding, fake signal calls, and cell phone jamming [6]. The DDoS attack launched by a Bot-Network [7] makes the internet a more vulnerable place. Spam emails with links to various threats and security breaches are alarming.

To detect such attacks, we need an intelligent detection system. Every detection setup uses machine learning, artificial intelligence (AI), soft computing, or mathematical and statistical knowledge. The recent research has also contributed more to studying correlation functions, fuzzy estimators, regression functions, or linear predicting models [8,9]. There has been an advancement in machine learning and AI techniques in early anomaly detection [10,11]. The traffic parameters used in most of the detectors include packet length, inter-packet intervals, protocol used, IP address, and number of packets. In recent techniques, the other parameters for identifying malicious data [12] include distance functions (Euclidean, City-Block, and Mahalanobish),

Corresponding author:
Beenish Habib

E-mail:
benish_habib@yahoo.com or
beenish_05phd17@nitsri.net

Received: January 11, 2022

Revised: May 13, 2022

Accepted: June 20, 2022

Publication Date: October 7, 2022

DOI: 10.5152/electrica.2022.21188

correlation functions (Person, Spearman, and Kendall), or entropy functions (Shannon or Kolmogorov).

A lot of research work is now done on DDoS detection using Software Defined Networking (SDN). OpenFlow technology used in SDN enable multiple interfaces to use one single protocol [13,14].

These techniques, though robust, give a pre-defined environment with no means of further modifications. Moreover, most intrusion detection systems (IDSs) work for varied forms of attacks and not specifically for DDoS only. BRO (Zeek) requires a UNIX platform, Snort is difficult to deploy with attacks causing information overload, OSSEC have default rules, Tripwire cannot generate a real-time alert, and Suricata is slower in detection. Most of the IDS-IPS aim at reducing the spread of attack by blocking outgoing malicious traffic from the compromised system. But it gets undetected due to non-continuous updating. Also, many IDS-IPS systems have complex training setups, non-contextual alert systems, and high attack false positive rate (FPR) [15].

II. RELATED WORK

A lot of research is carried out to enable the security of legacy or traditional networks [16]. The DDoS attack continues to grow significantly in the time-space paradigm both in size and frequency, depriving users of computing services [17]. The need is to detect and mitigate the attack with low computational cost, complexity, and latency [18]. While traditional networks have security implementations with limited features, modern networks like OpenFlow enabled SDN infrastructure try to be quite effective.

Software defined networking is a modern and robust platform, but the control plane faces many security flaws. The centralized architecture also makes it prone to various threats and attacks. Recently, massive DDoS attacks have been targeted on the control plane [19], making it more vulnerable than any other computing architecture. To use it for DDoS attack detection does not seem to be a good option. Even if we work with distributed architecture with each router monitoring the anomaly and alarming the next, it increases the network complexity and burdens the individual routers [20]. Some of the individual routers and IoT devices further aid in conducting DDoS attacks.

In SDN architecture, intrinsic approaches are not considered feasible. They have the requirement of extrinsic solutions. Recently, SDN utilizing the machine learning approach has made advancements against these volumetric attacks [21]. Artificial Neural Network and eXtreme Gradient Boosting with hybrid machine learning-based algorithm based on Self Organizing Map and k-Nearest Neighbour increased detection accuracy [22]. Other machine learning techniques used for attack detection include association-based approaches (fuzzy association, multivariate correlation, apriori, sequence analysis), classification-based approaches (Naïve Bayes, C4.5, support vector machine, entropy), clustering-based approaches (outlier detection, fuzzy C-means clustering, k-mean), hybrid-based approaches (Wavelet & Singular Value Decomposition and genetic algorithm-based approaches.

The high detection accuracy and low FPR of these algorithms may seem appealing, but the training phase, threshold selection, and prediction analysis increase the complexity. The further challenge of these algorithms is selecting optimal features and their multiple combinations as inputs for traffic processing.

Mathematical modelling with probability distribution functions is further used to enhance the detection of DDoS attacks [23]. Inter-packet interval follows the power law and is modelled using pareto distribution [24]. On the other hand, lognormal distribution models the network traffic information better. The Kolmogorov-Smirnov gives an idea of distribution curves, and so does Pearson's chi-square test. ANOVA gives the analysis of variance. Such statistical approaches aid in detecting layer 3 and 4 DDoS (internet control message protocol, user datagram protocol, SYN, TCP) [25]. The general analysis has assumed attack packets to arrive in pareto distribution and benign traffic in Gaussian distribution [26]. Such demarcations in traffic patterns using mathematical modelling may seem appealing, but they have lower detection accuracy and high computational complexity.

The detailed literature review of various DDoS detection methods proposed and used in the past along with their constraints is given in Table I [27-40].

A. Motivation

From the literature survey, we find that there is the use of various traffic features, attributes, and variations for the classification of normal and attack traffic. Most of the implementations involves the use of machine learning approaches and the use of software-defined networking. Though they are efficient in accuracy scores,the process of feature extraction and training of the machine learning models, with multiple algorithms for attack detection in AI and the intricacy of control plane and data plane in SDNs, adds to the computational complexity in these setups. The other issues are high traffic overhead, detection delay, low accuracy, and no inbuilt alert system with high FPR and false negative rate (FNR).

Among all these mathematical and statistical approaches, entropy is the most extensively used approach for traffic classification in most DDoS detection frameworks [41-43]. We know that any kind of change or anomaly causes change in the traffic characteristics. The entropy variation in a DDoS attack was extensively discussed by Lakhina et al. [44].

Some features vary considerably, while some have no change at all. It was source IPs that showed considerable variations than destination IPs [45]. This became valuable information and was used as a standard to demarcate the attacks [46].

Using entropy as the main feature for attack detection is also more practical and accurate with a low FPR and FNR [47]. It is also extendible in checking various traffic attributes, that is, source/destination IP addresses, source/destination port, and even protocol [37]. The feature extraction and distribution also showed improved performance with respect to decreased processing overhead, redundancy, and less response time [48,49].

To use entropy for DDoS detection on a simple setup is the main focus of our work. We thus propose a lightweight DDoS detector based on entropy variations in a Bot-Network with an inbuilt alert system. We code the whole architecture in python with the use of minimum resources, keeping the robustness and accuracy intact.

1) Contributions

We summarise our contributions as follows:

TABLE I. THE EXISTING DDOS DETECTION TECHNIQUES LITERATURE SURVEY

Author & Year	Technique/Approach	Findings	Constraints
1. Stavros N. Shiaeles et al. 2012 [27]	Fuzzy estimator	DDoS detection based on mean packet arrival (3 sec detection window).	Near real time detection, computational overhead, no alert system, data import delay, not feasible for flash crowd attacks.
2. Singh et al. 2015 [28]	Captcha-based DDoS detection	Robust IP blacklisting.	Not a full proof remedy against bots, time consuming, not compatible.
3. Tamer F. Ghanem et al. 2015 [29]	Machine learning, AI, genetic algorithm	High detection accuracy (96.1%).	Large scale data set requirement, high online processing time, cannot detect normal DDoS attacks, non-real time system with no alert generation.
4. Sufian Hameed et al. 2016 [30]	Hadoop, HADEC, Big Data	Live DDoS detection framework.	Detection delay while transferring log file from capturing to detection server does not provide parallelism.
5. N. Hoque et al. 2017 [31]	FPGA-based detection through correlation measures	High detection accuracy (99.95%), low FPR (0%), low FNR (0.008%).	Dedicated hardware module requirement, works only on two class problems, correlation measures not sensitive enough to detect DDoS attacks.
6. Yonghao Gu et al. 2017 [32]	Machine learning, semi-supervised clustering	Increased detection convergence speed (14–20 ms) and accuracy (NA).	No real-time attack traffic used, each of the algorithm uses different subset of features, is complex, no alert system.
7. JieCui et al. 2019 [33]	SVM, SDN, cognitive computing	High detection rate (97.65%), low FPR (NA)	Mininet does not support real environment, control plane vulnerability to attacks, complex hash listing, SVM not suitable for large data sets.
8. D. Erhan et al. 2020 [34]	Matching pursuit and wavelet techniques	High detection efficiency (95.3–95.7%) with TPR (80–81%) using multiple characteristics of network traffic.	Suboptimal approximation, hybrid detection framework increases complexity, high number of attributes, training requires a high-end computational setup.
9. Raja Majid Ali Ujjan et al. 2020 [35]	Machine learning, SDN, S-flow technology with stacked auto encoders	High detection accuracy (95%), low FPR (less than 4%).	High computational complexity, centralized deployment makes it more vulnerable to attacks (single point failure).
10. Zengguang Liu et al. 2020 [36]	Reinforcement learning, DCNN, SVM	Low-rate DDoS detection in Edge environment.	Feature extraction complexity, dependence of Mirai botnet, low accuracy on sparse data, no alert system.
11. Raja Majid Ali Ujjan et al. 2021 [37]	SDN, entropy variation	High accuracy (94%), Llow overhead and redundancy, low FPR and FNR (6%).	High computational complexity with two classifiers, centralized deployment makes it more vulnerable to attacks (single point failure).
12. Hassan Mahmood et al. [38]	SDN, smart grids, entropy variation	Tsallis entropy-based defence mechanisms in SDN architecture to improve performance.	High response time, increase in CPU utilization due to controller, single controller vulnerability to bandwidth bottlenecks.
13. Khundrakpam Johnson Singh et al. [39]	MLP model, entropy variation	High accuracy (98.31%), high sensitivity (0.99) and specificity (0.056). EPA-HTTP Data set with (MLP) classification and genetic algorithm (GA).	Complex structural framework, High response time, limited to only application layer attack.
14. Frederico Augusto Fernandes Silveira et al. [40]	Machine learning, SDN, IOT	High detection rate (96%) and Low false alarm rate (0.8 -1.8).	High CPU utilization, only few attacks information present in the training database, complex framework (Wireless & IOT).

DDoS, distributed denial of service; FPGA, field programmable gate array; SVM, support vector machine; SDN, software defined networking; FPR, false positive rate; FNR, false negative rate; HTTP, hypertext transfer protocol; MLP, multi-layer perceptron.

1) A lightweight real-time entropy-based DDoS Detection mechanism is proposed, with better efficiency, low computational cost, and overhead.
2) Proximate real-time traffic pattern from a Bot-Network is used. The self-generated DDoS traffic data set improves detection capability.
3) It has an inbuilt email alert system. The alert with the detailed contextual information is sent to a registered email ID.
4) It uses both signature as well as anomaly detection techniques. For signature detection, collaborative threshold is used, and for anomaly detection, entropy variations determine the attack.
5) The resource utilization (CPU-RAM) is optimized without compromising the detection capability. It is highly accurate with low FPR, FNR, and low response time.

Rest of paper is organized as follows. The proposed detection methodology for DDoS attack is presented as Section III. The experimental setup and entropy calculation and its proposed implementation are discussed in Sections IV and V, respectively, with results and performance evaluation discussed in Sections VI and VII, respectively. The paper is concluded along with future scope in Section VIII. The References and Appendix A (python-coding) are added at the end.

III. PROPOSED DETECTION METHODOLOGY

Based on information source (host/network), analysis strategy (sign ature/anomaly/hybrid), time aspect (real-tTime/post-time), architecture (centralized/decentralized), response (active/passive), we must set IDS features to match our requirements. This section presents the proposed DDoS attack detection architecture.

