Int. j. inf. tecnol. (March 2024) 16(3):1627–1643
https://doi.org/10.1007/s41870-023-01469-3

ORIGINAL RESEARCH

Machine learning assisted snort and zeek in detecting DDoS attacks in software-defined networking

Muyideen AbdulRaheem[1] · Idowu Dauda Oladipo[1] · Agbotiname Lucky Imoize[2,3] · Joseph Bamidele Awotunde[1] · Cheng-Chi Lee[4,5] · Ghaniyyat Bolanle Balogun[1] · Joshua Oluwatobi Adeoti[1]

Received: 28 March 2023 / Accepted: 28 August 2023 / Published online: 13 September 2023

Abstract A new network architecture called the Software-Defined Network (SDN) gives next-generation networks a more flexible and efficiently controlled network architecture. Using the programmable central controller design, network supervisors may easily supervise and manage the entire infrastructure. However, due to its centralized structure, SDN has been a target of various attack vectors. The most successful attack method against the SDN among these has been Distributed Denial of Service (DDoS). Therefore, this study proposes a snort and Zeek enabled with machine learning (ML) based model to classify the benign traffic from DDoS attack traffic. This study main contribution is the discovery of new features for DDoS attack detection, which made it difficult to distinguish authorized traffic from attack traffic when spread across so many points of origin. Using the ML-based enabled RYU controller with SNORT and ZEEK created fewer false positives and a smaller variety of true positives per attack than the existing methods. The processing time of ML-based enabled with SNORT and ZEEK on the real-time testbed is better contrasted to the existing methods. Using the open resource technologies offered a far better understanding of cyber safety and its benefits from the readily available programs to construct a solid network keeping an eye on the traffic.

Keywords Software-defined networks · Snort · Zeek · Virtual machine · Distributed denial of service · Cybersecurity · Intrusion detection systems

1 Introduction

Conventional network architectures have not been able to fully satisfy some criteria, including high bandwidth, availability, quick connection rates, dynamic administration, cloud computing, and hybridization. Thus, an alternative to this is Software Defined Networking (SDN), which is dynamic, configurable, and adaptable is recently

✉ Cheng-Chi Lee
cclee@mail.fju.edu.tw

Muyideen AbdulRaheem
muyideen@unilorin.edu.ng

Idowu Dauda Oladipo
odidowu@unilorin.edu.ng

Agbotiname Lucky Imoize
aimoize@unilag.edu.ng

Joseph Bamidele Awotunde
awotunde.jb@unilorin.edu.ng

Ghaniyyat Bolanle Balogun
balogun.gb@unilorin.edu.ng

Joshua Oluwatobi Adeoti
adeotishuaolabode@yahoo.com

[1] Department of Computer Science, Faculty of Information and Communication Sciences, University of Ilorin, Ilorin 240003, Nigeria

[2] Department of Electrical and Electronics Engineering, Faculty of Engineering, University of Lagos, Akoka, Lagos 100213, Nigeria

[3] Department of Electrical Engineering and Information Technology, Institute of Digital Communication, Ruhr University, 44801 Bochum, Germany

[4] Research and Development Center for Physical Education, Health, and Information Technology, Department of Library and Information Science, Fu Jen Catholic University, New Taipei City 24205, Taiwan

[5] Department of Computer Science and Information Engineering, Asia University, Taichung City 41354, Taiwan

developed [1]. There are control, data, and application planes in SDN architecture. On the data plane are devices like switches and routers. The control plane is in charge of programming and controlling this plane [2]. The management of transmission equipment positioned on the data plane is the responsibility of the control plane. On this plane is the controller, which serves as the network's central processing unit. Devices on the data plane transmit packets under the controller's established guidelines [3].

The controller serves as the conduit via which the application plane interacts with the network infrastructure's devices. SDN has several benefits, including manageability, scaling, and enhanced performance. SDN, however, presents unique security challenges, particularly if its controller is vulnerable to Distributed Denial of Service (DDoS) assaults [3–5]. When DDoS assaults are launched against the SDN controller, the process and communication capacity of the controller are overburdened. The capacity of the switch flow table fills up as a result of the unnecessary flow created by the controller for the attack packets, which causes the network performance to drop below a critical level [6].

The flow table in the network devices exposes the data plane to DDoS attacks. Such DDoS assaults involve sending packets to the switch from unidentified sources. For each of these incoming packets, the controller creates a rule and transmits it to the switch's flow table. The switch flow table's capacity eventually runs out of space. New rules cannot be added to the flow table as a result, and packets cannot be forwarded. Also, as the assault causes the buffer capacity to fill to its maximum, arriving packets automatically drop [7]. Only the flow entry, which comes from the controller, is used by OpenFlow switches to manage packets. As a result, reducing DDoS assaults in SDN is more challenging than in traditional networks [8].

The controller, the network's brain in the SDN architecture, receives control functions from the switch. The controller's assistance makes it simple to apply parent-level rules to the network. The controller can modify and add new rules to the transmission devices [6, 9]. It can implement these changes by utilizing the OpenFlow protocol to communicate with transmission devices over a secure connection. Using this link, continuity and unity of the data stream are guaranteed. The link between the controller and transmission devices is compromised if this secure channel is broken. DDoS attacks are aimed at SDN architecture. As seen in Fig. 1, the attacker's three major goals when assaulting an SDN network are to consume the controller's sources, take up space on the channel connecting it to the switch, and overflow the switch's flow tables with unneeded flows [10]. In DDoS assaults on the controller, the attacker uses zombie users to send a lot of packets to the OpenFlow switch [11]. The controller has a hard time telling the difference between legitimate traffic and traffic sent by the attacker.

