Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2022.023771
Article

Tech Science Press

Early DDoS Detection and Prevention with Traced-Back Blocking in SDN Environment

Sriramulu Bojjagani[1], D. R. Denslin Brabin[2,*] and K. Saravanan[2]

[1]Department of Computer Science and Engineering, School of Engineering and Applied Sciences (SEAS), SRM University-AP, Amaravati, Andhra Pradesh, 522503, India
[2]Department of Computer Science and Engineering, Madanapalle Institute of Technology & Science, Chittoor, Andhra Pradesh, 517325, India
*Corresponding Author: D. R. Denslin Brabin. Email: denscse@gmail.com
Received: 21 September 2021; Accepted: 12 January 2022

Abstract: The flow of information is a valuable asset for every company and its consumers, and Distributed Denial-of-Service (DDoS) assaults pose a substantial danger to this flow. If we do not secure security, hackers may steal information flowing across a network, posing a danger to a business and society. As a result, the most effective ways are necessary to deal with the dangers. A DDoS attack is a well-known network infrastructure assault that prevents servers from servicing genuine customers. It is necessary to identify and block a DDoS assault before it reaches the server in order to avoid being refused services. This prompted us to develop a unique way for detecting and preventing DDoS attacks at the router level in a Software-Defined Network (SDN) environment. This study demonstrates how the method efficiently integrates the first and second signatures in SDN infrastructure domains to identify and prevent DDoS attacks. It also proposes an Early DDoS Detection and Prevention (EDDDeP)-based approach for detecting and blocking malicious traffic in an SDN context. This article covers the EDDDeP, which assists in identifying and preventing DDoS in SDN to prevent malicious traffic from reaching its intended target. As a consequence, the DDoS assault is ultimately contained inside the environment, eliminating superfluous traffic in the DDoS network architecture. This method offers a unique technique to detect a DDoS assault and notify nearby neighbours in order to avert server damage.

Keywords: Software defined network; security; DDoS; network; legitimate clients

1 Introduction

The DDoS attack disables the service and prevents the server from providing it. Denial-of-Service (DoS) attacks primarily the network's bandwidth or connectivity. Bandwidth influences network traffic, rendering all resources unavailable, preventing legitimate users' requests from reaching the server. The server's operating system is impacted by a connectivity attack, which consists of many connection requests. DoS attacks are common because attacking tools that are easy to attack and difficult to prevent are widely available on the internet. DoS attacks can be generated in two ways: DoS attacks and DDoS. DoS attacks are carried out by a single machine, whereas DDoS attacks are carried out by a network of cooperative devices that all target the same victim.

Data integrity, confidentiality, and availability are the three main terms used to describe network security [1–3]. Among these security features, availability is the most important for mitigating DoS and DDoS attacks. Many works have been proposed to overcome these attacks in SDN [4,5]. Fig. 1 depicts a typical SDN architecture. From bottom to top, the SDN-infrastructure layers are made up of network devices such as routers and switches. The packets received from the control layer are forwarded by these devices. The SDN-Control layer is the middle layer, and it provides network services to the application layer via the northbound interface. The topmost layer is the SDN-Application layer, which communicates with SDN-enabled switches to receive packets. According to a literature survey, approximately 15.30 percent of U.S.A. business organizations are subjected to DDoS attacks despite the implementation of network security measures such as firewalls and intrusion detection systems. There are two types of solutions for DDoS attacks: application-based and protocol-based. Application-based primarily focuses on all system applications such as databases, email, and so on. Protocol-based infrastructure significantly improves existing infrastructure and protects against DDoS attacks.

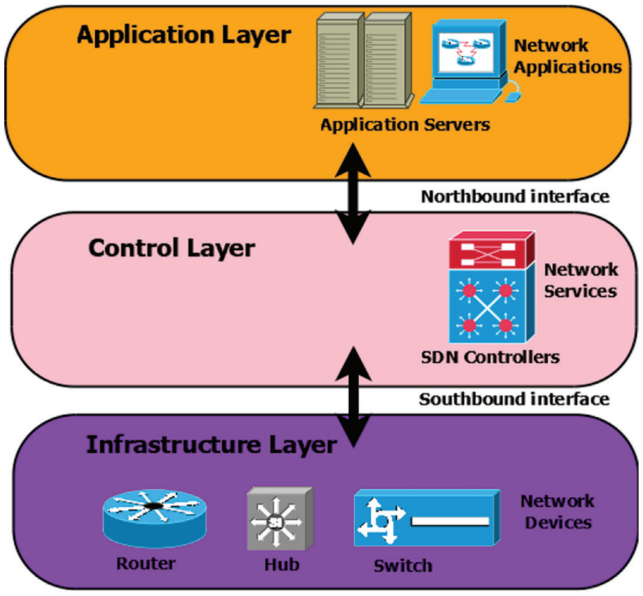


Figure 1: SDN architecture

Software-Defined Northbound Application Program Interfaces (SDN northbound APIs) are commonly named as SDN RESTful Application Programming Interface (API) [6]. These are used to communicate between the SDN controller and the network-based applications. SDN northbound APIs enable efficient network orchestration and automation to align the requirements of various applications via SDN network programmes.

SDN APIs serve as a direct interface between the SDN controller and the SDN applications. These applications provide information to the network such as bandwidth, storage, and data. The network calculates the number of resources needed based on these parameters. Load balancers, firewalls, all software-defined security services, and orchestration applications across cloud resources are examples of network applications that can be optimized via the northbound interface. Northbound APIs are also used to connect the SDN Controller to automation stacks like Puppet, Chef, SaltStack, Ansible, and CFEngine, as well as orchestration platforms like OpenStack, Virtual Machine (VM), vCloudDirector, and Apache's open source CloudStack. The goal is to abstract the network's inner workings so that application developers can "hook" into the network and make changes to meet the needs of the application.

Existing works consider DDoS to be extremely problematic, and as a result, several other works have been proposed to address the issues in SDN to detect DDoS attacks. However, these approaches do not address the issues, so a new mechanism to detect and prevent DDoS attacks in an SDN environment is required. This requirement prompted us to propose EDDDeP [7], a new model for early and effective detection and prevention of DDoS attacks. In this paper, we show that the EDDDeP employs both stateful and stateless signatures in tandem for attack detection, which differs from other related work that employs either of these two signature approaches.