We code our DDoS attacker which is a combination of Command and Control (C&C) Server and a Bot-Network, the bots being controlled by a C&C server. For generating the attack, the victim system is targeted with Transmission Control Protocol-Synchronize (TCP-SYN) packets. This TCP packet header causes the victim server to respond the SYN packet with the acknowledgment packet. Due to large number of bots and open connection, the victim is saturated of its resources and crash eventually.

For the identification and detection of attack, we use collaborative threshold value and packet header information. To further enhance our attack detection, we use generalized entropy (GE) calculation with Shannon Kolmogorov's complexity and extract other mathematical attributes of our network traffic data. The alert and the detailed information are sent to our registered email ID (with real-time traffic monitoring in the background). We code the compromised Bot-Network and C&C server, creating our own DDoS traffic pattern. This improves the detection efficiency and helps resolve issues of old data sets.

Our detection methodology is comprised of the following four modules (Fig. 1):

Module 1: DDoS attacker (C&C server and compromised Bot-Network)
Module 2: Network traffic collector—threshold and entropy calculation with DDoS detector
Module 3: Real-time traffic monitor—network performance monitor
Module 4: Email alert system

A. DDoS Attacker (C&C Server and Compromised Bot-Network)

To set up our DDoS attacker, we create a Bot-Network headed by a C&C Server (Fig. 2). A single compromised system is sufficient to cause the DDoS attack. With C&C server, which acts as a Botmaster:

1) It can recruit thousands of bots (infected systems) through a system vulnerability, a payload with an executable file sent or uploaded on a uniform resource locator (URL), through SMS service or even with email attachments.
2) It can carry the attack on a large scale in a very co-ordinated manner.
3) It can also establish multi-threading and multiprocessing executions with a set of connections and multiple command executions.

The IP address of the C&C server is kept hidden. In centralized approach, we control multiple bots in a single session.

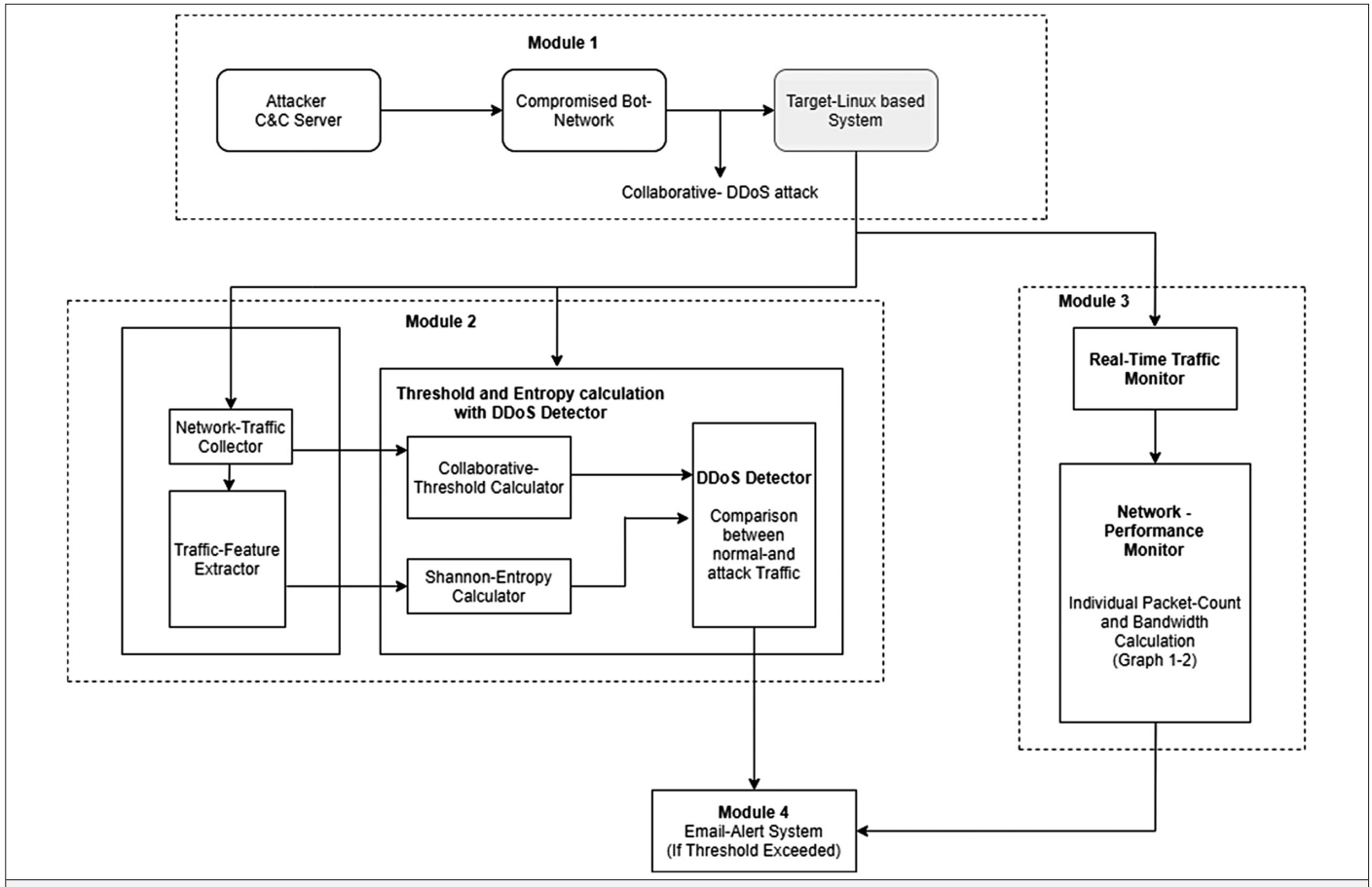

Fig. 1. Architectural framework—Proposed DDoS detector with real-time traffic monitoring and an inbuilt alert system.

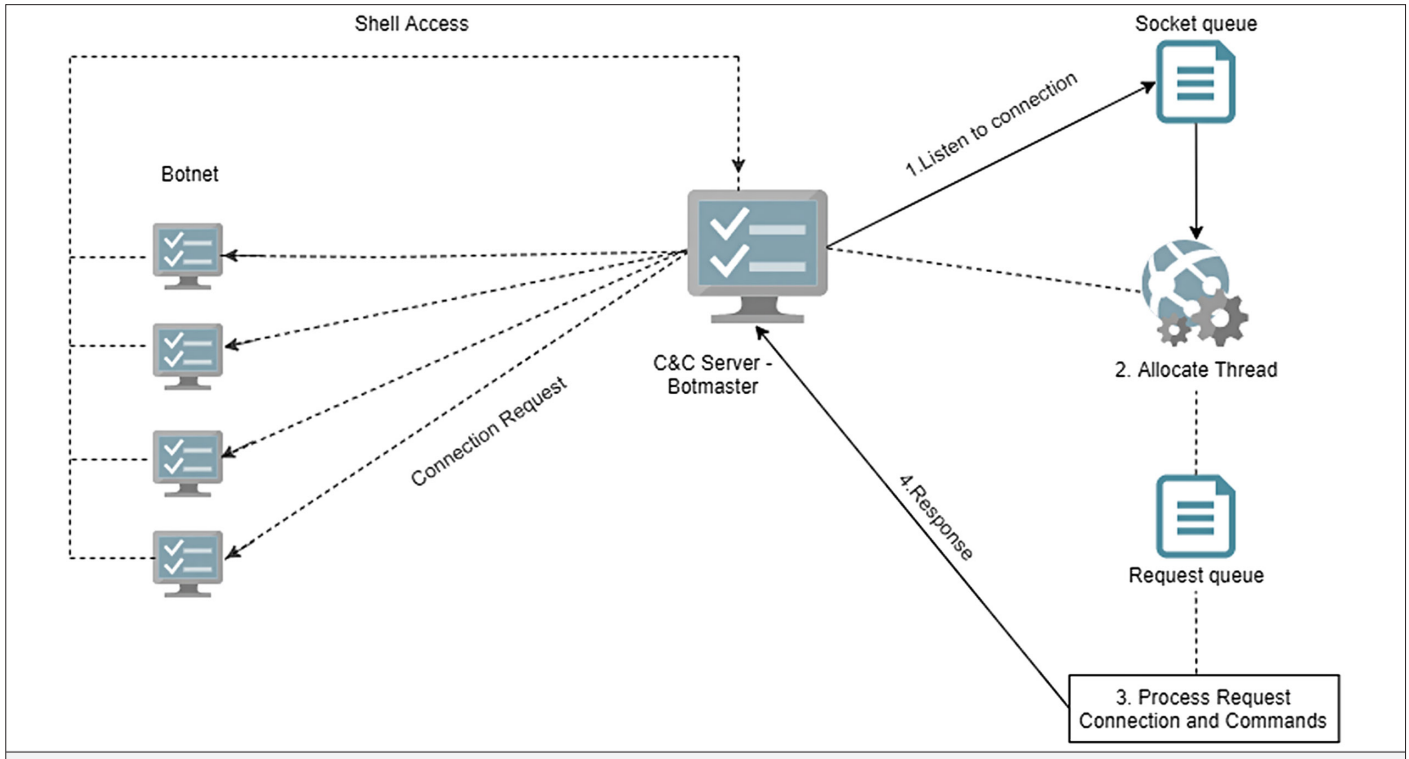

Fig. 2. Control of C&C Server on Bot-Network.

The bots are connected via the internet or even through an SMS gateway and can communicate via a secure protocol (Hypertext Transfer Protocol /Hypertext Transfer Protocol Secure, TCP/ Telnet Internet Relay Chat) or short messages. The Bot-Network can be in a single network domain or widespread over different networks.

B. Network Traffic Collector—Threshold and Entropy Calculation with Distributed Denial of Service Detector

We use Python-Scapy framework to design our network sniffer. The simple network traffic is collected for general threshold calculation. We collect the traffic packets with packet header information (SYN =1 and ACK = 0) and classify them as attack packets. A counter

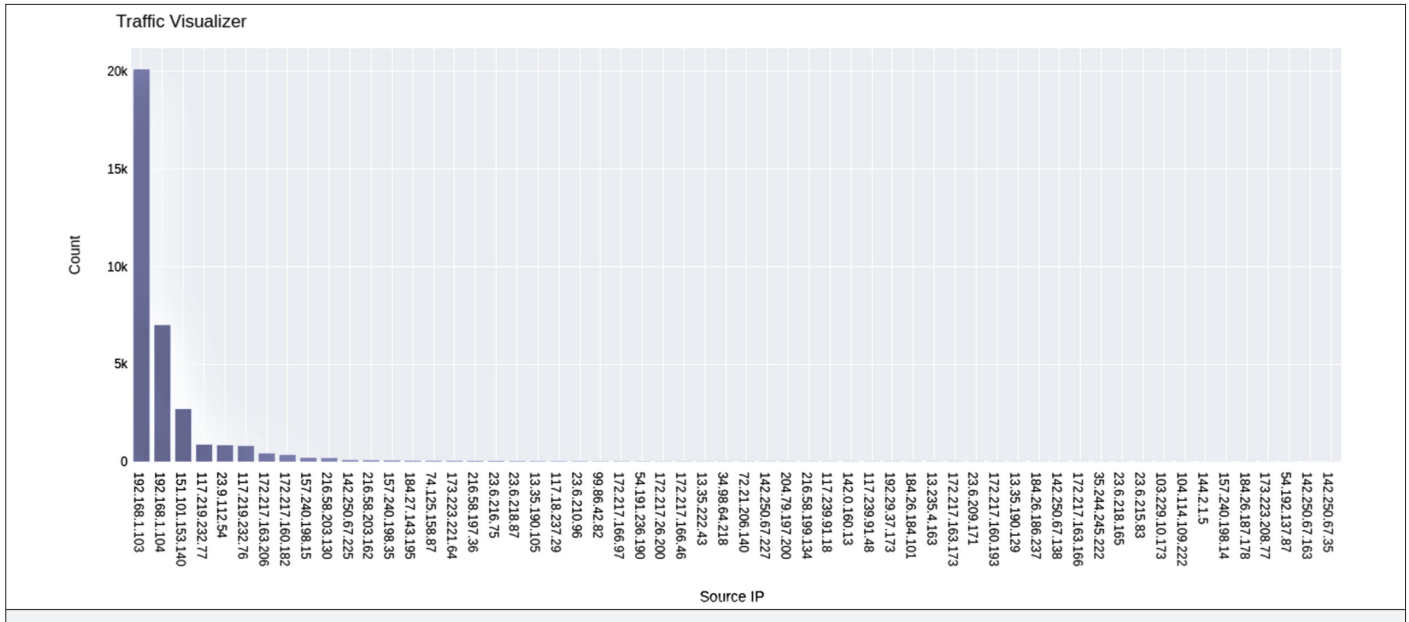

Fig. 3. Source IP count.