Using machine learning-based methods, SDN administration, security, and optimization problems can be solved in a more flexible, effective, and intelligent manner. The early detection of DDoS assaults is crucial for prompt action to be taken. Moreover, this study wants to integrate Snort and Zeek-based SDN with a built-in machine learning application as a reference model for constructing a secure framework. Therefore, this study proposes two intrusion detection systems (IDS) security mechanisms, SNORT and ZEEK, to secure SDN against DDoS attacks. In creating a virtual network, mininet shows to be beneficial. It has helped create several hosts and switches; VMs were created, Virtual Machine-1, Virtual Machine-2, and Virtual Machine-3. Virtual Machine-1 consists of Mininet and is combined with SNORT, an open-source network intrusion prevention system. In Virtual Machine-2, an RYU Controller is given

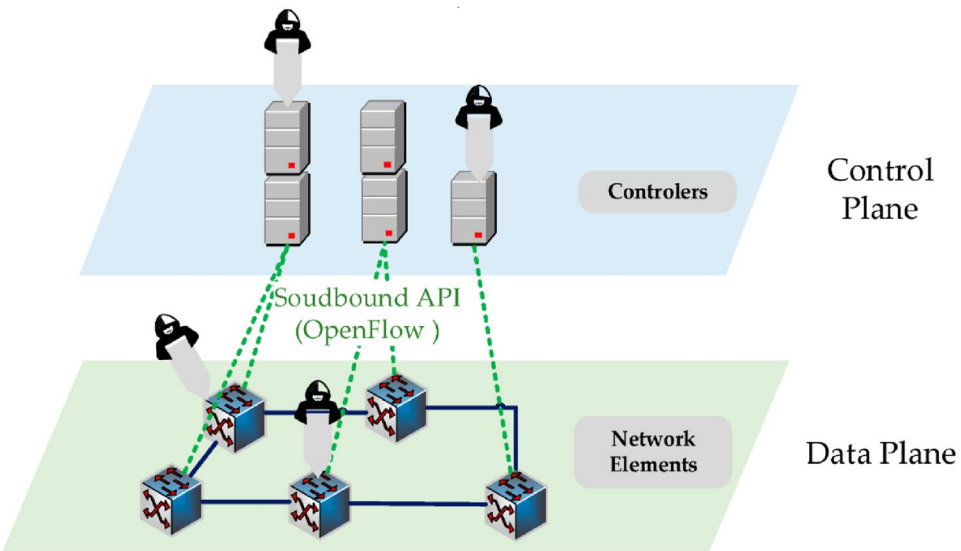

Fig. 1 Principal targets of the DDoS attacks on SDN networks

the multi-controller environment for experimentation. Kali Linux, an operating system used as a penetration tool, was installed on Virtual Machine-3 to introduce effective DDoS attacks. SNORT and ZEEK attacks with RYU controller specifications are more concise than utilizing software like Snort and Zeek independently.

1.1 Motivation

The controller is the most important aspect of setup in SDN, and it is always quick to detect a large series of attacks. The authors in [12] recognized the following four aspects that make SDN susceptible to attacks. In networking, a Software-defined network (SDN) is a system that supplies lots of benefits with the aid of dividing the knowledge of the network (controller) with the underlying network facilities (data plane). In SDN, Distributed Denial of Service (DDoS) is one of those attacks that end up being difficult for its advancement. Before a DDoS attack is mitigated, crucial action is taken to spot them. Because of that, this work analyzes the DDoS cyber-attack targeting SDN. Taxonomy mitigation of DDoS attacks in SDN is supplied. Authors in [13] developed a tool based on SNORT that detects a DDoS attack very early, which gives a great achievement to the SDN controller and does not compromise the functions of the whole network.

SDN is a modern networking technology that can potentially ease the constraints of present altering networking by completely disarranging the control and data plane, formerly performed inside routers and switches, and making it possible for a more versatile and viable one [1]. The control layer in SDN lies in a centralized controller that simplifies the enforcement of policies and the setup of network development [14]. High-level languages are used to design network policies rather than low-level, supplier-specific ones. Given that the controller makes use of beneficial network abstractions including an all-encompassing network opinion, flow administration, and analytics recording, application enhancement is simple. By adhering to the flow instructions provided by the control plane, switching devices become multifunctional devices. These features enable the creation of intricate network applications that interact with the controller's top level to offer improved network management and customization options [15]. Unfortunately, SDN architectural layout, regardless, when compared to standard networking, provides several points of failure.

The goal of this study is to classify SDN traffic as either regular or attack traffic using ML-based enabled sniff and Zeek software. The "DDoS attack SDN Dataset," which is freely accessible, has 23 features. The User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Internet Control Message Protocol (ICMP) are all represented in the dataset along with both legitimate and malicious traffic. Efficient modeling (characterization) and prevention of DDoS threats in SDN-based networks is a critical topic due to the increasing complexity of DDoS attacks and the vulnerability of SDN controllers. Even though this issue has been covered in various studies, more structured analytical models need to be developed to quantify how DoS/DDoS attacks affect the SDN control plane. In this study, the SDN is the main emphasis to provide a lightweight hybrid model outfitted with NCA and ML methods, which will help to guarantee a manageable network architecture for the next generation.

The key contributions are as follows:

1) this study proposes an enhancement technique to improve the security of SDN using ML-based enabled SNORT and ZEEK software.
2) detection of DDoS attack traffic in a network using ML-based model-enabled Snort IDS. Enhancement of Snort and Zeek to improve the security of SDN;
3) inspection of the enhanced system for mitigation of the DDoS attack, and the assessment of the SDN enhanced system performance level.

The other aspects of this work can be divided into diverse areas. Section 2 gives a detailed explanation of the work. The experimentation method is highlighted in Sect. 3. Section 4 illustrates the application and the results which were obtained from the experiment, while Sect. 5 concentrates on the conclusion and recommendation of this work in addition to the future scope given.

2 Related work

SDN has previously been widely used to improve network security [17] and efficiency [16]. However, recent research has identified several security flaws in SDN systems [18–20], leading to the conduct of numerous studies aimed at enhancing SDN resilience. Mahout [21] in particular suggested a method for preventing flooding threats in SDN. The problem's reliance on statistic aggregation, however, makes the solution unfeasible since data-to-control layer saturation attacks may take advantage of micro-flows. To combat DoS attacks, KernelDetect [17] proposed a minimal kernel space-based IDPS that makes use of modular string searching and filtering techniques. AVANT-GUARD [22] offered flow alteration management in OF devices as a way to lessen the effects of saturation attacks. Only flows that complete the TCP handshake are exposed by its design [28], which is based on a SYN proxy implementation and restricted to TCP saturation assaults.

Tian et al. in Ref. [23] described FlowSec and Blackbox, as two methods for reducing the DoS threat by monitoring

attack levels and enforcing the quantity of packets supplied to the controller per second. While PrioGuard [24] employs a non-cooperative repeated game as a method of DoS mitigation, FloodDefender [25] offers a protocol-independent alternative. FloodDefender uses table-miss engineering and packet filter techniques to secure both the control and data layers. Another defense against re-forwarded request-based flooding assaults in a multi-SDN context is FMD [26]. The re-forwarding rate is adjusted by FMD using an adaptive rate adjustment technique.