Our proposed framework is a protocol-based approach on the SDN network infrastructure known as EDDDeP, a domain-based solution distributed at the router level through the affected domain to the victim. Our proposed solution provides a natural way to identify attack sources that does not rely on any trace-back algorithms or techniques.

The following are some of the contributions made by the EDDDeP described in this paper.

1. There is no need for additional hardware or computational load to detect DDoS. Only a small proportion of packets are examined in comparison to the total throughput of network devices.
2. It only necessitates a minor increase in storage load for detection.
3. Finally, reports of attacks may refer to multiple packets rather than the "one packet, one alert" techniques used by traditional countermeasures to further reduce the protected network workload.
4. The efficient use of storage and computational resources enables the EDDDeP to be directed to their sources rather than their intended target.

2 Backgrounds and Related Work

DDoS attacks, which primarily deny the services of legitimate users, are one of the network's most serious threats. Many attacks attempt to steal the flow of information over the network, whereas DDoS attacks primarily focus on flooding traffic rather than stealing. Exhausting protocols, such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), allow the attacker to stymie legitimate services by generating excessive internet traffic. Synchronous (SYN) flood, for example, is a well-known DDoS attack that employs the TCP protocol. To establish a connection between the clients and the server, TCP requires a three-way handshake.

1. The client sends the server a SYN bit sequence number and waits for a response.
2. The server confirms to the client by sending SYN +ACK(Acknowledgement) sequence number, where ACK equals SYN +1.
3. The client sends a connection message to the server.

The SYN flood principle is to send a large amount of traffic in order to deplete the available resources on the internet during the TCP connection. When the server's resources are depleted, it cannot receive the confirmation message sent after the SYN + ACK response. As a result, the third handshake cannot be performed. As a result, the server tries again to send SYN + ACK packets to the client, and if the resources are depleted, SYN + ACK packets are not delivered. The server waits for a certain amount of time before terminating the connection.

Many approaches have been proposed to defend against DDoS attacks in both traditional and software-defined networks. This paper addresses security attacks by employing lightweight and protocol-independent mechanisms, resulting in an efficient and defence framework in SDN. Prakash et al. [8] propose Medium Access Control (MAC) filtration and cryptography-based authentication when a client's request must be authenticated efficiently. Subbulakshmi [9] created a framework for an Artificial Immune System (AIS) algorithm that uses threshold-based techniques to mitigate DoS attacks in the cloud. Their method can be used on cloud infrastructure to prevent unauthorised users from wasting resources on bogus requests. Zargar et al. [10] described an intrusion detection and prevention system for dealing with DDoS attacks. The survey primarily focuses on the challenges of DDoS attacks [11] and discussions on DDoS flooding attack prevention and countermeasures. Mopari et al. [12] proposed an IP spoofing mechanism that detects and discards malicious traffic using an IP-hop count table.

Thing et al. [13] create a non-intrusive IP traceback scheme that keeps valid source addresses while traversing the network. The network traffic flow is identified by determining which routers were used as unauthorized source addresses to construct the attack graph. Madhan et al. [14] implemented a framework in which the number of bandwidths allocated to the user is determined by the legal user's probability. Sterne et al. [15] made use of existing statistical monitoring, which tracks increases in traffic to a specific system. Badotra et al. [16] used a tool, the SNORT intrusion detection system, to implement a framework for early detection of DDoS. The tool aids SDN controllers in the early detection of DDoS attacks. The authors used five different network scenarios to detect the DDoS. Several machine learning classifiers were used to detect and mitigate DDoS attacks. They were working with the entropy-based, three collectors, and classification sections in one of these approaches [17]. Every time data is collected for new threats and the attack detection time is provided.

This paper proposes an EDDDeP, a domain-based mechanism that detects and prevents malicious traffic earlier. Malicious traffic is unable to reach the server. Our mechanism makes use of the benefits of both the statistical and the congestion algorithms. The proposed model has the advantage of reducing the computational load on the network because we have analyzed more packets from unknown hosts. If a domain is affected, it notifies the rest of the network. As a result, malicious traffic can be detected and avoided. Even if malicious traffic flows through the SDN, it cannot reach other layers.

3 Distributed DoS Attacks in Software Defined Network

The adversary sends an undetermined number of packets across the network to harm legitimate servers and clients. The two types of requests that result in DoS attacks from the adversary are general requests and nominal clients. DDoS attackers take control of the various hijacked devices and use this request to send massive packets to the victim.

1. The adversary launched DDoS attack with First Request: To launch a DDoS request, the attacker first maintains and controls numerous devices that will transmit DDoS packets to the target. These devices are often known as botnets or zombie computers. The first phase is DDoS detection, which tries to assess the vulnerability in connected devices such as WiFi routers, PCs, mobile phones, and closed-circuit video cameras. In this phase, the attacker may use any scanning method, such as local subnet scanning, permutation scanning, random scanning, hitlist scanning, and so on, to identify susceptible devices before launching a DDoS assault. The attacker may carry out a variety of attack schemes with high pertinency to hijack susceptible devices and get successful scanning results. The attacker may now identify the botnets in the network, transmit the appropriate amount of control messages to gain command of the devices, and send malicious requests to the target to conduct a DDoS assault.

2. The adversary launched DDoS attack with Second Request: A bogus request is the second kind of request. The attacker may fabricate or duplicate valid user requests and deliver them to the victims. On the other side, the victim receives many fraudulent requests, causing the server to go down, network bandwidth to be drained, or computational resources to be spent. DDoS assaults on three tiers of SDN infrastructure are shown in Tab. 1.

Table 1: The DDoS attack in various SDN layers

Type of DDoS attack in SDN	Victims	Example
DDoS targeted at SDN infrastructure layer	Network devices Ex: Routers, network Switches etc.	The DDoS attacker tries to send massive, bogus packets to the SDN-enabled switch to target the victim.
DDoS targeted at SDN control layer	SDN controller Ex: Databases	The DDoS attacker sends a new packet generated by the malicious traffic to depreciate resources and consume the controllers' bandwidth.
DDoS targeted at SDN application layer	Network Applications	The DDOS adversaries try to send malicious packets to all the victims who actively participated in the SDN application layer.

The attacker can successfully launch a DDoS attack in SDN infrastructure layer through following steps. The sequence of these steps is shown in Fig. 2.