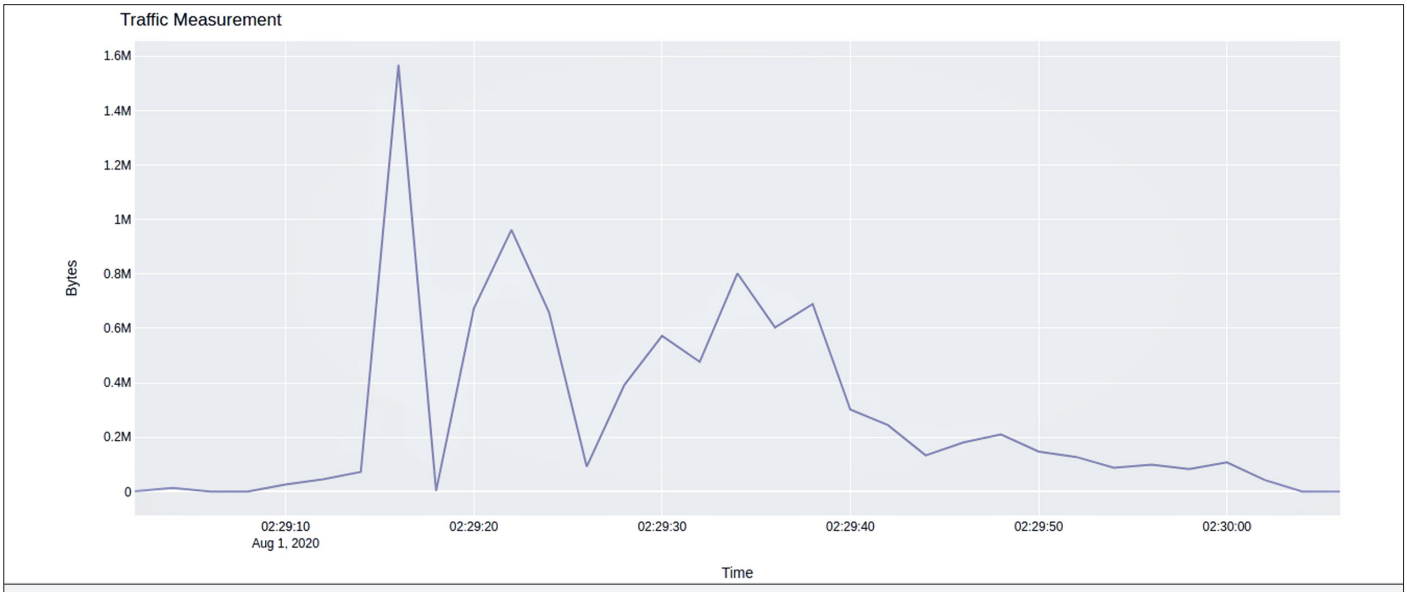

Fig. 4. DdoS—bandwidth utilization.

counts the packets, and for the number of packets exceeding the threshold, the detector verifies it as attack traffic. This forms a basic threshold detector based on the number of SYN packets.

To further enhance our detection efficiency, we use entropy as another traffic attribute for attack detection. We code the entropy calculator based on Source IP-Port to extract the network randomness. We can also use other traffic attributes like srcIP, dstIP, src Port, dst Port, src Bytes, dst Bytes, and protocol [6].

To make our detector lightweight, we select the source IP address. It is the most efficient attribute to detect the attack and also easy to extract. In some real-time setup, feature extraction becomes complicated with varied data rates, high packet diversity, and irregular data patterns. Setting up a proper entropy-based threshold becomes difficult.

To overcome this issue, we use collaborative threshold, that is, both general threshold and Shannon-Entropy threshold and check the individual performance. Both traffic attributes are easy to calculate and require minimum usage of resources.

C. Real-Time Traffic Monitor—Network Performance Monitor

A real-time monitoring system feature is added to our detector to analyze the traffic effectively in a continuous time frame and generate the alert if the threshold gets exceeded. We set the interface (ens33 of Virtual Machine (VM1)) and sniff the live traffic in an interactive mode. It is similar to plotting in *MATLAB*. While Plotly is used to generate offline graphs, Matplotlib gives the real-time traffic analysis.

Figure 3 shows the total network data in a particular time frame (bandwidth) and Fig. 4 is our traffic visualizer and plots the total count of packets sent by a particular IP. For our real-time traffic analysis, we have network performance monitor, which plots the incoming and outgoing network traffic in a continuous time frame.

Figure 3 gives us the insight of individual IP list and their individual packet count in particular time duration. Fig. 4 gives us the total amount of packets send in a continuous time frame.

D. Email Alert System

We add an alert feature to our IDS. While third-party internet security providers charge a lot for the same, we used an email service to make it cost free. Most of the IDSs generate alerts, which gets stored in their log directory. These alerts need to be constantly checked. To avoid this complexity, we set a registered Gmail account and get alert (Fig. 5) along with the details (Figs. 3 and 4) whenever a DDoS attack takes place. The email sent is encoded using a base64 encoder.

IV. EXPERIMENTAL SETUP

The experimental implementation involves two virtual machines on an Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz 1.80 GHz 16 Gb processor, a straightforward configuration with base hypervisor VMware Workstation 15.5 Pro. The structural framework is illustrated in Fig. 1. We need a C&C server, compromised bots, a real-time detector, an entropy-threshold calculator, and an inbuilt email alert

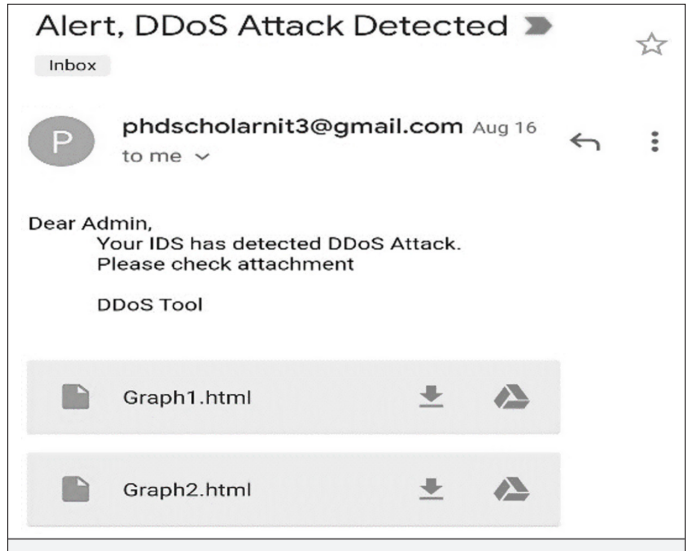

Fig. 5. Emailalert.

TABLE II. LIST OF EXPERIMENTAL PARAMETERS

Features	Description
IDE platform	PyCharm Professional
Python version	2.7 with updated PIP packages
Memory information	pcap (register database)
Python libraries	Socket, Queue, Threading and SYS, Scapy, Plotly, Datetime, Pandas, smtplib, SSL, Email, Matplotlib. Animation, PSUTIL, Multiprocessing, Collections/ counter {JSON format}, email-encoder, web-browser.
Packet capture standard	Libpcap
Packet encoding	base-64
Internet protocols	IPv4 - IPv6
Protocols (Layer 3/4, 7)	TCP, MIME
Supported alert format	Skype, short-messaging service, multimedia, live streaming

system to design our DDoS detector .VM1 is an Ubuntu (16-04-64 bit) operating system, and Virtual Machine 2 (VM2) is Kali-Linux OS with IP addresses 192.168.1.104 and 192.120.225.114, respectively. Virtual Machine 2, which is our attacking front, involves a C&C server and a compromised Bot-Network. The experimental requirements are given in Table II. The integrated development environment (IDE) platform and some library functions used are also detailed later.

Our C&C server comprises a host and multiple compromised bots with multiple connections. The details of the parameters required to code one are given in Table III.

The bots are compromised using a gaming payload with the shell access given to our C&C server. The payload is a simple python-coded snake game. It is an essential tool to set connections and execute an attack (Fig. 6). The gaming payload is uploaded with its URL on a website. While it is alluring to download the game and play, it creates a persistent connection (a connection that is established even after we stop the game) in the background. Crontab module is used to have a continuous-time schedule. This creates our Bot-Network, and the number of downloads sets the number of compromised bots.

TABLE III. THE DESCRIPTION AND VALUES OF VARIOUS PARAMETERS OF A C&C SERVER

Parameters	Description	Value (if any)
Socket ***S***	Determine the host and port number (TCP/IPv4)	Host =" for dDynamic IP Port =12345 {Random}
Commands ***cmd***	Set command line arguments	List, select, quit
Connections **C**	Accept IP addresses and bind them with host and port number	Heartbeat connection Hi and Hello
Threads & Jobs {***t, j***}	Connection and command execution	{1,2}
Queue {***q***}	FIFO-based counter to set a task	Put, get and join

C&C, command and control; TCP, transmission control protocol.

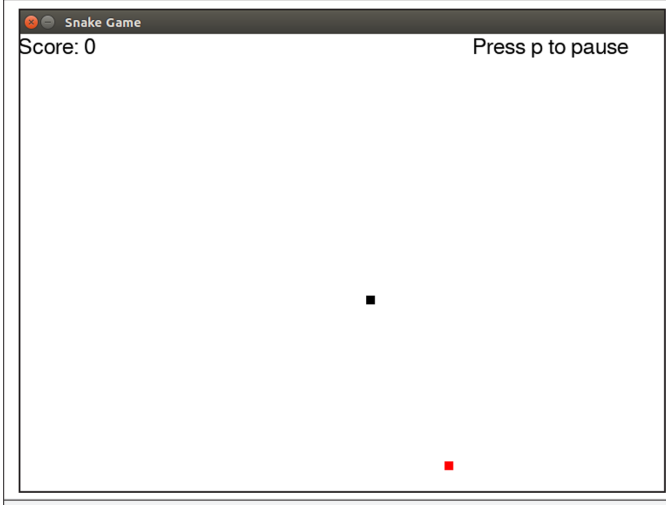

Fig. 6. Snake game.