To improve controller security, MinDoS [27] also prevents DoS attacks by forwarding the flows to several buffer queues with different priority levels. To address the DoS impact on switch flow tables, the author in [28] offered path randomization and flow aggregation-based approach. Additionally, SDNManager is a compact framework for DoS prevention in SDN based on predictions of flow bandwidth change. Last but not least, secure controller techniques SECO [29] and SECOD [30] are used to guard against DoS attacks against the control plane and data plane of SDN.

The ratio of SYN packet counts to ACK flows generated by the same entities can also be used to detect SYN flooding traffic [31]. However, it is unrealistic to expand this method since it is difficult to effectively label dangerous flows in runtime and identify malicious packets from a vast traffic pool. By utilizing the SDN capabilities and infrastructure features, authors in [8] conducted research on how to address flooding concerns and enhance network resiliency. However, the expertise in detection and mitigation for complex SYN flooding attacks is still lacking (noting that the SDN modules that are being used may be susceptible to malicious SYN behavior).

Similarly, authors in [38] present a brand-new structure for autonomic security and threat detection called Distributed Threat Analytics and Response System (DTARS). The study developed exceptional guard usage under the DTARS system on a malicious testbed impersonating the authentic DDoS/botnets, for example, The Mirai.

Various researchers have either worked on suggesting or carrying an Intrusion Detection System (IDS). One of the detecting tools for DDoS that is being used frequently is [39]. This style uses mainly two types of aspects: Migrating the recognized popular link and the trigger of actuation are the two main aspects. Primarily, the components always have a type of proxy that it obtains. Due to this, it categorizes several TCP-SYN requests. Secondly, an event is created and sent to the controller once the main aspect is done with categorizing all destructive network traffic [13]. The log documents "Wi-Fi," which the SNORT device produces, is transferred to Wireshark, which looks at the captured network packages. The I/O graph in this research detailed the package circulation that provides an overall network that is evaluated in bytes or packets per second [40].

In another work, authors in [41] develop an MMS parser for the open-source Zeek IDS to examine MMS web traffic and spot breaches fully. Furthermore, to decrease the dealing with heaps, execute filtering guidelines in our parser to customize which MMS packets are used by Zeek guidelines for intrusion analysis. Their results reveal that custom filtering of MMS packages might obtain higher result as well as decreased hold-up compared to no filtering system.

Authors in [42] provided a unique formula for producing quick port checks that remain undiscovered by Zeek, subsequently revealing an issue that east–west cyberattacks might manipulate. Port scans were conducted on a physical exam and gotten in touch with a business-grade switch. Network security screening and an analysis device were used to create history site traffic from different application profiles. Speculative details exist which show that (i) the unique scans stay unnoticed by Zeek for a scanned price of approximately 1 million ports per 2nd and that (ii) neither Zeek nor Snort can find the unique scans if the scan price is reduced to 0.86 ports per minutes or less. Therefore, is recommended to incorporate the connection method with a customized being declined strategy for identifying the newly suggested fast port scans. It is concluded that this combined approach is possible for the more trustworthy discovery of port scans than personal strategies.

This study will concentrate on DDoS threats on SDN architecture and proposed snort and zeek supported by an ML-based model to detect attacks. As a result, the goal is to create DDoS attack detection systems for SDN architecture that are based on a high fertility rate ML model.

3 Materials and methods

The DoS/DDoS attack is the most serious hazard to the environment where SDNs are largely used. Based on this research paper, the concentration was directed towards focusing extensively on one type of attack. When traffic is dispersed across so many locations of origin, it is impossible to discern actual user traffic from attack traffic. The vulnerability of a network may include generating IP sender addresses (IP address spoofing) as a variation or augmentation of a DDoS attack, making identifying and countering the attack much more difficult. To secure SDN, this research provides two IDS security approaches that address DDoS. SNORT and ZEEK would be the IDS mechanisms used in this research. Figure 2 shows the snort and Zeek proposed experimentation, and Fig. 3 displays the snort.

Snort is executing real-time traffic evaluation as well as packet logging on IP networks. It executes method evaluation and web content searching/matching, as well as finds a variety of attacks.