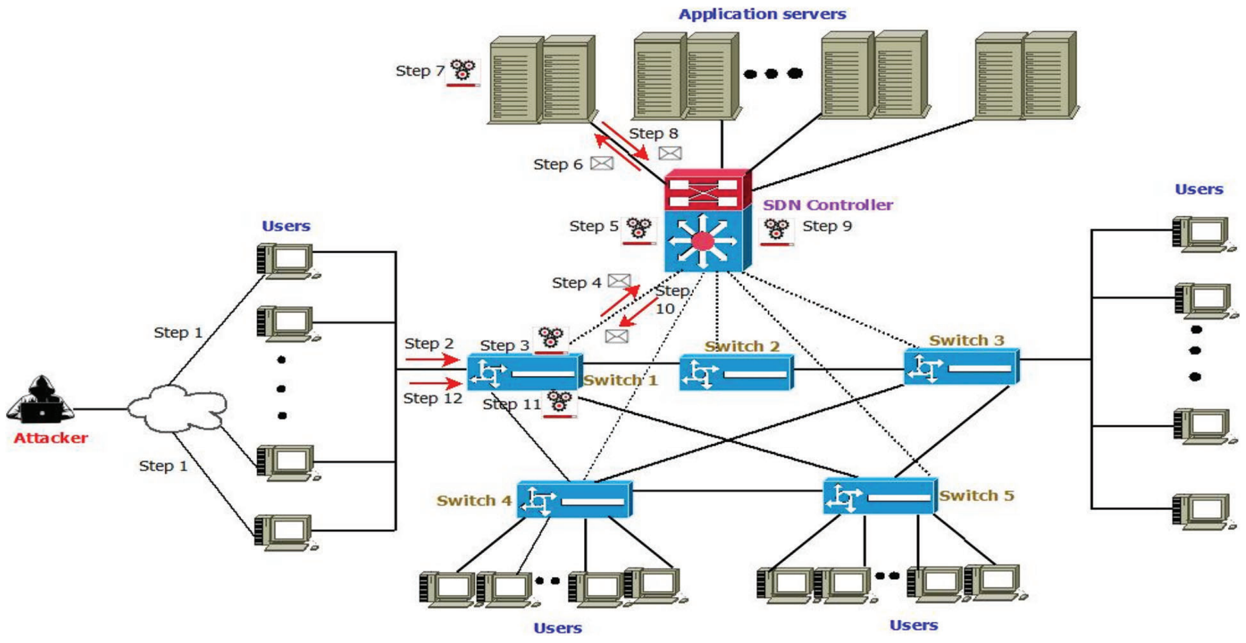


Figure 2: Adversary target DDoS attack in SDN

Step 1: The attacker can send packets continuously to a specified victim with varying IP address. Sometimes the source address of the IP is invalid. The victim can be a switch, a router or anyother part of the network devices in the SDN infrastructure layer [18].

Step 2: This set of malicious traffic aims at the switch

Step 3: The switch identifies the signatures of the packets using the table entries.

Step 4: Now, the switch cannot identify these packets as the signature of these malicious DDoS packets did not match. The unmatched packets have been sent to the controller for further verification.

Step 5: The controller also verifies the signature and header of the packets.

Step 6: After verification, the unmatched packets are forwarded directly to the corresponding application server.

Step 7, 8: The application server decides on these unmatched packets. The unmatched packets' signature and the header is different from the table flow entries.

Step 9, 10: The controller may add additional information to the packets using encapsulation and redirects the switch.

Step 11: The switch may add one or more entries according to the direction given by the controller.

Step 12: Now, the attacker may continuously send messages (anonymous packets) that can be matched with the newly added flow entries at the switch and deny the resources.

4 Proposed Methodology

This section explains in detail how an EDDDeP-based system detects and prevents DDoS attacks. If malicious traffic reaches its intended destination, the resources are denied to normal users. The EDDDeP-based system tries to identify and block malicious communication from reaching its intended destination as early as possible. In an SDN context, our whole system has a collection of cooperating EDDDeP domains. As illustrated in Fig. 3, each domain is comprised of a Command Controller (CC) server and a PreFilter (PF) that monitors traffic. Within the SDN domains, only one PF is installed on a single router. The server's services are supplied in collaboration with adjacent domains, which get the attack report from PF. The PF accounts for attack detection by reporting first- and second-signature attacks to its domain server.

The first signature is requesting that a statistical approach be used to gather data over a period. The threshold value is determined by examining the data, which is then verified using a statistical approach. The second-signature algorithm compares network traffic to a collection of recognised signatures. We must ensure that all servers and PFs are operating in secure conditions.

Within its SDN infrastructure domain, a server is linked to number of routers and other network devices. The domain ingress-router manages PF, ensuring that the burden is not distributed over several PFs on the server. An ingress-edge router primarily routes incoming packets to an unused IP address. They are the initial router in the domain via which packets are routed to enter a certain domain. The PF at the domain's ingress router is in charge of detecting traffic reported at the server and blocking it when packets enter the domain from neighbouring ones. As a result, just one PF in the router is used for the attack, and the other routers are not used for the same assault.

Each domain in the SDN layers goes through a two-stage detection and response procedure. The initial level of detection for a suspected assault is performed by PF. If PF discovers the first signature of a DDoS assault from the information in the second signature, it suggests that an attack may occur against a certain host. The PF performs first-signature detection, and its associated router employs the congestion algorithm. The identification of traffic spikes is performed by this congestion algorithm. When packets arrive at the router, they create a queue and are routed via the router on the network. Routers implement a threshold value by monitoring the amount of queue packets. When the threshold value is exceeded, the packets are discarded rather than added to the queue. The router's PF intercepts the packets and runs statistical sampling to assure the direction of the high traffic flow. If all the packets have the same destination, they are most likely all going to the same place, which might result in a DDoS assault. As a result, we can see that the traffic passes against the host without storing any information, which saves us storage space. After the first signature is completed, the sample packets are subjected to the second signature to guarantee the attack. If an attack is launched using a second signature, PF sends a warning to its server. This alert provides the traffic's destination address. The first step is the attack detection stage, which is completed in the manner previously indicated.