The C&C server, which is our Bot-master, controls these bots. The vital thing to consider here is that a single vulnerability(security-breach) can establish a connection and disrupt the victim of its services by just a single download.

The shell access to the bots through C&C server sets a way to initiate the DDoS attack on our target IP (Fig. 7).

To collectively set the connections, execute commands (i.e., multi-threading), and launch the attack, we use parallel programming or multi-threading. This ensures that connections are established even after the game (payload) is closed. We set our sub-processes as shown in Fig. 8. This is our VM2.

A. Data Set

To code a DDoS generator, we use Python-Scapy framework. A random IP generator with source IP/Port and destination IP/Port is set in a continuous time frame (Fig. 9).

The other parameters as detailed in Table IV. Here we use TCP-SYN packets and the IP addresses are random which are sent over a continuous time-frame. The parameters (packet count/volume, time-duration, range of spoofed IPs, source IP/Port, destination IP/Port, victim IP) can be altered as per our own requirements.

This determines our legitimate real-time data set. We can also use emulated DDoS data sets (MIT Darpa/TUIDS/ CIC-DDoS2019/hping3/ Hyenae 0.36), but coding a real-time data set gives us the flexibility of modifying the parameters as per our requirement.

V. ENTROPY CALCULATION

Our work involves entropy-based DDoS attack detection methods and its implementation in a simple coded Bot-Network. Entropy is the statistical measure of randomness of data. For DDoS attack, the Bot-Network sends a huge number of similar packets to the victim, which in turn decreases the randomness parameter of the network traffic. For flood attack, where spoofed source-IP based packets are used for attack, the randomness increases. These fluctuations that cause change in entropy pattern is a clear indication of an attack. This randomness can also be calculated using different packet fields. We consider the most affected packet field, that is, source IP address

```
Shell > Connection has been established from IP: 192.168.225.51 Port 46178

list
Not a Valid Command!
Shell > Connected Clients:
ID  |   IP Address    |  Port number
0    192.168.225.51   46178

Shell > select 0
you are now connected to 192.168.225.51
192.168.225.51> cd /tmp
/tmp> ls
ddos
_MEInuJmpW
_MEIQa9Rjy
persistence
ssh-YLsdYeFmRfih
systemd-private-5de56bbb532445a484c5b83b0f3849c4-bolt.service-EcXoKz
systemd-private-5de56bbb532445a484c5b83b0f3849c4-colord.service-S6HlH9
systemd-private-5de56bbb532445a484c5b83b0f3849c4-fwupd.service-BDechq
systemd-private-5de56bbb532445a484c5b83b0f3849c4-haveged.service-nb3KvA
systemd-private-5de56bbb532445a484c5b83b0f3849c4-ModemManager.service-OosZ8D
systemd-private-5de56bbb532445a484c5b83b0f3849c4-rtkit-daemon.service-QhnPTg
```

Fig. 7. Shell access to the Bots.

for our attack determination, although multiple packet fields could be checked at the same time without affecting the performance of our detector (Fig. 10). Here the advantage of using this simple method of attack determination is that there is no increase of any network traffic, memory and CPU overhead is also very less, almost negligible.

The source and destination IPs both form a probability distribution function [25].

$$p = \{x_1, x_2 ..... x_n\} \tag{1}$$

$$\text{with} \sum_{i=1}^{\eta} x_i = 1, 1 \geq x \geq 0, i = \{1, 2 ..... n\} \tag{2}$$

To calculate information entropy value, we use:

$$H_\Upsilon(\chi) = \frac{1}{1-\Upsilon} \log_2 \sum_{i=1}^{n} p_i^\Upsilon \tag{3}$$

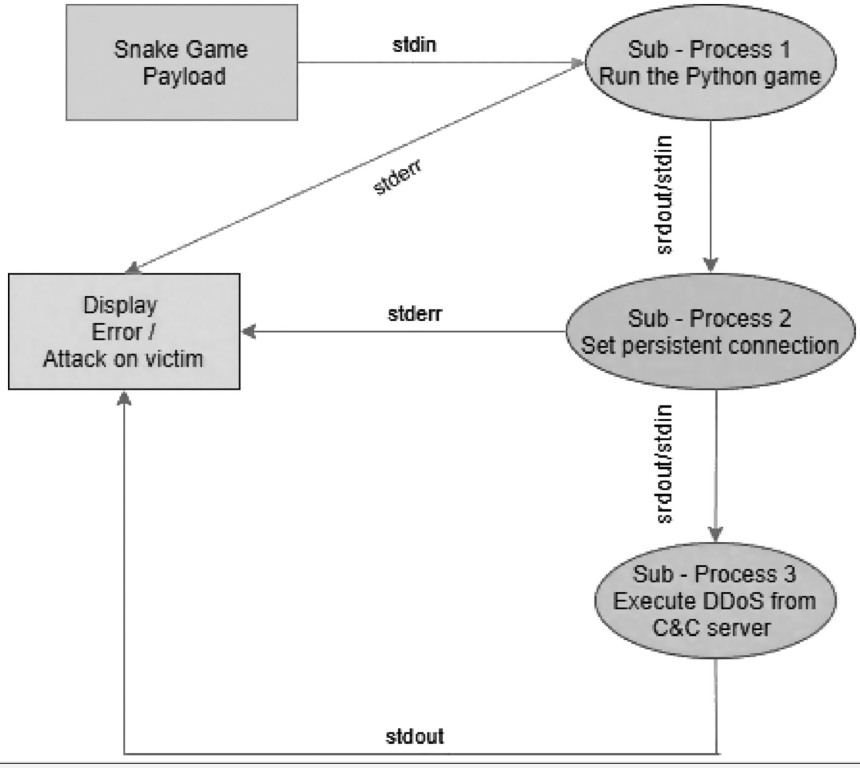

Fig. 8. Multi-processing in a Bot-Network.

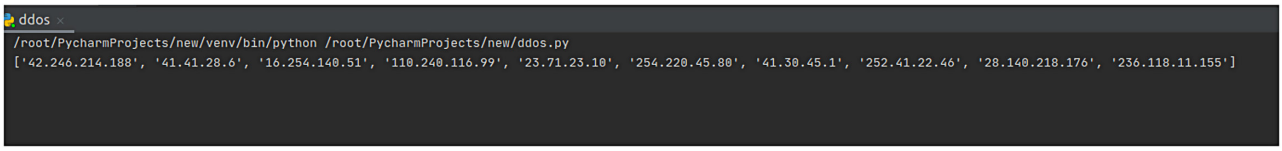

Fig. 9. Random IP generator.

TABLE IV. THE DESCRIPTION AND VALUES OF VARIOUS PARAMETERS IN A CODED DDOS ATTACK

Parameters	Description	Value (if any)
ip_list	Determine the list of random IPs generated	Any IP address
target_ip	Victim where attack is to be launched	Victim IP
src,dst {ip1}	Source and destination IP	Source_ip, target_ip
sport,dport{tcp1}	Source and destination Port	s_port,80
Pkt	ip1/tcp1	-

Assertion 1: When $\Upsilon = 0$, i.e., with equal probabilities.

$$H_0(P) = \log_2 \eta \tag{4}$$

Assertion 2: When $\Upsilon = 1$

$$H_1(P) = -\sum_{i=1}^{n} p(x_i)\log_2(x_i) \tag{5}$$

To model it for DDoS detection, we calculate the source IP address-based probability distribution function for different network traffic patterns and their average entropy values for different timeframes, respectively.

A. Proposed Implementation

For our detector we need to determine various attributes and features. We have VM1, where we code the attack detection using general threshold and entropy variation with inbuilt email alert system and real-time traffic monitoring,

1) **Incoming packets:** We use T-shark to extract the source-IP list from packet capture (PCAP) file and store in a csv format to count the individual IPs. This sets as our packet inspection.
2) **Window size:** Window size determines the time frame and thus is proportional to packet volume and number of packets. The window size can be modified as per our requirement.
3) **Threshold set-up:** We set up the threshold reference value ourselves to decide the possibility of an attack. For some

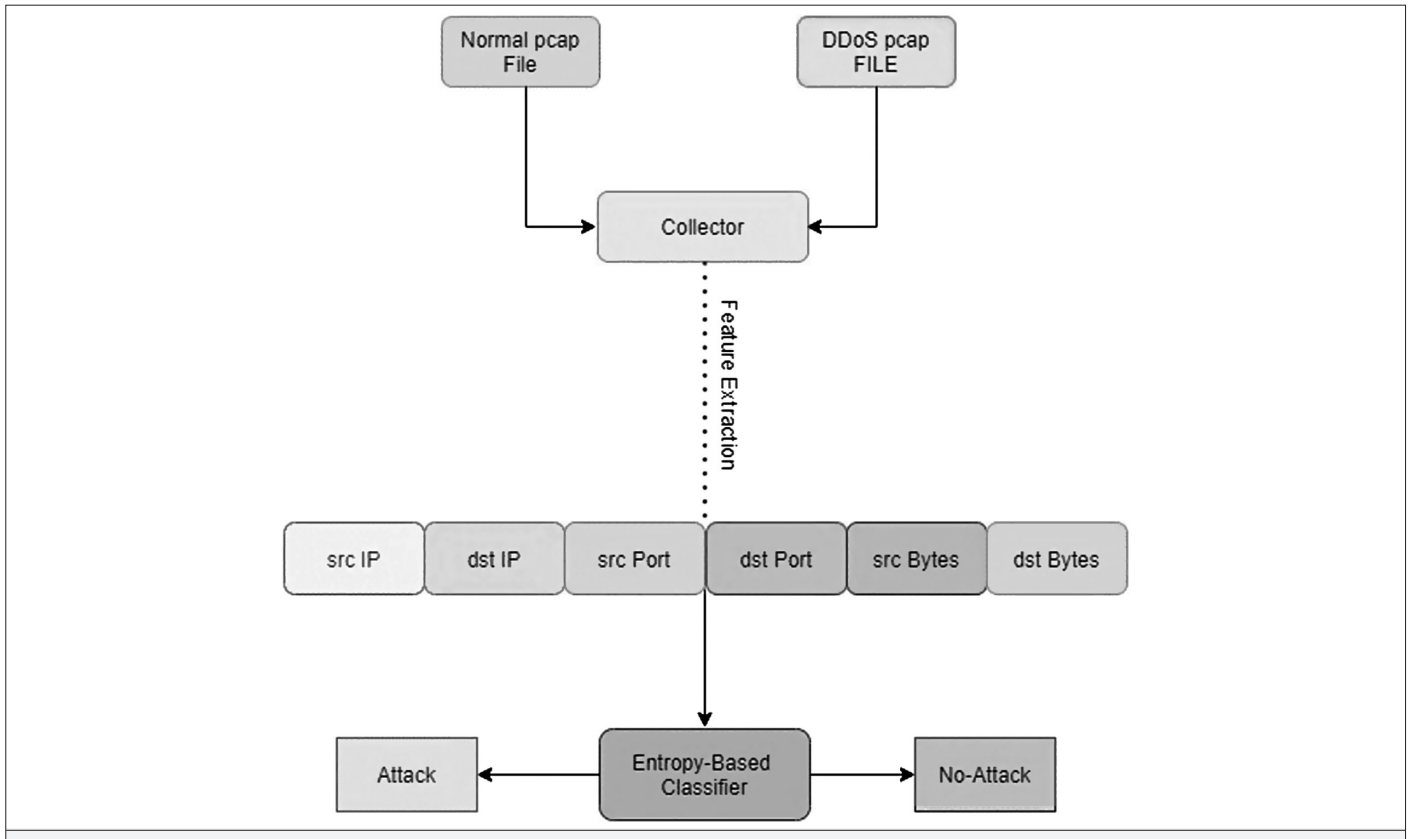

Fig. 10. Entropy classifier.

setups, it is pre-defined and for some it is determined during run-time. Both have their individual importance and usage. We calculate the parameters during a normal network traffic and set that as our threshold value. Any deviation from the normal pattern, gives us the indication of an attack. For the basic threshold calculation, based on number of TCP-SYN packets we select number of packets generated during normal internet streaming (in our case, data pattern of more than 25 SYN packets/s determines the attack). This sets our detection rule. The frequency of each IP address is used for the probability calculation. We get the average entropy after normalization to be used in Shannon code.