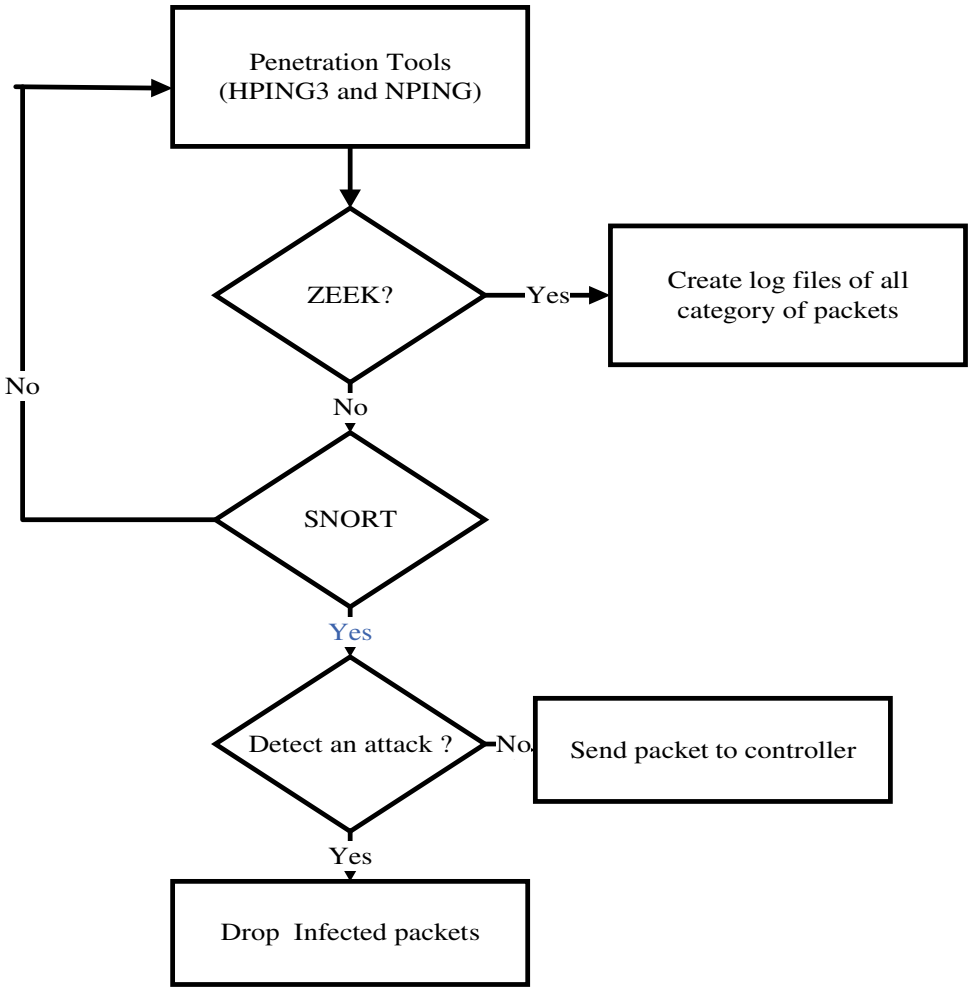

Fig. 2 Snort and Zeek proposed experimentation

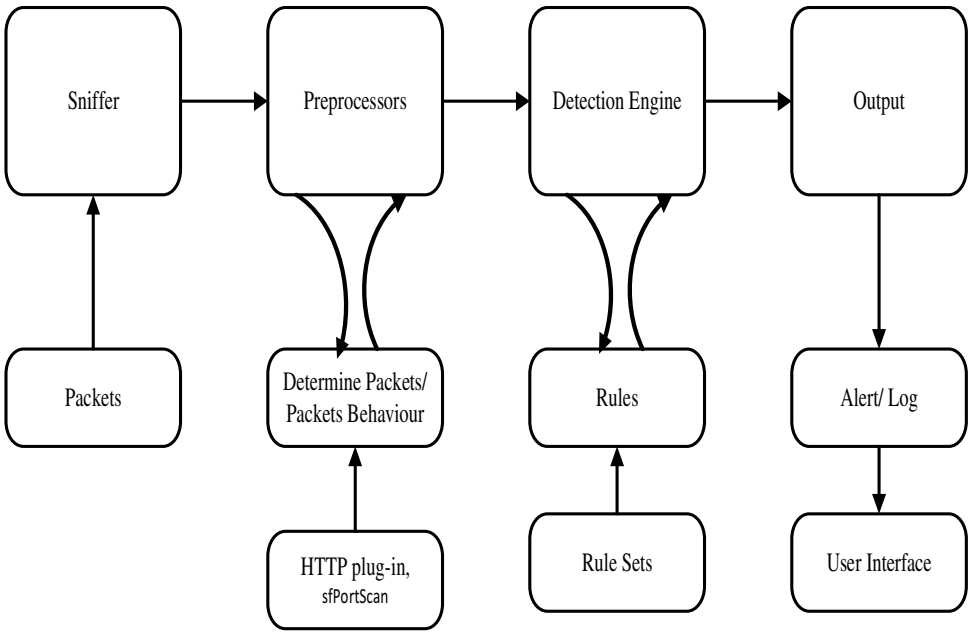

Fig. 3 Snort

3.1 Proposed implementation strategy

Many research studies have been studied on explaining the Mininet Emulation tool utilized to find various DDoS attacks versus the main controller [10]. This tool proves to be very helpful in developing a virtual network. It has assisted in developing numerous hosts and switches. Virtual Machine-1, Virtual Machine-2, and Virtual Machine-3 were created for our application. Mininet is included in the Virtual Machine-1, as well as SNORT, which is an open-source network invasion avoidance solution. In VM-2, an RYU Controller is offered the multi-controller environment for experimentation. VM-3 consists of Kalli Linux, which consists of a penetration tool to introduce efficient DDoS attacks. Figure 4 displayed the implementation framework approach.

3.2 Proposed evaluation technique

The proposed analysis technique has in view a detailed contrast between the existing methods and the proposed method concerning the advantages and disadvantages. There are various specifications used for various circumstances in the experiment. These criteria (with modifications in the variety), along with numerous situation, is referred to as follows:

3.2.1 Parameters used for evaluation

The range of information packages flooded towards the controllers from different infiltration gadgets. While determining this requirement, there is another sort of spec to get assessed, i.e., the kind of DDoS attack. From various seepage gadgets, various kinds of attacks are created. In this experiment, TCP-SYN flooding attacks were used. As soon as the DDoS attack launches, there is a requirement to know the minute when the IDS drops a package and the number of packets that were dropped. Analyzing the minute the IDS receives an attack and drops such packages is one of the essential specs for the research study.

3.2.2 Network scenarios

By making use of the Mininet emulation tool, various topologies are produced. These scenarios will be used to evaluate our findings in our experimentation.

Figure 4 shows the steps of the procedure used to obtain the results. Also, the classes of the open-source dataset utilized in this subsection for the ML-based model are discussed. The ML-based algorithm used to assess the classification dataset used is present.