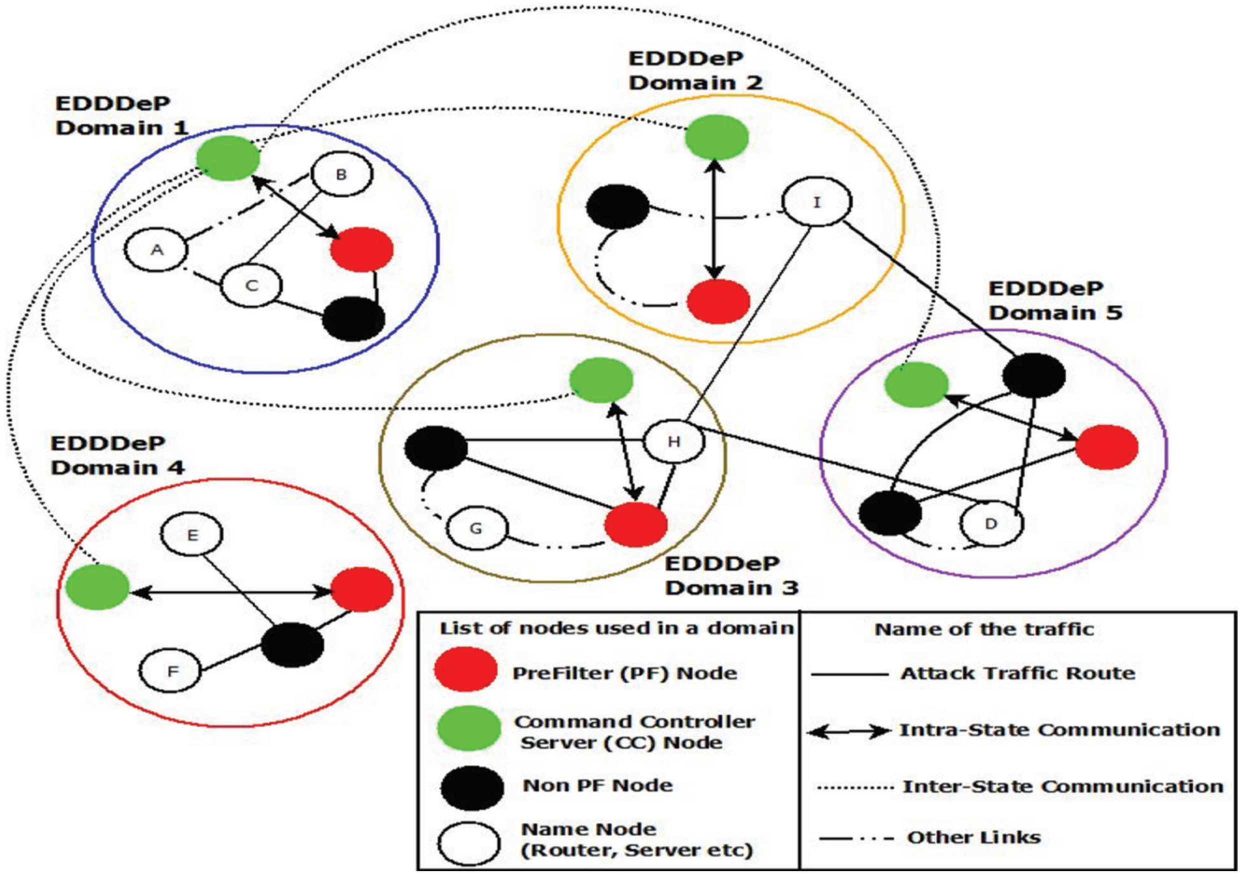


Figure 3: The EDDDeP domain operation in SDN environment

The second step begins with the server monitoring its EDDDeP domain. When the server gets a warning from PF about the assault, it validates that no comparable attack has been reported before. The server then sends a response directive to each PF on a router, instructing it to block incoming packets with a certain destination address. PF stops incoming packets until the packet arrival rate falls below the observed threshold for a certain period. This indicates that only a restricted number of packets may travel through the PF-associated router. PF does not block all traffic since a legitimate service might have large traffic, and totally blocking a legitimate service may result in a DDoS assault. To provide comprehensive security, the server sends a combined response to all domain servers in its vicinity. To produce such a request, the impacted server must first wait for a report from PF. Only when the rate of blocked packets exceeds the measured threshold does the PF send a report to the server. As previously stated, the procedure does not enable traffic to be propagated and is confined to its originating domain. Without adopting traceback methods, we may even trace back the attack source by a real-time reaction of the assault zone.

We have presented the EDDDeP-based system in the following architectural diagram to help you understand it better.

• Domain 1 nodes B and C execute a TCP-SYN flood, a DDoS assault on domain-4 node D as a victim.

• When traffic reaches domain 3, the PF node in this domain notices an increase in network traffic to one destination. The PF node in domain 3 begins analyzing the signature confirmation and discovers that a DDoS assault is causing traffic. The PF node notifies its domain server. The server investigates and validates that no identical response to the alleged assault has been delivered previously. The server then sends a signal to PF to prevent the incoming attack packets.

4.1 Overview of EDDDeP PreFilter

This section discusses the comprehensive design of PreFilter, one of the cores EDDDeP components. DDoS attacks need a tremendous amount of bandwidth to fulfil their goal of denying genuine user's access. PF has been created to warn and alert an assault so that a counterattack may be delivered. The assault must be detected by PF before the traffic reaches its destination.

The PF is a combination of three modules as shown in Fig. 4 that are explained below.

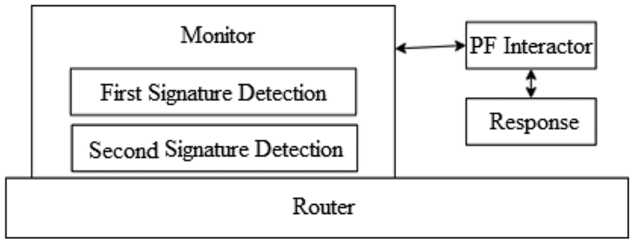


Figure 4: EDDDeP prefilter operation

• **PF Interactor:** It communicates with the domain's supervising server. It authenticates with its server and interacts with both the PF node and the server. The PF could not work without authentication, causing the associate router to discard genuine users' packets.

• **Monitor:** It is made up of two submodules: first-signature detection and second-signature detection. The congestion algorithm is used by the first signature to identify network traffic flowing through the router towards a certain domain. When an attack is discovered, the attack packet is sent to the second-signature detection submodule for additional examination. If the signatures match, the PF interactor module sends an alert to the supervising server with the attack details.

• **Response:** When the PF gets a response directive from the server, it creates a set of parameters with the response and sends it to the router, which then discards the attack packets based on the parameters. The programme then begins counting the number of packets deleted over time. Finally, the module gathers the rejected packets, and this is the amount of traffic entering the present domain from neighbouring domains. If this value is more than the threshold, the router must discard the assault traffic. The module sends data to the server using the PF Interactor.