4) **Real-time monitoring:** To have the real-time monitoring, we code a traffic monitoring visualizer which uses the date-time module (inbuilt in python) to give the insight of both incoming and outgoing traffic pattern.
5) **Other traffic parameter characterizations**: We also find change in packet size, inter-packet interval, mean and variance with respect to entropy variations. These parameters are also analyzed for the attack determination.

We calculate the values of parameters for the network traffic. For the network traffic with number of packets exceeding the general threshold and with entropy threshold value $\vartheta nA < \vartheta n^T \| \vartheta nA < \vartheta n_0$, we predict the traffic as attack traffic. Here ϑn^T is an entropy threshold value for benign network traffic, and ϑn_0 is an entropy threshold value for no-network traffic (Fig. 11).

We specify our time slot (3–7 minutes) and, specific to that, check sudden rise or drop in number of packets and entropy values, respectively, compared to the predefined threshold value. The packets are captured, and details are stored in a PCAP file. The PCAP file is graphed and sent to a registered email ID alerting us of a possible DDoS attack.

VI. RESULTS

The main aim of our proposed work is to detect the DDoS attack with improved accuracy and low computational usage. We mainly rely on threshold entropy calculations of TCP-SYN Packets. Python is our scripting tool and various library functions (Table V) are used to design and code a real-time DDoS detector with an email alert system. This approach increases the efficiency as well as detection accuracy rate. We tried to decrease the resource utilization as much as possible, keeping intact the detection accuracy.

For a particular window size, we use the number of packets received. The entropy values are calculated depending on the number of packet occurrences within the same window size. If the number of

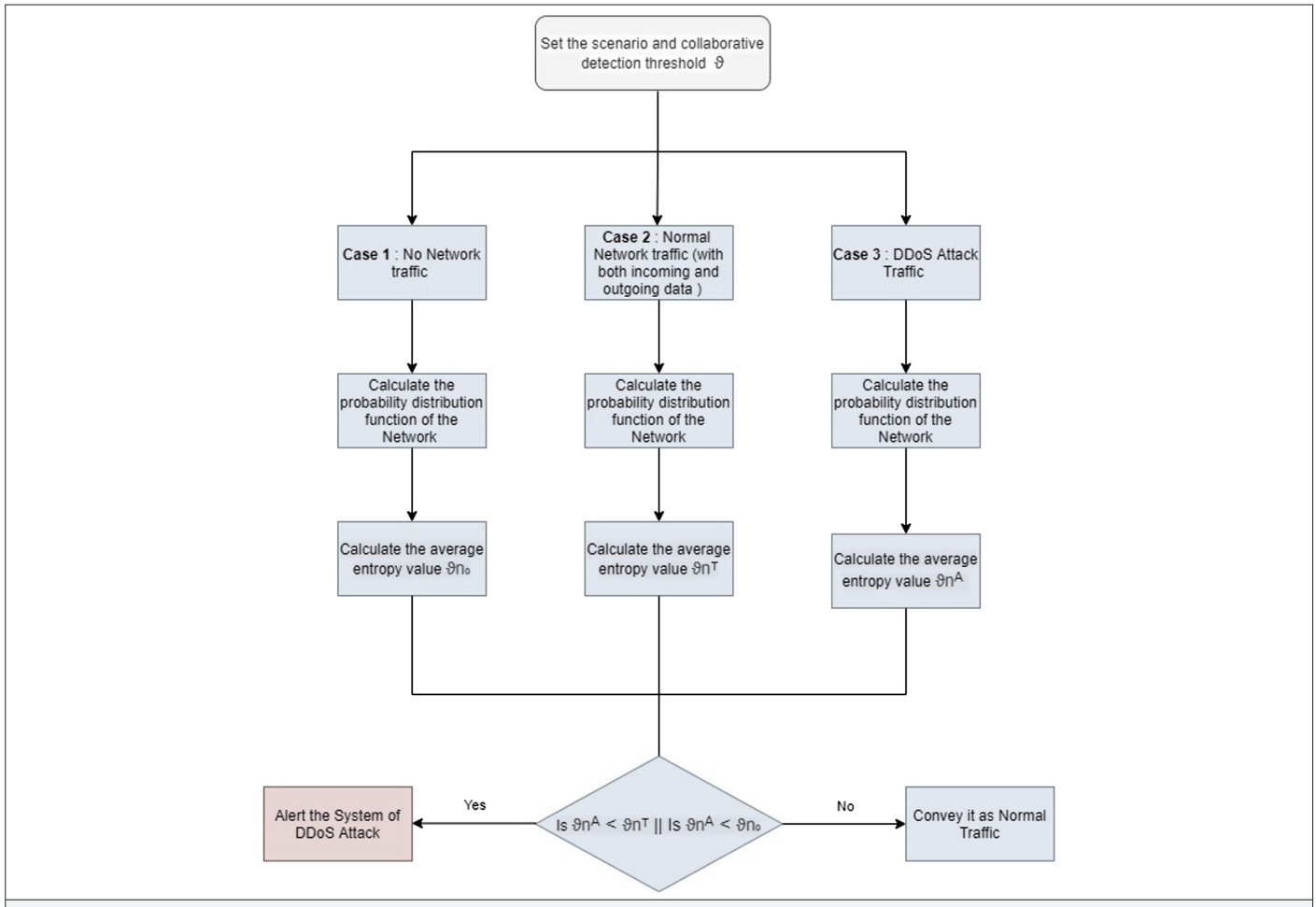

Fig. 11. Processing flowchart of the collaborative entropy-based detection algorithm in DDoS attack detection system.

TABLE V. THE DESCRIPTION OF VARIOUS SYMBOLS AND PARAMETERS USED IN MATHEMATICAL MODELLING

Notation	Description
Υ	*Order of the system* $\Upsilon \geq 0, \Upsilon \neq 1$
p	*probability distribution function*
$Log_2\eta$	*maximum probability function*
$H_\Upsilon(\chi)$	*Entropy value of probability distribution function formed by source/destination IPs with order* (Υ)
χ	*Random variable representing Source/Destination IPs*
η	*Total number of possible IP values*

frequencies for each IP is equal, then maximum entropy is established. If there is a sudden deviation (during an attack), then the entropy value is minimum as a single IP sends the attack traffic, thus decreasing the deviation or randomness (Fig. 12). For flood attack, the entropy value increases suddenly. Any deviation as such determines the possibility of an attack.

We use threshold values as our base for demarcation (Table VI). The distributed features are processed with the three base time slots, and entropy deviation beyond the average threshold entropy of benign or no-network traffic determines our attack. The details are sent to a registered email-ID in case it has exceeded the threshold limit. The PCAP file generated gives the flexibility to analyze network traffic at our convenience.

We calculate the most common attributes from the flows like packet count, source IP, destination IP, packet size, time, and date. For our network performance monitor, we use these details for both incoming and outgoing network traffic (Fig. 13).

TABLE VI. AVERAGE ENTROPY VALUES FOR VARIOUS TRAFFIC PATTERNS IN DIFFERENT TIME FRAMES

Time Slot (TW)	Average-Entropy No-Network Traffic	Average-Entropy Benign traffic	Average- Entropy Attack Traffic
3 minutes	0.7901	0.6478	0.3810
5 minutes	0.7806	0.6518	0.3426
7 minutes	0.7899	0.6437	0.0310

While we analyzed the normal and attack data, we found that other parameters also show significant variations during an attack. They are packet size, inter-packet interval, mean, and variance with respect to entropy variations. These data-information metrics and their deviations can further aid in detecting DDoS attacks to improve the detection performance and decrease the processing time, though we have only used entropy variations as base of attack determination to keep our detector lightweight.

A. Packet Size

The packet size distribution varies significantly in different protocols and different traffic patterns. In a DDoS attack, the packet size of some may even cross over to 1600 bytes or even more, but the majority are in less packet size range (Fig. 14).

B. Inter-packet Interval

Another feature to check is inter-packet interval. A vast majority of DDoS attack traffic have close to zero inter-packet intervals (ΔT) and high first and second derivatives of inter-packet intervals. Using ΔT, ∂ΔT/dt and ∂2ΔT/dt2 as features allows a classifier to capitalize on this difference between normal and DDoS traffic (Fig. 15).

C. Mean and Variance

From mean–variance variation data in attack and normal traffic [50] for different source IPs, we infer that the mean is low and