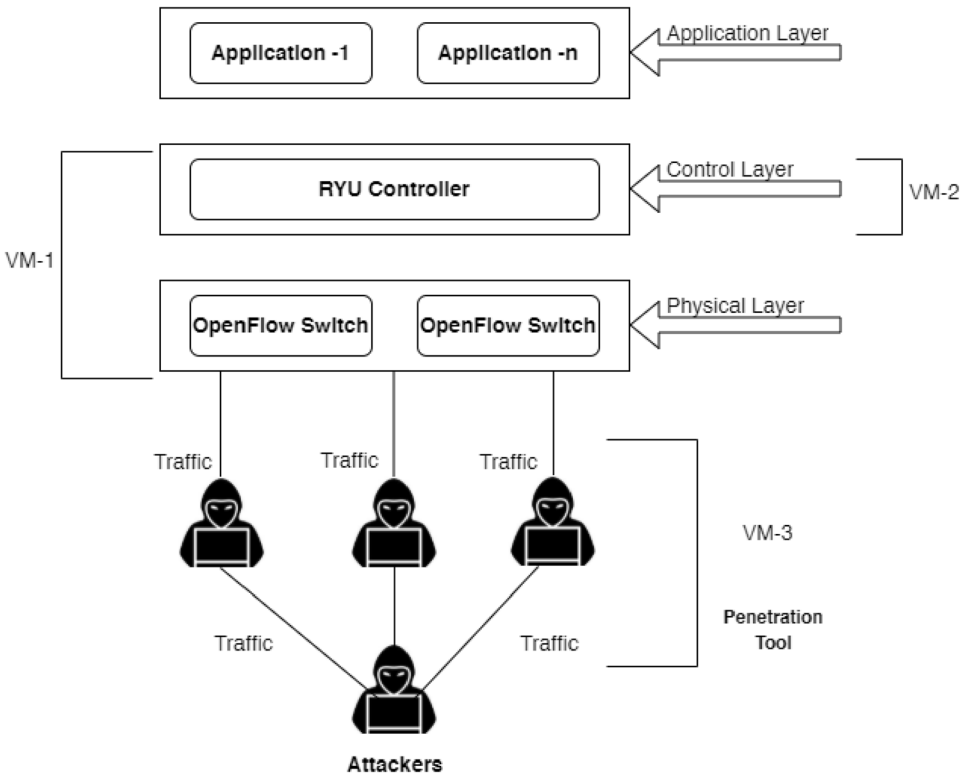

Fig. 4 Implementation framework strategy

3.2.3 Dataset

This study makes use of the "DDOS attack SDN Dataset," which was created in the SDN ecosystem and made accessible to experts for use in ML and DL investigations [14]. There are 23 characteristics in the dataset, which contains 104,345 traffic flows. The normal and attack traffic class designations are used to display the TCP, UDP, and ICMP traffic datasets. The dataset includes statistical parameters such as byte count, duration sec, packet rate, and packet per flow aside from those that describe the source and target devices. Before beginning to train an ML-based model, the data must be preprocessed. Since the properties for packet rate, bytes per flow, and packets per flow contain duplicate values, the dataset is now empty. For categorical data that lacks numeric values, such as source–destination IP and protocol, one-hot encoding was utilized [16]. The following phase involved employing an ML-based model and correlation approaches to try and determine the association between input features and output features. This method led to the discovery that the column with the "dt" characteristic that contained time information was superfluous and was thus eliminated from the dataset. By applying normalization to numerical data, the data pretreatment stage was completed. Algorithm 1 shows the generation of SDN dataset traffic.

motions, especially in network systems with high-density data flowing. In this subsection, the properties of the ML-based model employed in this study are briefly explained.

A scalable, distributed gradient-boosted decision tree (GBDT) ML-based package is called Extreme Gradient Boosting (XGBoost). The tree-based XGBoost [38] algorithm was developed, and the framework is gradient-boosting. It applies to tasks involving regression and classification ranking on tabular or structured datasets, which is relevant to our study. Utilizing parallel processing criteria, XGBoost constructs the trees using the depth-first search method and optimizes the gradient. To prevent overfitting/bias occurrences during the training phase, it applies a regularization term penalty. A tree ensemble model utilizes K additive functions to forecast the outcome for a dataset with n samples and m features (x, y) as follows.

$$Y = \sum_{k=1}^{k} fk(x_i), fk \in F \tag{1}$$

The space designated to regression trees, in this case, is F. The regularized objective used by XGBoost to try and optimize and minimize the loss function is as follows.

Input: Statistics on switch traffic. At each interval of 30 s, the switches' flow and port information are collected.

Output: After adding all the flow and port statistics, create an SDN traffic dataset.

Initialization: Packets per flow of normal traffic, packets per flow of attacks, the number of packet_in messages flows, and packet rates that were generated.

1: Compile flow and port statistics for each flow. Some of these are computed entries, as demonstrated below:

2: **for** $i = flow_1$ to $flow_n$ **do**

3: To produce the dataset, compile all of the switch's flow and port statistics into a single file.

4: Add the annotation "Attack traffic = 1" to the Dataset.

5: Mark the dataset with the annotation "Normal traffic = 0"

6: **end for**

7: **return** Interpreted Dataset.

3.2.4 ML-based model

ML-based algorithms are used to analyze system efficiency and unusual behaviors that depart from conventional network behavior are discovered. Mathematical models created using machine learning algorithms are used to quickly apply preventive rules to network systems and spot anomalous

$$I() = \sum_{1}^{k} I\left(Y_{true}, Y_{predicted}\right) + \sum_{i}^{k} \Omega(fk) \tag{2}$$

$$\Omega(fk) = \gamma * T + \frac{1}{2}\lambda |W|^2 \tag{3}$$

where $\frac{1}{2}\lambda|W|^2$ is the regularization term penalty and $\gamma * T$ is the model's complexity penalized term.

Performance evaluation

Various criteria were employed to assess the accuracy (AC), F-measure (F), recall (R), and precision (P) performance of the proposed models. Confusion matrix parameters can be used to obtain the following metrics: positive (how many anomalous occurrences are classified correctly); false positive (the percentage of normal occurrences that are mistakenly labeled as abnormal); true negative (the percentage of correctly identified normal cases); and false negative (the proportion of abnormal occurrences that are mistakenly labeled as regular). The Detection Rate (DR) and False Alarm Rate (FAR) of a good network intrusion detection system must be high.

Accuracy (AC): The percentage of network activity that was accurately categorized. Equations 4–7 presented the formula for the performance metric.

$$Accuracy = \frac{(TPTN)}{(TP + \mathrm{TN} + FP + FN)} \tag{4}$$

Precision (P): The proportion of expected anomalous occurrences that occur; the larger P, the smaller FAR.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall (R): The ratio of anticipated attack occurrences to all attack occurrences shown.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F-measure (F): using the harmonic mean of the P and R to gauge NIDS performance. A high F-score is what we are going for.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

4 Experimental results

4.1 Experimental setup

In this research study, a Mininet Virtual Machine (VM) was hosted and set up on the Ubuntu 20.04 OS. With the mininet setup, a software-defined network was simulated utilizing a Python script. Topology, Switch, and Controller are the three essential functions of this script. Mininet has been stopped from developing a network utilizing default worths to create a network. The controller class must specify the remote gadget and determine the name and IP address of the Controller gadget IP address. Port 6633 has been set (Table 1).

The performance dimensions for various versions of passing traffic through Snort and Zeek (Bro) were developed, starting with scenarios of packets of information for tracking differences in the facilities surrounding the endpoints. To verify the Snort version, open a terminal and type snort -V followed by entering. Next, create rules to allow snort to detect a DDoS attack. For example, use the command Sudo gedit/etc./snort/rules/local.rules to open the local. Rules file in a text editor as root. Next, establish policies or, better yet, recommendations for configuring the HOME NET value: the network to be safeguarded.