When the threshold is inside the first-signature detection submodule, the router employs the congestion algorithm to delete packets. When the threshold becomes too high owing to network traffic, the packets in the queue are dropped. The submodule collects the rejected packets and sends them to the second-signature detection submodule. It determines if this traffic is destined for a certain host. In this situation, just drop packets are examined rather than all packets in the network. This minimises the system's computing effort. If all the drop packets are destined for the same destination, the packets are sent to the second-signature detection module for additional analysis.

The first signature module sends a sample of packets to the second signature module. The number of packets and the sample percent ratio are inversely proportional; if the number of packets is more, the sample percent ratio is fewer, and vice versa. Please see [19,20] for further details on the second signature.

Some of the most important prefilter (PF) goals are listed below:

a) The PF is triggered after a DoS attack is detected in the network. It delivers a real-time notice to the neighbouring devices for them to respond to the assault.

b) The PF architecture guarantees that it can deduce stateful information about the network being monitored from stateless information for DoS attack detection, communicates a detected attack to the relevant command controller, and executes the command controller's reaction actions.

c) System stateful information, such as anomalous increases in traffic volume of a certain protocol, is a key signal of a DoS assault. The upkeep of stateful information is more difficult outside the confines of a local network. It degrades performance owing to a significantly larger demand on more extensive system monitoring. As a result, it is difficult for PF design to infer stateful information from stateless information to decrease monitoring effort.

4.2 Overview of the Command Controller Server

The server is responsible for the authentication and management of all the PFs within the domain. It has to authenticate and coordinate with all the adjacent servers and domains. The server of the domains consists of three modules namely Coordinator, PF Interactor, Authentication director as shown in Fig. 5.

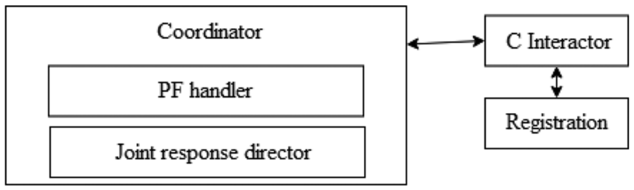


Figure 5: EDDDeP command controller server operation

• **Coordinator:** When an assault is reported by the PF, this module organises the attack response. This module is broken down into two submodules: PF handler and joint response director. The PF handler responds to an attack reported by the PF or a neighbouring domain by issuing a response directive. When the PF handler gets a message through C Interactor, it checks to see whether a comparable attack has already been issued. When a large amount of traffic enters the router, the PF handler invokes the joint response issue to send a collective response to all surrounding domains. The joint response director sends a response request to the nearby domains. The PF finds a set of nearby domains that are directly linked to their associated router and delivers a combined response to those domains only.

• **C interactor**: It authenticates all PFs in the domain as well as the server in the surrounding domains.

• **Registration**: It is in charge of registering all PFs in the domain as well as servers in neighbouring domains.

5 Implementation of EDDDeP System in SDN Environment

The packets are routed via the router's first-in-first-out (FIFO) mechanism. The resilient routing algorithm determines if the volume of incoming packets satisfies the threshold. If it does not, the router accepts incoming packets and places them in a queue. If the threshold is exceeded, the packets are dropped and picked up by the PF. Only two packets are compared at a time in this approach. The system administrator adjusts the number of packets compared to guarantee successful packet comparison.

One packet's destination address is compared to the destination address of another packet. If both matches, these packets are delivered for second-signature detection; otherwise, the current destination is saved for future comparison. When the traffic value is less than the threshold, the ingress router forwards the arriving packets to an unused Internet Protocol (IP) address; otherwise, the ingress router blocks the incoming packets after receiving the server's report. Fig. 6 depicts the activity flow in the EDDDeP domain system.

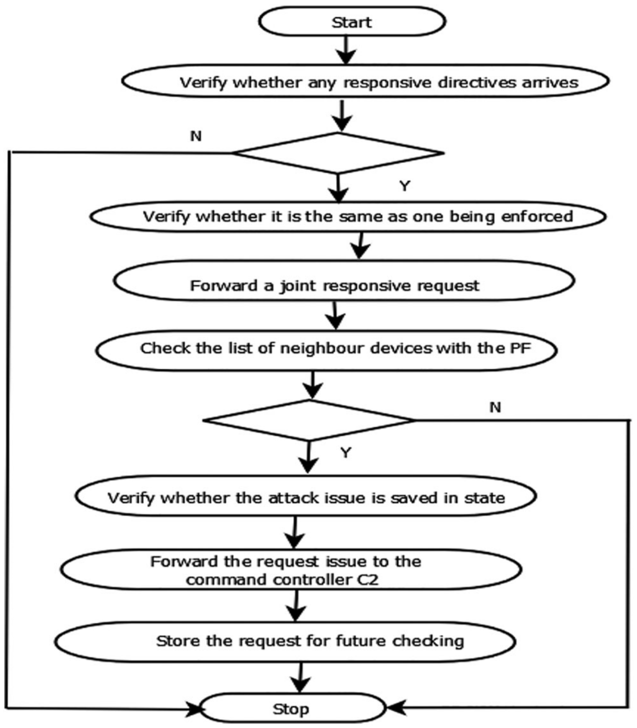

Start
Verify whether any responsive directives arrives
N
Y
Verify whether it is the same as one being enforced
Forward a joint responsive request
Check the list of neighbour devices with the PF
N
Y
Verify whether the attack issue is saved in state
Forward the request issue to the command controller C2
Store the request for future checking
Stop

Figure 6: The flow of activity in EDDDeP domain system in SDN

The hacker might provide bogus information to its neighbours, causing them to contact the controller and seek new regulations. This might jeopardize the operation and functionality of both the controller and the network devices. The major purpose of this attack is to raise the processing overhead of the controller and neighbours, as well as the network's packet traffic. The EDDDeP method is used to prevent erroneous information from spreading to neighbouring devices.

6 Results

In this section the evaluation of various testing scenarios of the proposed framework and the obtained results are discussed. For testing purpose, the authors in this paper simulate the DDoS attack but does not support any attack in network organizations, especially in SDN environment.