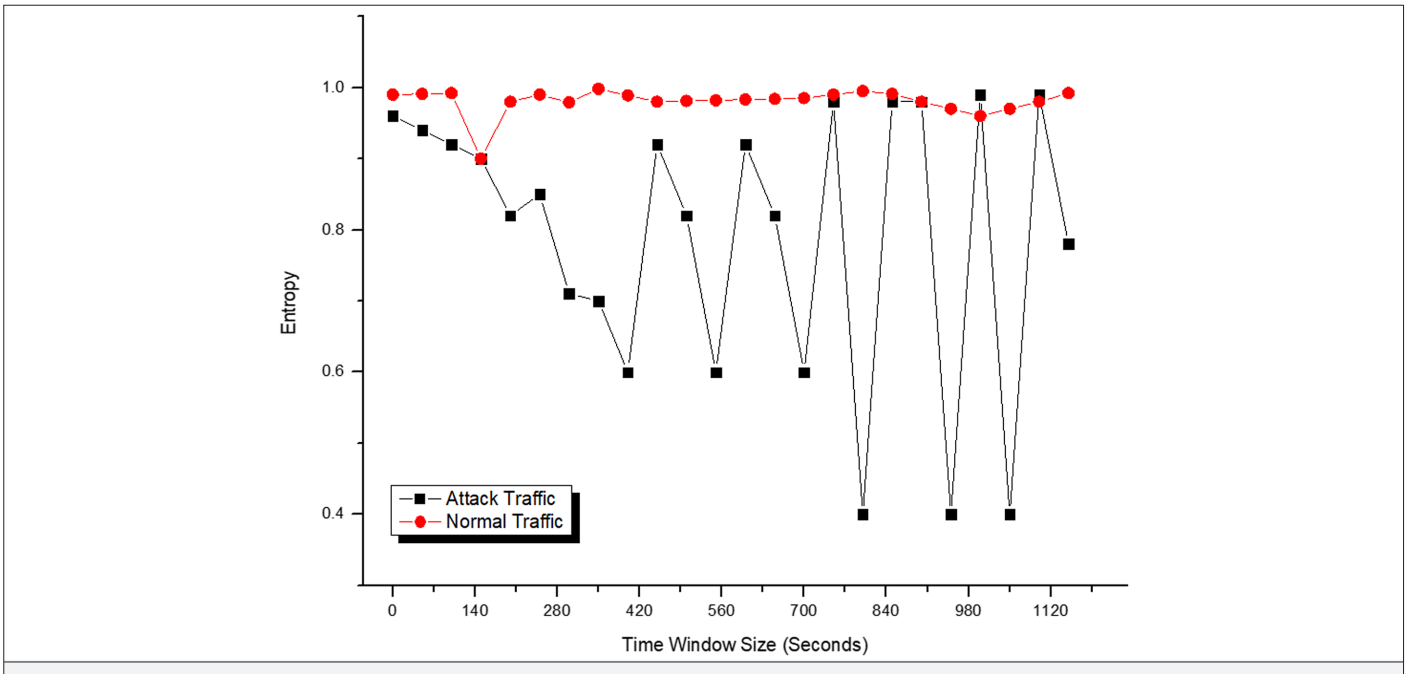

Fig. 12. Shannon—Entropy variation (a) attack traffic and (b) normal traffic

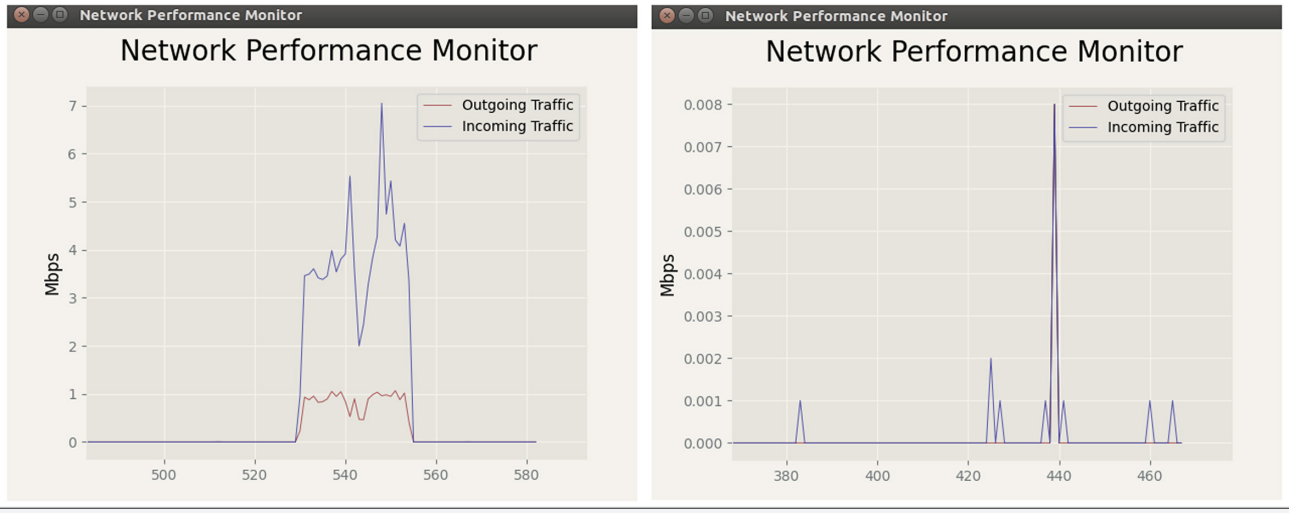

Fig. 13. Network performance monitor.

variance is high for an attack traffic. Also, the normal data pattern shows low variance approximating 0, while the variance is slightly high for attack data (Tables VII and VIII). The calculations are taken with respect to entropy variations. The entropy variation is high in IP (192.120.225.114) and thus has high variance than the rest. The same results are obtained in [39], but the limitation of the proposed method is that it is limited to application layer attack. The method also has a high response time and complexity in training the neural network.

To generalize, these parameters vary when a system is under a DDoS attack, and any deviation from the normal values predicts its occurrence. Table IX gives us the idea of these variations in normal traffic and DDoS traffic.

VII. PERFORMANCE ANALYSIS

For the performance analysis of our DDoS detector, we use well-known metrics and parameters such as *accuracy, FPR, sensitivity, specificity, precision, recall, F1 score*, *CPU-RAM utilization, and response time*.

We use confusion matrix and AUC curve to calculate these parameters (Fig. 16). We have 779 true positives, 193 true negatives, 6 false positives, and 14 false negatives. The accuracy score and other performance metric scores are given in Table X.

We also performed the CPU utilization analysis to check the efficiency of our DDoS detection system.

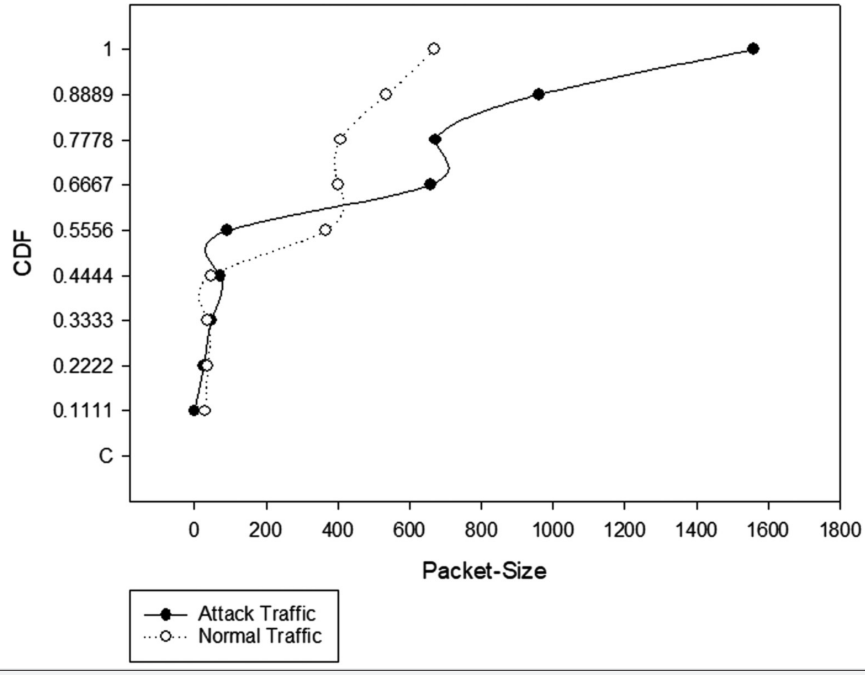

Fig. 14. Cumulative distribution function vs. packet size variation (a) attack traffic and (b) normal traffic.

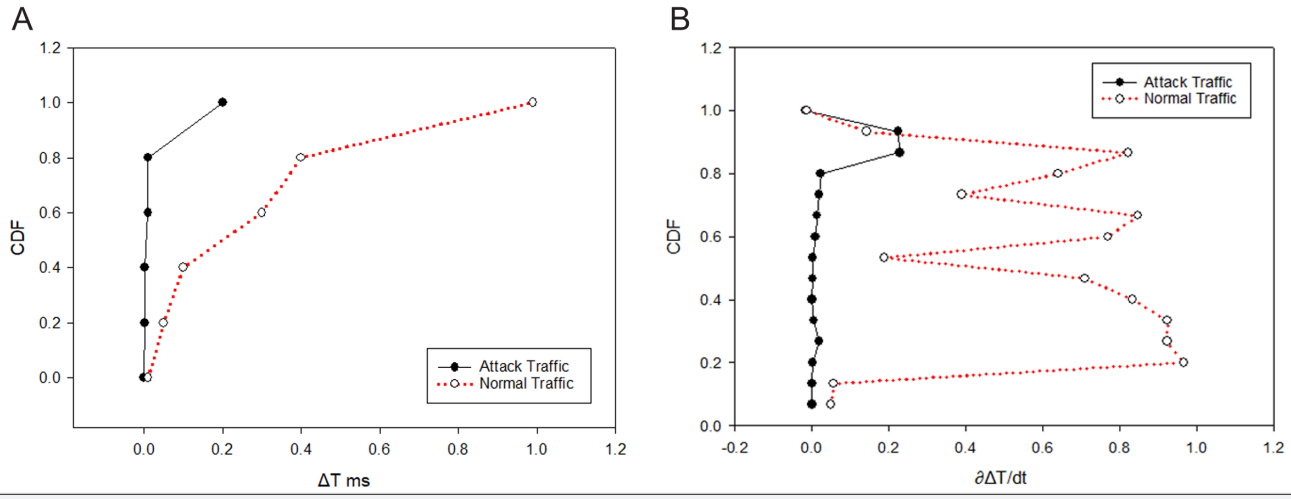

Fig. 15. A, B. Cumulative distribution function vs. inter-packet interval (A) attack traffic and (B) normal traffic.

TABLE VII. MEAN AND VARIANCE WITH RESPECT TO ENTROPY VARIATIONS IN DIFFERENT TIME FRAMES FOR ATTACK TRAFFIC

	Entropy Calculations						
Source IP-Address	**Time Frame I**	**Time Frame II**	**Time Frame III**	**Time Frame IV**	**Time Frame V**	**Mean**	**Variance**
192.168.1.104	0.97	2.078	2.031	1.89	1.21	1.6358	0.260234
192.120.225.114	0.829	0.930	2.432	0.111	0.201	0.9006	0.866021
104.18.21.226	1.625	1.654	1.88	1.765	1.454	1.6756	0.025476
49.44.138.104	1.351	1.4207	1.507	1.660	1.002	1.38814	0.059895
34.107.221.82	1.351	2.479	1.432	1.234	0.998	1.4988	0.327005
172.217.167.35	1.351	1.838	1.99	1.897	1.543	1.7238	0.071393
192.0.73.21	1.625	1.943	1.67	1.406	1.444	1.6176	0.045899
18.205.100.99	1.625	1.789	1.709	1.777	1.201	1.6202	0.059179

TABLE VIII. MEAN AND VARIANCE WITH RESPECT TO ENTROPY VARIATIONS IN DIFFERENT TIME FRAMES FOR NORMAL TRAFFIC

	Entropy Calculations						
Source IP-Address	**Time Frame I**	**Time Frame II**	**Time Frame III**	**Time Frame IV**	**Time Frame V**	**Mean**	**Variance**
192.168.1.104	0.8010	0.801	0.788	0.796	0.843	0.8058	0.000461
192.120.225.114	0.944	0.823	0.852	1.065	1.146	0.966	0.019032
104.18.21.226	1.16	1.2	1.216	1.065	1.146	1.1574	0.003481
49.44.138.104	1.420	1.2	1.477	1.31	1.4	1.3614	0.011745
34.107.221.82	1.420	1.46	1.477	1.31	1.146	1.3626	0.018888
104.22.0.175	1.16	1.2	1.216	1.31	1.4	1.2572	0.009399
104.22.1.1	1.16	1.46	1.477	1.065	0.8	1.1924	0.080966

Figure 17 gives us the details of CPU utilization of our virtual machine (with VMware workstation as our hypervisor) set up with 4 Gb allotted RAM. We used a simple system setup with an ubuntu operating system and PyCharm installed as our IDE. During an attack, the average CPU utilization was 34.46% (range 30–45%), which is a considerable low resource utilization for a detection setup. When the incoming packets reached a rate of 7–12 Mbits/s (during an attack), the CPU utilization reached a high of 65–85%. Following the detection and entropy calculation, the CPU utilization remained the same and got lower to 22.4% after 1 minute. The outgoing traffic remained almost constant during

TABLE IX. TRAFFIC PATTERN VARIATIONS IN NORMAL AND ATTACK TRAFFIC

Parameters	Normal (Benign Traffic)	Attack Traffic
Average packet size	0–600 bytes	0–1800 bytes
Average inter-packet interval	0–0.2 ms	0–1 ms
Average mean (entropy)	1.16	1.507
Average variance (entropy)	0.019	0.21438

the entire session ranging from 0.02 to 1.75 Mbits/s. The minimum CPU utilization during the attack was 45% which lowered to 17.9% after the DDoS attack was stopped. Our base system configuration (Windows 10 Operating/Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz 1.80 GHz 16 Gb) had CPU utilization of 12% with no attack to a high of 77% with an attack. The memory use was 79% (8.5 MB). The average CPU utilization remained 28.6% which is significantly less.