Secondly, install Zeek (Bro) on the Ubuntu operating system; Zeek is the network information everybody wishes they had. When a security alert fires or you have a network vulnerability to examine, Bro aids in discovering the issue faster. It matches signature-based tools to help rapidly map complicated events across multiple circulations and methods easily to determine and resolve safety concerns quickly. On that, Zeek (Bro) can be configured to act in the evaluation of events (e.g. to perform a program to supply active action for the discovered event) in addition to supplying forensics capabilities thanks to its event logging system. For the following series of runs, the Ryu controller will be added to manage the switch. The Ryu controller was started in addition to the tests that have been performed. Our tests could show that Zeek can split traffic over several IDS sensing systems.

4.2 Traffic generation and event rules enactment

Under simulated real-world networking circumstances, preliminary tests are conducted to gauge the system's response time to attacks. Here, two forms of network traffic intent—adverse (malicious) and normal—are specifically taken into account (benign). The network is then filled with packets created from these flows. Additionally, to produce packet traffic and quantify/control it across the network, the distributed Internet traffic generator (D-ITG) and iPerf toolkits are utilized together. It should be noted that iPerf also permits the saturation of edge network links, which can be used as a baseline technique to assess the effectiveness of reaction to saturation attacks.

Additionally, rule-based detection is also implemented here using the Zeek and Snort IDS toolkits. In other words, if pre-defined detection criteria have previously been implemented in a certain version, they are manually specified or added. The only rules utilized for Zeek and Snort, however, are those for SYN flood attack detection and mitigation to ensure proper performance comparison. Comparing the related overheads of these tools with those of other rules activated will be meaningless because both of them can handle a wider variety of network threats. (and as a result,

Table 1 Summary of ML-based Algorithms against DDoS attacks in SND

Solution	Models	Dataset	Remarks
Hnamte & Hussain (2023) [8]	DCNN	CIC-IDS2017 and the CIC-DDoS2019	The technique only had a 0.0016 loss rate while achieving a 99.99% accuracy rate
Chin, et al. (2017) [17]	lightweight kernelDirect	Real-world testbed	A reliable mechanism that outperforms SNORT and BRO in terms of threat identification and prevention. But failed to test on DDoS attacks
Shin et al. (2013) [22]	connection migration	Self-dataset	A realization of our two data plane extensions called Avant-Guard assesses the performance impact and looks at how it may be used to create more resilient and scalable SDN security services
Tian et al., (2019) [23]	FlowSec and Blackbox	Not mentioned	The framework failed to properly identify attacks on the SDN network
Wu et al., (2018) [26]	flow migration defense	Self-generated	Simulations show that model effectively mitigates the SDN-targeted DDoS with superior efficiency in terms of request response time, packet loss rate, and mitigation time compared to multilayer feedback queue and FloodDefender
Wang et al., (2018) [27]	MinDoS	Self-generated	The findings demonstrate that MinDoS is efficient and very slightly increases overall SDN/OpenFlow network complexity
Bharathi et al. (2019) [28]	Path randomization	self-generated traffic	An evaluation of the system's performance in a simulated environment produced a favorable outcome
Alshamrani et al. (2017) [32]	ML-based	Self-dataset	The devised feature selection technique improved the accuracy of attack detection by choosing an appropriate number of features from the NSL
Latah & Toker (2018) [33]	Enhanced SVM	Self-generated dataset	The system framework has a false alarm rate of 0.26% and a 96.25% accuracy in detecting DoS flooding attacks
Li, et al. (2018) [34]	DL-based	Self-generated dataset	The model reduces the complexity of updating or altering the detection strategy, as well as the real-time updating of the monitoring system
Ye et al. (2018) [35]	SVM model	mininet and floodlight	The findings indicate that, with a minimal amount of flow gathering, our approach has an average accuracy rate of 95.24%
Li et al. (2015) [36]	CV-GA-SVM	Private dataset	Detect DDoS attacks using the cross-validation-genetic algorithm (CV-GA) using SVM-optimized parameters c and g
Jain & Anubha (2021) [40]	Router enabled Snort	Self-generated	The model failed to detect DDoS attacks with a high rate of false positives
Jankowski & Amanowicz (2016) [44]	Selected ML-based	Adopted Mininet	The model just monitors the SDN network without attack detection

Table 1 (continued)

Solution	Models	Dataset	Remarks
Mowla et al. (2018) [45]	SVM & Logistic Regression	traffic produced by oneself	The model does not use ML-based techniques to make decisions or dynamically inject security rules, which we consider to be potential future work
Wang et al. [24]	SVM, KNN, NB, & ANN	PC1-PC6 virtual machines	The findings imply that feature selection and ML-based Models can improve DDoS attack detection in SDN while promisingly reducing processing load and time
Proposed model	SVM, GLM, NB, DA, FNN, DT, KNN and BT	DARPA, InSDN, and self-generated traffic	Multiple techniques have over 99% accuracy in the simulation, and over 90% in the real-time detection

Fig. 5 Attacking controller with Hping3

```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -c 1000000 -d 1000 -S -p 80 --flood --rand-source 192.168.0.1
68
HPING 192.168.0.168 (eth0 192.168.0.168): S set, 40 headers + 1000 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.168 hping statistic ---
177196 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Fig. 6 Attacking controller with Nping

```
┌──(root💀kali)-[/home/kali]
└─# nping --tcp-connect -rate=50000 -c 900000 -q 192.168.0.168