6.1 Test Description

We examined three test scenarios for the proposed EDDDeP framework, which are shown in Tab. 2. The attack is discovered to have to be mitigated to its source domains of three test scenarios. The suggested architecture has been tested on three different server domains. To neutralize a DOS attack over thousands of nodes in an SDN context, a separate testbed and network simulator are required.

Table 2: EDDDeP test scenarios

Test scenarios	Description	Network parameters considered	Host ping arguments
I	DDoS attack with spoofed IP address simulated with one malign host and two benign hosts	Round Trip Time, Packet loss and DDoS mitigation time	TCP-SYN, FLOOD
II	DDoS attack simulated with two malign hosts	Round Trip Time, Packet loss and DDoS mitigation time	TCP-SYN, FLOOD
III	DDoS attack with encapsulate packets size with one malign host and two benign hosts	Round Trip Time, Packet loss and DDoS mitigation time	TCP-SYN, FLOOD

6.2 Discussion on Testing Scenarios

This section contains a full overview of the proposed framework. The switch delivers the timestamp and quantity of packets transported in the network during packet transmission from source to destination inside the network. Tab. 3 displays the Quality of Service (QoS) metrics in three domain servers when no DDoS assault was reported. Tab. 4 displays three DDoS attack testing scenarios. For DDoS attack mitigation, the host IP address is 10.1.60.43, and the target machine is "DESKTOP-UQ61BU1". BurpSuite [21] was used to gather QoS measurements for DDoS attack mitigation time [22,23]. The remaining metrics, such as average Round Trip Time (RTT) and packet loss %, are obtained through host machine ping commands [24,25].

Table 3: The performance of different domain servers without DDoS Attack in EDDDeP

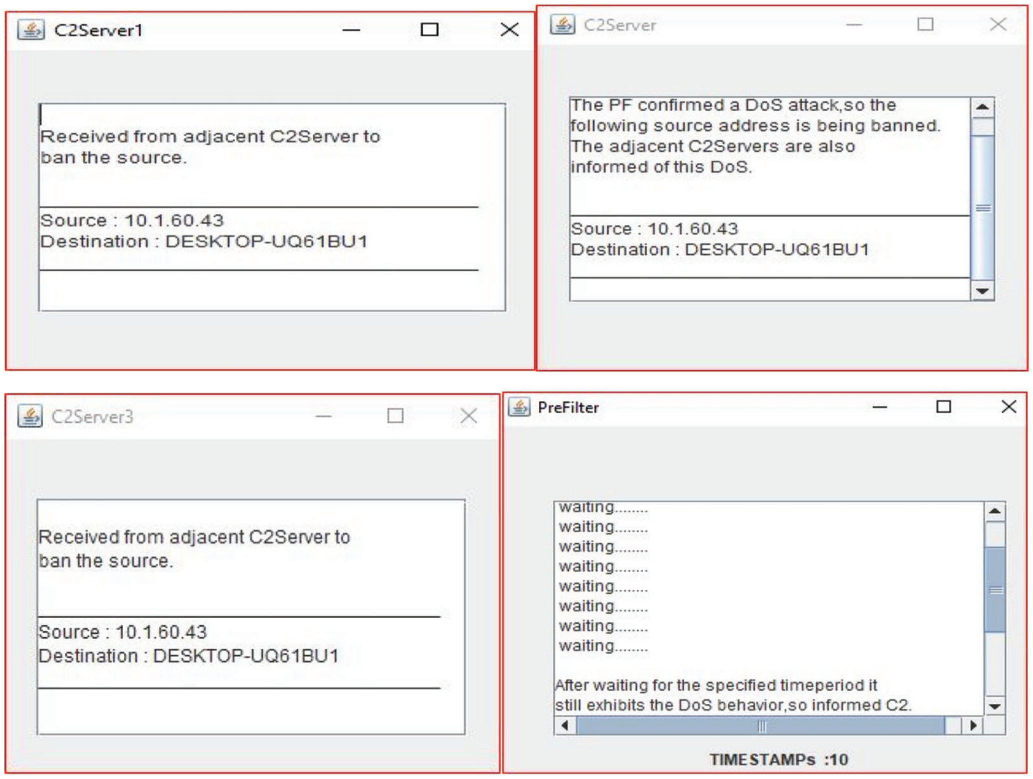
Test case no	C2Server1		C2Server2		C2Server3	
	Average RTT (ms)	Packet loss (%)	Average RTT (ms)	Packet loss (%)	Average RTT (ms)	Packet loss (%)
1	0.645	0	0.452	0	0.748	0
2	0.421	0	0.653	0	0.542	0
3	0.754	0	1.231	0	0.245	0
4	0.341	0	0.546	0	1.365	0
Average	0.540	0	0.720	0	0.725	0

The table shows that detecting and mitigating a DDoS attack takes 4–5 s in scenarios 1 and 2 in a multi-server system. We utilized several host IP addresses to test the third scenario for detecting and mitigating a DDoS assault, which took about 2–3 s. We created a JAVA eclipse Integrated Development Environment (IDE) environment application for real-time testing scenarios to identify DDoS attacks in network flows

[26,27]. When a DDoS attack is detected, it instantly propagates to neighboring domains in a multi-server environment for testing scenarios 1 and 2, as illustrated in Fig. 7. Fig. 8 depicts the testing of scenario 3 for DDoS detection and mitigation. This paper discusses DDoS detection in a multi-server scenario, DDoS with IP packet size manipulation, DDoS with IP Spoofing, and other related topics. As demonstrated in Tab. 5, the proposed framework can identify all the assaults with an average RTT of 0.538 ms and an average DDoS mitigation time of 2.25 s, with no packet loss. Fig. 9 depicts a performance comparison of QoS settings for several test situations. According to the findings of the suggested EDDDeP approach, DDoS attacks may be detected in less computing time.

Table 4: Performance of three test scenarios

Test no.	Test scenario I			Test scenario II			Test scenario III		
	DDOS mitigation time (s)	Average RTT (ms)	Packet loss (%)	DDOS mitigation time (s)	Average RTT (ms)	Packet loss (%)	DDOS mitigation time (s)	Average RTT (ms)	Packet loss (%)
1	2	0.621	0	2	0.321	0	3	0.464	0
2	4	0.487	0	3	0.236	0	2	0.491	0
3	5	0.654	0	2	0.471	0	2	0.482	0
4	6	0.753	0	3	0.638	0	2	0.364	0
Avg	4.25	0.628	0	2.5	0.416	0	2.25	0.450	0



Received from adjacent C2Server to ban the source.