Moreover, our system has very less data memory requirement with our python code and PCAP file taking 8 Kb and 560 Kb of space,

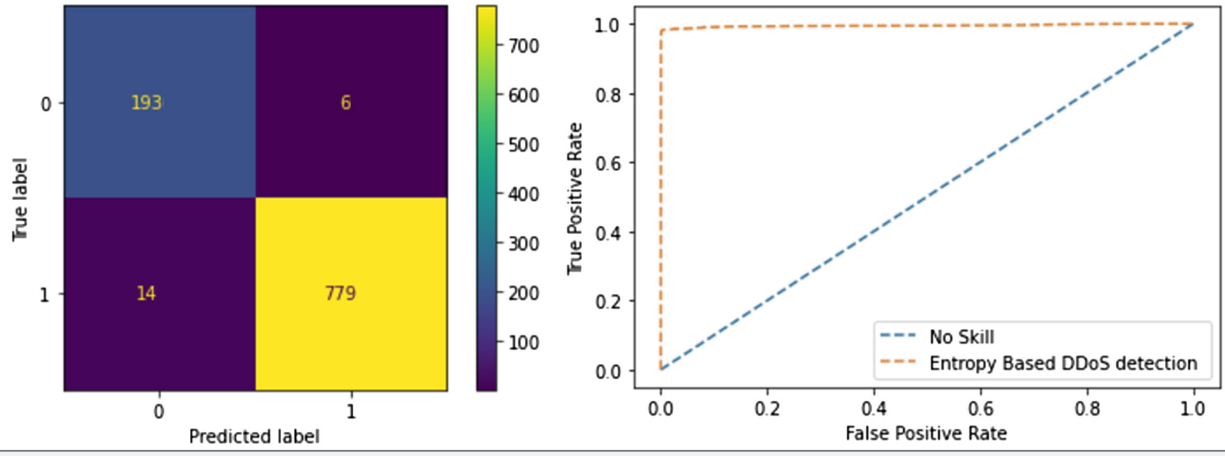

Fig. 16. Confusion matrix and AUC curve.

TABLE X. PERFORMANCE METRIC SCORES

Accuracy (%)	FPR (%)	Sensitivity (%)	Specificity (%)	Precision	Recall	*F1*
97.9	0.03	0.9823455	0.96984924	0.992356	0.0075662	0.9873257

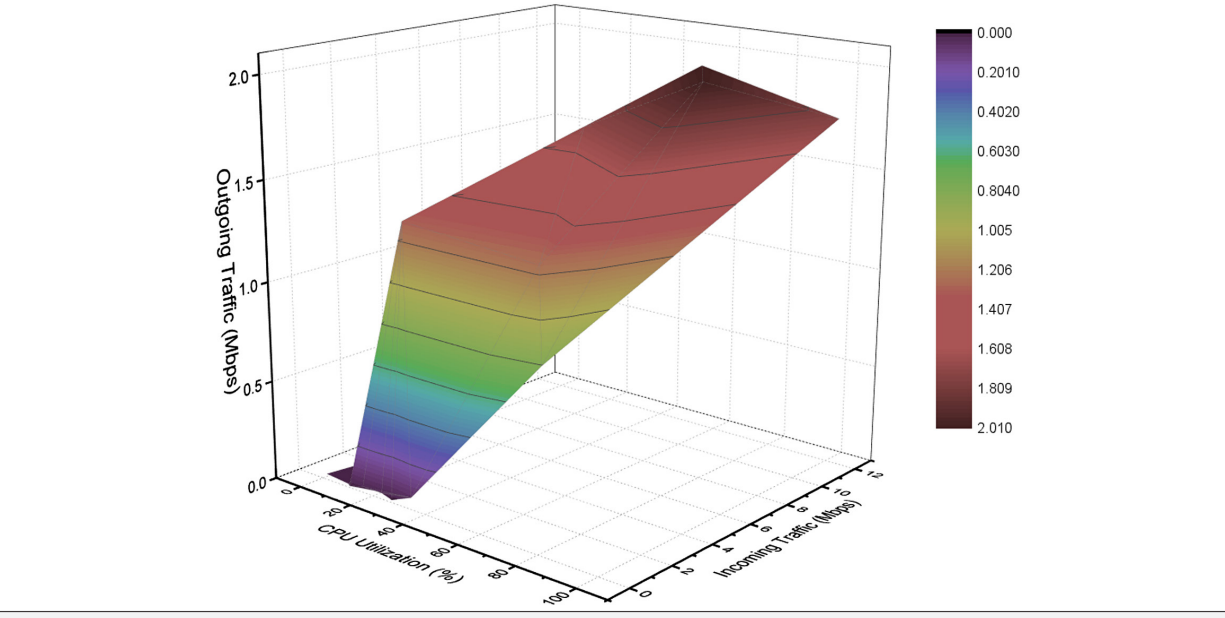

Fig. 17. CPU utilization of our DDoS detector virtual machine (Ubuntu) with incoming and outgoing traffic progressing from No Traffic to Attack Traffic.

TABLE XI. COMPARATIVE ANALYSIS OF VARIOUS ENTROPY BASED DETECTION SETUPS

Method	Detection Accuracy (%)	FPR	FNR	Average CPU Utilization (%)	RAM MB	Average Response Time (Seconds)
1. Entropy-based anti DDoS model in SDN [46]	93–94	0.06	0.06	40-50	NA	NA
2. S-DPS SDN based DDoS protection for smart grids [38]	100	0	0	55	205	25
3. Entropy-based application layer DdoS detection using ANN [39]	98	NA	NA	50-88	NA	20
4. Smart detection [40]	93-96	0.2	0.04	60-80	NA	NA
5. Proposed method	**97.9**	**0.03**	**0.001**	**28.6**	**8.5**	**0.1-3**

FPR, false positive rate; FNR, false negative rate; RAM, random access memory.

respectively. The response of our detector to the attack is fast. Our proposed method can detect the attack within 0.1–3.0 seconds even if the number of packets is comparatively low.

A. Comparison with Existing Methods

We use specific parameters to compare our DDoS detection method with existing methods and techniques based on entropy variations. The noteworthy being the *accuracy, FPR, FNR, average CPU-RAM utilization, and response time*. These parameters are paramount in predicting the novelty and efficiency of an approach.

Comparing with the recent existing techniques (Table XI), one can infer that our proposed method of DDoS detection shows high accuracy with low FPR and FNR. It is also noteworthy that our proposed detector has a fast response time and CPU utilization during the whole process remained low.

VIII. CONCLUSION

Various security challenges are seen in different legacy and traditional networks with different IDSs working to detect the DDoS traffic. High computational complexity, high traffic overhead, low accuracy, and high FPR add to the issues already faced by users due to the attack. We used a simple setup and a cross-platform python scripting tool to make it lightweight and robust. The email alert feature is added to notify the attack detailing the attack sent in the email attachments. We used parallel programming to set the connections, provide real-time traffic monitoring, and detect the attack based on entropy variations simultaneously to reduce the traffic-overhead and latency.

We used *accuracy, FPR, sensitivity, specificity, precision, recall, F1 score, CPU-RAM utilization, and response time* as our detection performance metrics. Our implementation uses 1.2% CPU initially with 29.8 MB memory usage. After the attack detection and alerting, the average CPU utilization remained 28.6% which is significantly less. Calculating the FPR, FNR, accuracy, and average response time, we have FPR and FNR as 0.03 and 0.001, respectively, and an accuracy of 97.9% with an average response time of 0.1–3 seconds. The proposed detection method has improved performance and produces efficient results with low resource utilization compared to the other existing techniques.

As for future work, we can incorporate the DDoS mitigation framework into it to have both detection and mitigation in a single setup.

Peer-review: Externally peer-reviewed.

Author Contributions: Concept – B.H.; Design – B.H.; Supervision – F.K.; Materials – B.H.; Data Collection and /or Processing – B.H.; Analysis and/or Interpretation – B.H., F.K.; Literature Review – B.H., F.K.; Writing – B.H., F.K.; Critical Review – B.H., F.K.

Declaration of Interests: The authors have no conflicts of interest to declare.

Funding: The authors declared that this study has received no financial support.

REFERENCES

1. 2021 Akamai Report. Available: https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp.
2. Available: https://securelist.com/ddos-attacks-in-q3-2020/99171/.
3. Available: https://www.businesswire.com/news/home/20200929005235/en/NETSCOUT-Threat-Intelligence-Report-Shows-Dramatic-Increase-in-Multivector-DDoS-Attacks-in-First-Half-2020.
4. Available: https://www.netscout.com/blog/what-watch-2021.
5. Y. Mirsky, and M. Guri, "DDoS attacks on 9-1-1 emergency services," *IEEE Trans. Depend. Sec. Comput.*, 1–1, 2020. [CrossRef]
6. B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep learning-based DDoS-attack detection for cyber–physical system over 5G network," *IEEE Trans. Ind. Inform.*, vol. 17, no. 2, pp. 860–870, 2020. [CrossRef]
7. Madiha H. Syed, E. B. Fernandez, and J. Moreno, "A misuse Pattern for DDoS in the IoT," *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, 2018.
8. N. Umamaheswari, and R. Renugadevi, "A subset feature selection based DDoS detection using cascade correlation optimal neural network for improving network resources in virtualized cloud environment," *IOP Conf. S. Mater. Sci. Eng.* vol. 993, no. 1, 2020. [CrossRef]
9. J. Cheng, C. Zhang, X. Tang, V. S. Sheng, Z. Dong, and J. Li, "Adaptive DDoS attack detection method based on multiple-kernel learning," *Sec. Commun. Netw.*, vol. 2018, 1–19, 2018. [CrossRef]
10. R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," 2018, *IEEE Security and Privacy Workshops (SPW)*. IEEE Publications.
11. S. Behal, K. Kumar, and M. Sachdeva, "D-FACE: An anomaly based distributed approach for early detection of DDoS attacks and flash events," *J. Netw. Comput. Appl.*, vol. 111, pp. 49–63, 2018. [CrossRef]
12. R. Vishwakarma, and A. K. Jain, "A honeypot with machine learning based detection framework for defending IoT based botnet DDoS attacks." 3rd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE Publications, 2019.
13. B. Kriti, and B. B. Gupta. "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment." *J. Amb. Intell. Human. Comp*. Vol. 10, no. 5, 2019, pp. 1985–1997.
14. C. Li et al., "Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, p. e3497, 2018. [CrossRef]