Starting Nping 0.7.92 ( https://nmap.org/nping ) at 2022-01-11 03:58 EST
^CMax rtt: N/A | Min rtt: N/A | Avg rtt: N/A
TCP connection attempts: 109334 | Successful connections: 0 | Failed: 109334
(100.00%)
Nping done: 1 IP address pinged in 23.03 seconds
```

they are disabled or offloaded). Because of this, the sample scripts in Figs. 5 and 6 also demonstrate the actual Snort and Zeek IDS tools detection rules for SYN-based flood traffic. For example, anytime 70 or more packets arrive during a 10 s interval, Snort issues a TCP-SYN flagged threat alert based on a default threshold value, as shown in Fig. 5. Figure 6 shows Zeek using a predetermined threshold, n, which is manually adjustable during configuration setup. When this amount is surpassed, an alarm flag is raised. This parameter keeps track of how many TCP connections have failed.

4.3 Inspection time

The most important factor that directly affects the reaction times of harmful flows is the inspection time for incoming flows. In particular, the associated packet data must be placed into a buffer while awaiting inspection when flows are intercepted by the IDPS. This waiting period will inevitably

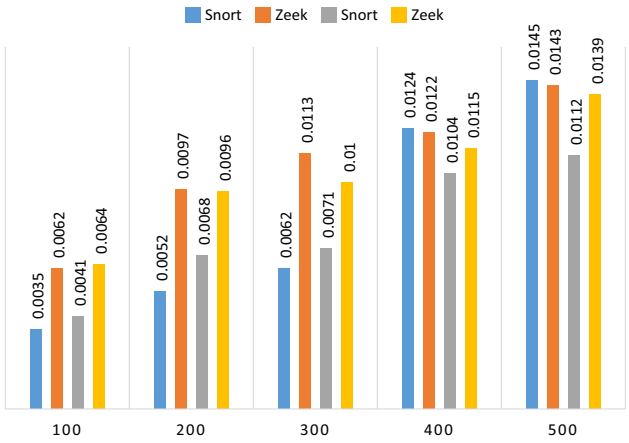

Fig. 7 Time on average for inspecting various SYN flooding packet DoS attacks (10 s threshold in Snort and Zeek)

Table 2 Different machine specification

Name of the VM	IP address	Specification
Emulation tool Mininet (VM-1); SNORT IDS and ZEEK	192.168.0.168	64-bit Ubuntu VM
RYU Controller (VM-2)	192.168.0.167	64-bit Ubuntu VM
Kali Linux (VM-3)	192.168.0.180	64-bit Ubuntu VM

extend the amount of time needed to mitigate the attack if an alert is issued. Therefore, both the Snort and Zeek solutions are implemented with almost identical settings to evaluate the inspection times of the suggested solution. Each IDPS solution is set up with comparable threat detection signatures for evaluation fairness.

The hping3 tool is used to build a communication scenario between two hosts to estimate inspection times. To achieve a high network link saturation, smaller-sized packets are specifically generated at a high rate in a very brief time window (for example, 1 ns). It should be noted that the proposed approach automatically modifies its detection threshold, whereas the Zeek and Snort schemes both keep this number static. (and must therefore be manually set at the time of initialization). Therefore, the default values for each of these static detection thresholds are set to 10 s. Additionally, automatic experimental runs are carried out for 15 subsequent trials, and the results are averaged, as shown in Fig. 7 and Table 2. For instance, Zeek provides inspection times that are over 100% greater, whereas Snort provides latencies that are 10–20% higher. However, for some randomized runs, Snort can perform as well as the other method (but not better).

5 Mitigation and response time

Next, think about the mitigation time, which is the reaction time to threatening opponent flows. The mitigation time is specifically described as the interval between generating an attack alarm and performing the necessary associated action, such as dropping, blocking, etc., once a given flow has been identified and tagged as malicious. This delay is quite significant in SDN contexts for ensuring the availability of critical operational resources. Although each IDPS scheme's detection attributes (for malicious traffic) are now distinct, the Snort and Zeek mitigate time. This value is calculated as the interval between the start of an attack and its correction.

The first presumption is that each IDPS scheme starts blocking activity against hazardous IP addresses as soon as an attack alert is produced. The two IDPS solutions' average mitigation times, inspection times, and resource usage for the same experimental configuration as previously described

Table 3 System load and typical inspection and mitigation times

Traffic Load (k)		100	200	300	400	500
Metric	*IDPS*					
Inspection (s)	Snort	0.0035	0.0052	0.0062	0.0124	0.0145
	Zeek	0.0062	0.0097	0.0113	0.0122	0.0143
Mitigation (s)	Snort	0.0041	0.0068	0.0071	0.0104	0.0112
	Zeek	0.0064	0.0096	0.0100	0.0115	0.0139
System (%)	Snort	8.13	15.02	16.21	18.23	19.72
	Zeek	16.32	25.17	26.31	32.21	34.20

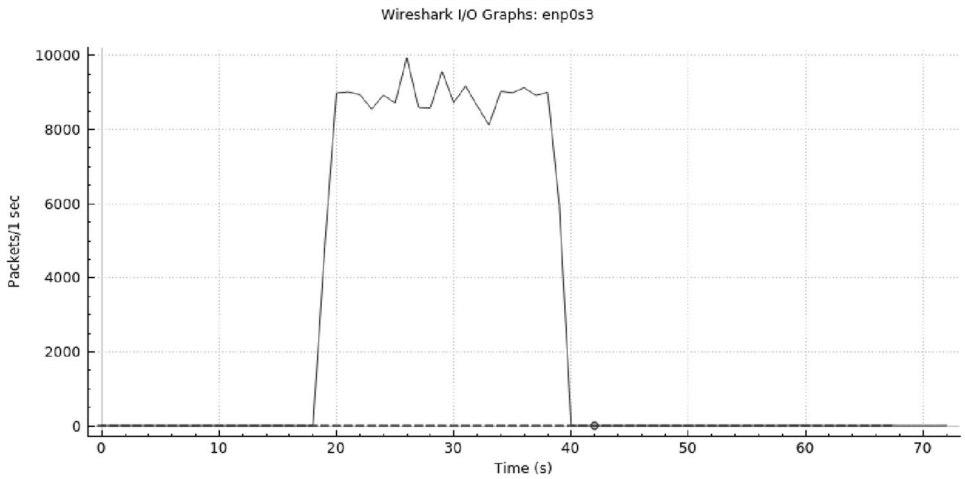

Fig. 8 TCP traffic flood of packets

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   reporter
#open   2022-01-07-12-53-34
#fields ts      level   message location
#types  time    enum    string  string
1641540214.212396       Reporter::INFO  received termination signal     (empty)
1641540214.212396       Reporter::INFO  805 packets received on interface enp0s3, 0 (0.00%) dropped     (empty)
```