Source : 10.1.60.43
Destination : DESKTOP-UQ61BU1

The PF confirmed a DoS attack,so the following source address is being banned. The adjacent C2Servers are also informed of this DoS.

Source : 10.1.60.43
Destination : DESKTOP-UQ61BU1

Received from adjacent C2Server to ban the source.

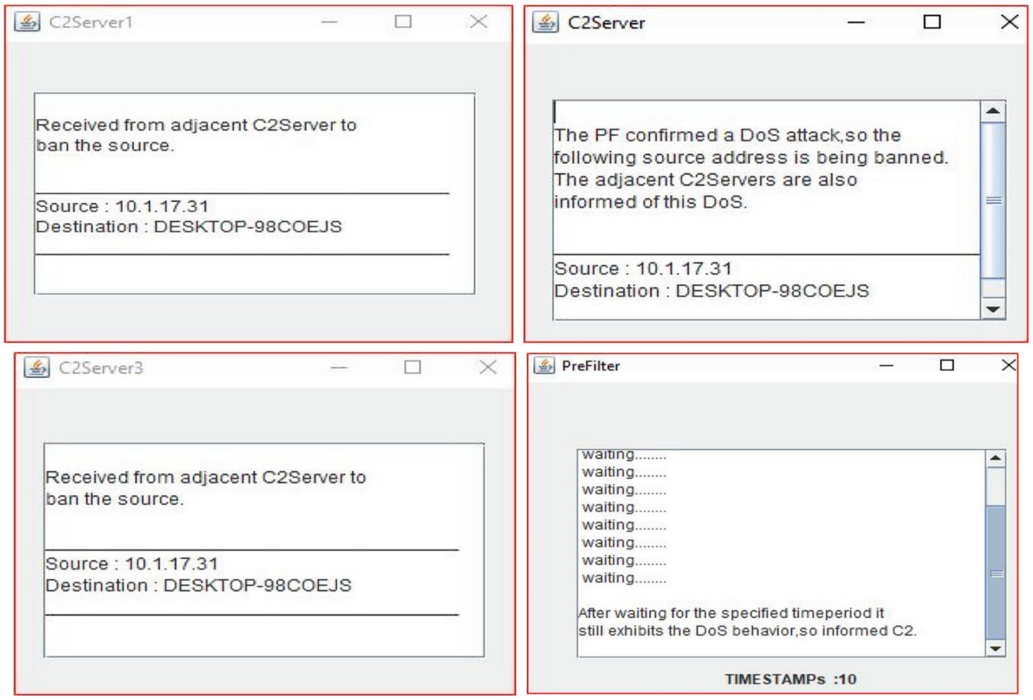
Source : 10.1.60.43
Destination : DESKTOP-UQ61BU1

```
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........

After waiting for the specified timeperiod it
still exhibits the DoS behavior,so informed C2.
```

TIMESTAMPs :10

Figure 7: DDoS in multiple server environment


C2Server1

Received from adjacent C2Server to ban the source.

Source : 10.1.17.31
Destination : DESKTOP-98COEJS

C2Server

The PF confirmed a DoS attack,so the following source address is being banned. The adjacent C2Servers are also informed of this DoS.

Source : 10.1.17.31
Destination : DESKTOP-98COEJS

C2Server3

Received from adjacent C2Server to ban the source.

Source : 10.1.17.31
Destination : DESKTOP-98COEJS

PreFilter

```
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........
waiting........

After waiting for the specified timeperiod it
still exhibits the DoS behavior,so informed C2.
```

TIMESTAMPs :10

Figure 8: DDoS in networking environment

Table 5: The results after comparing all scenarios of the proposed EDDDeP

Scenario	DDOS mitigation time (s)	Average RTT (ms)	Packet loss (%)
Normal behavior	—	0.661	0
Scenario I	4.25	0.628	0
Scenario II	2.5	0.416	0
Scenario III	2.25	0.450	0
Avg	2.25	0.538	0

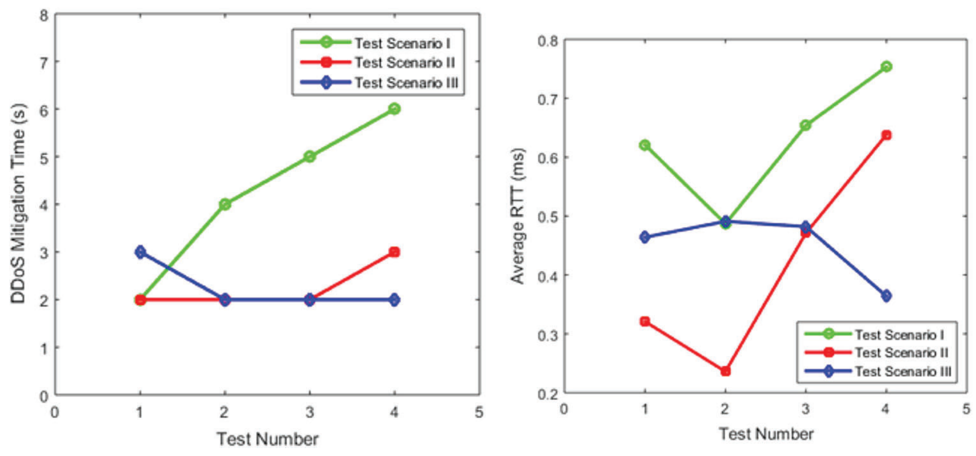


Figure 9: Comparison of QoS in different test scenarios

7 Conclusions and Future Work

The flow of information is the most important element for any network organisation, and a DDoS assault may seriously disrupt this flow. Several ways to defending against DDoS assaults have been presented in SDN. This work presents EDDDeP, a novel method that is more successful at detecting and preventing DDoS assaults in an SDN context. In this strategy, the server and PreFilter execute two-stage verification to guarantee the network is free of malicious traffic. The PreFilter node notifies the nearby neighbours after the traffic has been inspected and verified to be DDoS packets. The suggested system detects and mitigates DDoS assaults while using little processing effort and causing no packet loss. The findings show that the proposed framework can identify DDoS assaults with a low RTT and a short mitigation time. This work may be expanded in the future to incorporate a congestion algorithm for first and second signatures at all SDN levels. It combines them more closely at the network level in order to eliminate false warnings and increase system performance. Implementing EDDDeP at all SDN levels requires increased security and a trustworthy environment.