15. M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," *Appl. Intell.*, vol. 48, no. 10, pp. 3193–3208, 2018. [CrossRef]
16. N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and DDoS in SDN," *Sec. Commun. Netw.*, vol. 9, no. 18, pp. 6386–6411, 2016. [CrossRef]
17. N. Ravi, and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3559–3570, 2020. [CrossRef]
18. K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards IoT-DDoS prevention using edge computing," *{USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.
19. M. P. Singh, and A. Bhandari, "New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges" *Comput. Commun.*, vol. 154, pp. 509–527, 2020. [CrossRef]
20. L. Tan, K. Huang, G. Peng, and G. Chen, "Stability of TCP/AQM networks under DDoS attacks with design," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, 3042–3056, 2020. [CrossRef]
21. G. Somani, M. S. Gaur, D. Sanghi, M. Conti, M. Rajarajan, and R. Buyya, "Combating DDoS attacks in the cloud: Requirements, trends, and future directions," *IEEE Cloud Comput.*, vol. 4, no. 1, pp. 22–32, 2017. [CrossRef]
22. V. Deepa, K. Muthamil Sudar, and P. Deepalakshmi, "Design of ensemble learning methods for DDoS detection in SDN environment." International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), IEEE Publications, 2019.
23. K. Li et al., "Effective DDoS attacks detection using generalized entropy metric." International Conference on Algorithms and Architectures for Parallel Processing, Berlin, Heidelberg, Springer, 2009.
24. M. Roopak, G. Y. Tian, and J. Chambers, "Multi-objective-based feature selection for DDoS attack detection in IoT networks," *IET Netw.*, vol. 9, no. 3, pp. 120–127, 2020. [CrossRef]
25. J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Sec. Commun. Netw.*, vol. 2018, 1–8, 2018. [CrossRef]
26. A. Fadil, I. Riadi, and Sukma Aji, "A novel ddos attack detection based on gaussian naive bayes," *Bulletin EEI*, vol. 6, no. 2, pp. 140–148, 2017. [CrossRef]
27. S. N. Shiaeles, V. Katos, A. S. Karakos, and B. K. Papadopoulos, "Real time DDoS detection using fuzzy estimators," *Comput. Sec.*, vol. 31, no. 6, pp. 782–790, 2012. [CrossRef]
28. K. J. Singh, and T. De, "DDOS attack detection and mitigation technique based on Http count and verification using CAPTCHA." International Conference on Computational Intelligence and Networks, IEEE Publications, 2015.
29. T. F. Ghanem, W. S. Elkilani, and H. M. Abdul-Kader, "A hybrid approach for efficient anomaly detection using metaheuristic methods," *J. Adv. Res.*, vol. 6, no. 4, pp. 609–619, 2015. [CrossRef]
30. S. Hameed, and U. Ali, "Efficacy of live DDoS detection with Hadoop," NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium, IEEE Publications, 2016.
31. H. Nazrul, H. Kashyap and D. K. Bhattacharyya. "Real-time DDoS attack detection using FPGA," *Computer Communications*, vol. 110, pp. 1–10, 2017.
32. Y. Gu, Y. Wang, Z. Yang, F. Xiong, and Y. Gao, "Multiple-features-based semisupervised clustering DDoS detection method," *Math. Probl. Eng.*, vol. 2017, 1–10, 2017. [CrossRef]
33. J. Cui, M. Wang, Y. Luo, and H. Zhong, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," *Future Gener. Comput. Syst.*, vol. 97, pp. 275–283, 2019. [CrossRef]
34. D. Erhan, and E. Anarim, "Hybrid DDoS detection framework using matching pursuit algorithm," *IEEE Access*, vol. 8, pp. 118912–118923, 2020. [CrossRef]
35. R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future Gener. Comput. Syst.*, vol. 111, pp. 763–779, 2020. [CrossRef]
36. Z. Liu, X. Yin, and Y. Hu, "CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-Learning," *IEEE Access*, vol. 8, pp. 42120–42130, 2020. [CrossRef]
37. R. M. A. Ujjan et al., "Entropy based features distribution for anti-DDoS model in SDN," *Sustainability*, vol. 13, no. 1522." (2021), 2021. [CrossRef]
38. H. Mahmood, D. Mahmood, Q. Shaheen, R. Akhtar, and W. Changda, "S-DPS: An SDN-based DDoS protection system for Smart grids," *Sec. Commun. Netw.*, vol. 2021, 1–19, 2021. [CrossRef]
39. K. Johnson Singh, K. Thongam, and T. De, "Entropy-based application layer DDoS attack detection using artificial neural networks," *Entropy*, vol. 18, no. 10, p. 350, 2016. [CrossRef]
40. F. A. F. Silveira et al., "Smart detection-IoT: A DDoS sensor system for Internet of Things." International Conference on Systems, Signals and Image Processing (IWSSIP), IEEE Publications, 2020.
41. S. Behal, and K. Kumar, "Detection of DDoS attacks and flash events using novel information theory metrics," *Comput. Netw.*, vol. 116, pp. 96–110, 2017. [CrossRef]
42. M. Idhammad, K. Afdel, and M. Belouch, "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest," *Sec. Commun. Netw.*, vol. 2018, 1–13, 2018. [CrossRef]
43. M. Nooribakhsh, and M. Mollamotalebi, "A review on statistical approaches for anomaly detection in DDoS attacks," *Inf. Sec. J. Glob. Perspect.*, vol. 29, no. 3, pp. 118–133, 2020. [CrossRef]
44. A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM Sigcomm Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, 2005. [CrossRef]
45. J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of dos and ddos attacks in iot-based stateful sdn: An experimental approach," *Sensors (Basel)*, vol. 20, no. 3, p. 816, 2020. [CrossRef]
46. N. M. AbdelAzim, S. F. Fahmy, M. A. Sobh, and A. M. Bahaa Eldin, "A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism," *Egypt. Inform. J.*, vol. 22, no. 1, 85–90, 2021. [CrossRef]
47. S. Behal, and K. Kumar, "Detection of DDoS attacks and flash events using information theory metrics–an empirical investigation," *Comput. Commun.*, vol. 103, pp. 18–28, 2017. [CrossRef]
48. A. Mishra, N. Gupta, and B. B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," *Telecommun. Syst.*, pp. 1–2, 2021.
49. N. Agrawal, and S. Tapaswi, "An SDN-assisted defense Mechduanism for the shrew DDoS attack in a cloud computing environment," *J. Netw. Syst. Manag.*, vol. 29, no. 2, pp. 1–28, 2021.
50. S. Sambangi, and L. Gondi, "A machine learning approach for DDoS (distributed denial of service) attack detection using multiple linear regression," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 63, no. 1, 2020.

Beenish Habib is a PhD candidate in the Department of Electronics and Communication Engineering in NIT Srinagar. Her field of expertise is Cloud and Network Security and is mainly working on detecting and mitigating DDoS attacks. She has done her Bachelors in Technology in ECE from IUST Awantipora and Masters in Technology in Communication and IT from NIT Srinagar. She has 3 years of teaching experience and 4 years of research experience.

Dr. Farida Khurshid is currently providing services as Associate Professor in ECE department in NIT Srinagar. She has authored and co-authored multiple peer-reviewed scientific papers and presented works at many national and international conferences. Her research interests include Digital Image Processing, Security and Biometrics.

APPENDIX A

Algorithm *A*: C&C Server

1: procedure *Set Connection Parameters (Addresses of Hosts, Ports, Threading and Queue)

2: //Create Socket ***S***

3: Set it for TCP/IPv4 protocol

4: Connect it to the IP and Port (Listen)

If error Print (error)

If Previous Connections Present {Delete addresses}

5:// main loop. Stops when Exit Command Executed

6: while (no keyboard interrupt)

7: // set Connection ***C*** (Heartbeat: "hi" and "hello" encoded and decoded in binary and human readable format utf-8)

8: Print (Connection has been established with the IP and Port)

9 Set value (Commands ***cmd***)

10. while

if ***cmd*** = input ('Shell >') //list all the connections

select >1 to select bot 1, >2 for bot 2 and for any exception print None

if ***cmd***=quit: break the connection and return back to Shell (9)

if ***cmd*** = exit: exit connection and close the socket {delete all connections and addresses}

if ***cmd (ln >0)*** {character string}

encode (character string-Bytes)

decode (Bytes-character string)

print (character string)

11. end while

12: //

13: Set Threads ***t*** (Server Multiple Tread Connections, ***Jobs j {1,2}***)

14: 1 a. Set Connections {***j==1***}

b. Accept Connections

2. Command Execution ***{j==2}***

15: Return //main

Algorithm B: DDoS Attack

1: procedure *Import Scapy & random

2: //Set ***target_ip***

3: Create random IP {***ip***=string in range {1,254} and ***i*** in range {4}}

4: Join (***ip***)

5: ***ip_list*** {random IPs and in range i= {specified number}}

6: Print (***ip_list***)

7: //while (True)

8: ip***1*** = IP with source and destination IP

tcp***1*** = TCP with source and destination port (80)

9: Generate Packets {***packet = ip1 / tcp1*** with interval .10 seconds}

10: Increment (***i***) 11: end //

Algorithm C: Real Time DDoS Attack Detection and Email Alerting

1: Procedure * Import Scapy, Counter, Plotly, Web-browser, Pandas, smtplib, datetime, ssl, email, psutil, multiprocessing, time, email.mime and encoder-64.

2: while(true)

3: call pkt_chk(0) //check individual packets

4: check pkt_chk(i)

5: If {pkt_header flags Sij ,Ack}

6: calculate pkt_number Np

7: if Sij = 'SYN' || Ack = '0'

8: call pkt_number Np(i)

9: if Np(i) > ϑj { where ϑj = 25 }

10: print ("DDoS in Progress")

11: else {print ("Looks Good !!")}

12: //end if

13: Store Input = data ("DDoS.pcap")

14: start parallel processes

15: //Plot Offline Network Graphs for IP list and individual Packet Count (bytes received and bytes sent).

16: //Visualize Real -Time Incoming/Outgoing Traffic (Traffic-Visualizer).

17: //If alert ("DDoS Attack in Progress")

18: // send email {server login to a registered email-id)

19: //email-alert = '''Dear Admin,

Your IDS has detected DDoS Attack.

Please check attachment

DDoS Tool

''' //

20: continue (Traffic-Visualizer)

Algorithm D: Entropy Calculation

1: Procedure *Import maths and NumPy

2: //Set ***entropy and counter =0***

3: for loop to iterate through every IP in the list

4: item x in range(len(data))

5: counter +=1

6. Probability = float(counter)/len(data)

7: Entropy+=-prob*maths.log{prob,2}

8: print ("Entropy: {}".format(entropy))

9: end //

Algorithm E: Entropy based DDoS Detection

1: Procedure *Input: Traffic data , ϑn^{T} , ϑn^{0}

2. Output: Entropy value

3: //Set Time_Window {TW ← 3 min,5 min, 7 min}

4: for loop to iterate through every IP in the list

5: Probability = float(counter)/len(data)

6: Entropy+=-prob*maths.log{prob,2}

7: Calculate average entropy ϑn^{A_1}

8: $\vartheta n^{A} \leftarrow \vartheta n^{A_1} / \log \eta$ (Normalization)

9: if $\vartheta n^{A} < \vartheta n^{T} \,||\, \vartheta n^{A} < \vartheta n^{0}$

Set traffic ← Attack

Else ← Benign /Normal Traffic

10: end //