Acknowledgement: The authors sincerely acknowledge the support given by SRM University-AP and Madanapalle Institute of Technology & Science for this research.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declared that they have no conflicts of interest to report regarding the present study.

References

[1] R. L. Smeliansky, “SDN for network security,” in *Proc. IEEE Int. Science and Technology Conf. Modern Networking Technologies (MoNeTeC)*, Moscow, Russia, pp. 1–5, 2014.

[2] S. Bojjagani and V. N. Sastry, “Stamba: Security testing for Android mobile banking apps,” in *Advances in Intelligent Systems and Computing*, Proceedings of Second International Symposium on Signal Processing and Intelligent Recognition Systems (SIRS-2015) December 16-19, 2015, Trivandrum, India, vol. 425, pp. 671–683, 2016.

[3] S. Bojjagani and V. N. Sastry, “VAPTAi: A threat model for vulnerability assessment and penetration testing of android and iOS mobile banking apps,” in *Proc. IEEE 3rd Int. Conf. on Collaboration and Internet Computing (CIC)*, San Jose, CA, USA, pp. 77–86, 2017.

[4] L. Dridi and M. F. Zhani, “SDN-Guard: DoS attacks mitigation in sdn networks,” in *Proc. IEEE Cloudnet*, Pisa, Italy, pp. 212–217, 2016.

[5] D. Paulraj, “An automated exploring and learning model for data prediction using balanced CA-SVM,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 4979–4990, 2021.

[6] P. Mohann, “Resource selection in grid environment based on trust evaluation using feedback and performance,” *American Journal of Applied Sciences*, vol. 10, no. 8, pp. 924–930, 2013.

[7] M. A. Berlin and S. Tripathi, “IoT-based traffic prediction and traffic signal control system for smart city,” *Soft Computing*, vol. 25, no. 9, pp. 12241–12248, 2021.

[8] M. Prakash, R. Farah Sayeed, S. Princey and S. Priyanka, “Deployment of multicloud environment with avoidance of ddos attack and secured data privacy,” *International Journal of Applied Engineering Research*, vol. 10, no. 9, pp. 8121–8124, 2015.

[9] P. Subbulakshmi, “Mitigating eavesdropping by using fuzzy based MDPOP-Q learning approach and multilevel stackelberg game theoretic approach in wireless CRN,” *Cognitive Systems Research*, vol. 52, no. 2, pp. 853–861, 2018.

[10] S. T. Zargar, J. Joshi and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[11] M. Chhabra, B. Gupta and A. Almomani, “A novel solution to handle DDOS attack in MANET,” *Journal of Information Security*, vol. 4, no. 3, pp. 165–179, 2013.

[12] I. B. Mopari, S. G. Pukale and M. L. Dhore, “Detection of DDoS attack and defense against IP spoofing,” in *Proc. IEEE Int. Conf. on Advances in Computing, Communication and Control*, Karur, India, pp. 489–493, 2009.

[13] V. L. L. Thing, M. Sloman and N. Dulay, “Non-intrusive IP traceback for DDoS attacks,” in *Proc. ACM Symp. on Information, Communication and Computer Security*, Singapore, pp. 371–373, 2007.

[14] E. S. Madhan and R. Annamalai, “A novel approach for vehicle type classification and speed prediction using deep learning,” *Journal of Computational and Theoretical Nanoscience*, vol. 17, no. 5, pp. 2237–2242, 2020.

[15] D. Sterne, K. Djahandari, R. Balupari, W. La Cholter, B. Babson *et al.,* “Active network based DDoS defense,” in *Proc. IEEE DARPA Active Networks Conf. and Exposition*, San Francisco, CA, USA, pp. 193–203, 2002.

[16] S. Badotra and S. N. Panda, “SNORT based early DDoS detection system using opendaylight and open networking operating system in software defined networking,” *Cluster Computing*, vol. 24, no. 1, pp. 501–513, 2021.

[17] T. Ravichandran, “An efficient resource selection and binding model for job scheduling in grid,” *European Journal of Scientific Research*, vol. 81, no. 4, pp. 450–458, 2012.

[18] R. Kamalraj, M. Ranjith Kumar, V. Chandra Shekhar Rao, R. Anand and H. Singh, “Interpretable filter based convolutional neural network (IF-CNN) for glucose prediction and classification using PD-SS algorithm,” *Measurement*, vol. 183, no. 2, pp. 1–10, 2021.

[19] C. Ramalingam and P. Mohan, “Addressing semantics standards for cloud portability and interoperability in multi cloud environment,” *Symmetry*, vol. 13, no. 2, pp. 1–17, 2021.

[20] N. Subramanian, “A gradient boosted decision tree-based sentiment classification of twitter data,” *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 18, no. 4, pp. 1–21, 2020.

[21] BurpSuite, [Online]. Available: https://portswigger.net/burp/ (Accessed:10/07/2021).

[22] P. Asha, L. Natrayan, B. T. Geetha, J. Rene Beulah, R. Sumathy *et al.,* “IoT enabled environmental toxicology for air pollution monitoring using AI techniques,” *Environmental Research*, vol. 205, no. 1, pp. 1–15, 2022.

[23] D. Venu, A. V. R. Mayuri, S. Neelakandan, G. L. N. Murthy, N. Arulkumar *et al.,* “An efficient low complexity compression based optimal homomorphic encryption for secure fiber optic communication,” *Optik*, vol. 252, no. 1, pp. 1–13, 2022.

[24] S. Neelakandan, P. Mohan, A. Youseef, A. Saleh and I. K. Osamah, “An efficient metaheuristic-based clustering with routing protocol for underwater wireless sensor networks,” *Sensors*, vol. 22, no. 2, pp. 1–16, 2022.

[25] R. R. Bhukya, B. M. Hardas and T. Ch, “An automated word embedding with parameter tuned model for web crawling,” *Intelligent Automation & Soft Computing*, vol. 32, no. 3, pp. 1617–1632, 2022.

[26] C. Al-Atroshi, V. K. Nassa and B. Geetha, “Deep learning-based skin lesion diagnosis model using dermoscopic images,” *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 621–634, 2022.

[27] C. Pretty Diana Cyril, J. Rene Beulah, N. Subramani, M. Prakash, A. Harshavardhan *et al.,* “An automated learning model for sentiment analysis and data classification of Twitter data using balanced CA-SVM,” *Concurrent Engineering Research and Applications*, vol. 29, no. 4, pp. 386–395, 2021.