Fig. 9 ZEEK dropped zero packets in an ICMP traffic

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   reporter
#open   2022-01-08-06-02-25
#fields ts      level   message location
#types  time    enum    string  string
1641601945.850675       Reporter::INFO  received termination signal     (empty)
1641601945.850675       Reporter::INFO  113760 packets received on interface enp0s3, 694456 (85.92%) dropped    (empty)
```

Fig. 10 ZEEK dropped packets received from a flood attack

the threshold initialization, threat detection signature, etc., are summarized in Table 1. Keep in mind that the experimental results in Table 3 are an average of 25 runs. For the scenarios of 100 and 500 K SYN flagged packets, as shown in Table 1, Snort outperforms Zeek in terms of all three performance parameters, namely inspection time, mitigation time, and the percentage of system load. Snort still outperforms Zeek in all three performance categories while dealing with 200 K SYN-flagged packets.

5.1 Overhead evaluation

Now, overly intensive inspection activities may result in resource depletion on machines running the IDPS software (due to the overheads involved in inspecting packets). To determine system overheads, the memory usage for each IDS technique for a single assault with 100,000 SYN-flagged packets is also plotted in Fig. 8. Here, the default detection threshold is used by the Zeek and Snort solutions (10 s). Please take careful note that in this attack scenario, link saturation also happens. These findings unambiguously demonstrate that the Snort technique uses less RAM than the Zeek solution.

Many penetration tools, such as nping and hping, were used throughout the implementation of the work on a vSwitch with and without a controller, as well as with and without Snort hosts keeping track of redirected traffic.

All simulations and experiments were carried out in a Linux-based virtual machine, as shown in Table 3. Both Kali Linux and the RYU controller are intertwined with the same network, having the same IP addresses, 192.168.0.167 and 192.168.0.126. TCP-SYN attacks were used in the experiment. These numerous penetration tools were launched from the Kalli Linux VM and successfully went through the

Table 4 Snort packet received

	Packet logged
Snort	3,612,051
Zeek	3,582,374

Table 5 Snort packet logs

	Packet logged	Packet dropped	Packet analyzed
Snort	3,612,051	2,935,639	676,411
Zeek	3,582,374	3,396,788	185,586

DDoS attack on the Virtual Machine-3, while Hping3 and Nping were used for TCP SYN attacks. Snort did not meet any type of failure in its principal mission of examining all traffic participating in the network in any of the experiments conducted within the boundaries of the infrastructure. As it turned out, we reduced the overhead for both an expert and a specialist by only requiring a single manufacturer for varied Snort handling.

As an IDS/IPS, Snort does a deep assessment of the incoming packets, then tags packets with a trademark that will detect and possibly obstruct the traffic. Snort is an energetic blocker rather than a passive system when configured with certain policies or rules. Zeek is similar to Snort, but as opposed to proactively obstructing traffic that is regarded as malicious, it functions as an easy network display as well as a traffic analyzer and will certainly notify the individual that harmful code has potentially gone into the system and also needs to be dealt with.

From the experimentation performed and shown in Figs. 9 and 10, it was realized that there was an event of the flooding of substantial (TCP SYN) network traffic

Table 6 Snort percentage dropped and received

IDS	Percentage of dropped packets	Percentage of packets analyzed
Snort	81.27%	18.73%
Zeek	94.82%	5.18%

toward the RYU controller. It can be seen that the SDN controller is vulnerable to DDoS attacks. While the DDoS attacks were being identified, other crucial criteria were also included. This shows that the amount of traffic pummeled and the number of network devices (hosts, switches) used is directly linked to the DDoS detection time. To obtain the information for the DDoS attack SNORT and ZEEK were configured, and rules were constructed. As a result, it may be inferred that DDoS detection has been performed, considerable data about lost packages have been collected, and ideal actions can be taken afterward.

The author in [43] stated that using Raspberry Pi with SNORT and ZEEK carries out a deep examination of the inbound packets and tags packets with a trademark that will certainly spot and obstruct the traffic, which is restricted to safeguarding a home network or a small network. It also has problems with overheating and needs to be placed in such a way that a cooling fan is attached to it. In this paper, SNORT and ZEEK attacks with RYU controller specifications are more concise than software like Snort and Zeek independently. Tables 4 and 5 shown the Snort packet received and packet logs respectively (Table 6).

The proposed method DDoS detection system is based on several metrics, including the time it takes to locate the DDoS attack, the number of packages found, and the percentage of packets dropped and inspected. Unlike ZEEK, SNORT takes the least time to discover a successful DDoS attack and drops fewer packets. In addition, ZEEK has a comprehensive assessment log of all attacks in the network, which provides a practical network analysis of all traffic. The enhancement guarantees the prompt identification of DDoS attacks that provides better performance of the SDN controller and does not jeopardize the total functionality of the network.

Using the RYU controller with SNORT and ZEEK created fewer false positives as well as a smaller variety of true positives per attack than the author in [43].

Table 7 The performance of the proposed models

	Accuracy	Precision	Recall	F1-Score
XGBoost	97.2%	94.9%	98.5%	95.9%

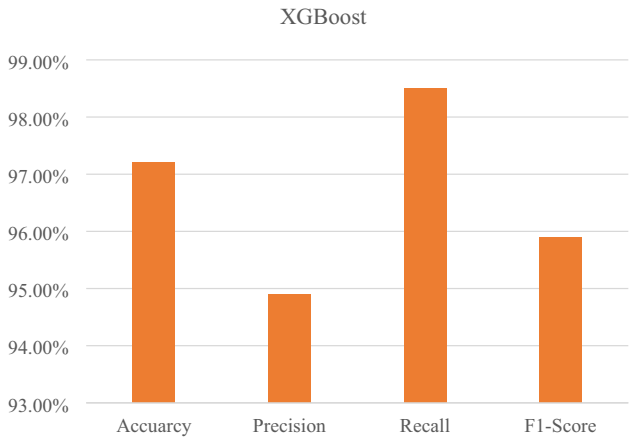

Fig. 11 Performance of the proposed classifier

In addition, the processing time of SNORT and ZEEK on the real-time testbed is better contrasted to that of the author in [43]. Finally, using the open resource technologies offered a far better understanding of cyber safety and its benefits from the readily available programs to construct a solid network keeping an eye on the traffic.

5.2 The performance of the ML-based model

In this study, the dataset was used to assess data on SDN architecture during periods of normal traffic and periods of DDoS attack traffic. The findings of the proposed classifier are shown in Table 7 (Fig. 11).

It was discovered after looking at the performance results that the proposed classifier was effective in looking at the attack data. Table 8 shows the results of the confusion matrix using the XGBoost classifier. We found that the performance rate is impacted by the attack traffic types of ICMP, TCP, and UDP. When the UDP flood attack was conducted, the greatest effectiveness rate was attained.

Table 9 shows that when the proposed model is compared to the findings from [32, 33, 35, 37, 44–46], the model outperformed the existing models. It should be emphasized that comparable studies in the literature have employed various datasets and modeling approaches. Thus, it is challenging to justify the comparison results.

Findings indicate that by using ML-based approaches, the SDN structure was successful in identifying DDoS attacks. A secure and effective network mechanism can be created with the planning methods for the SDN architecture. Capturing the packets in the stream and trying to compare them to the historically understood flow data of the packet helps improve the performance of the controllers on big networks. Using a two-stage technique to identify DDoS attacks on the dataset could be another strategy.

Table 8 The results of the proposed model in terms of the confusion matrix

Predicted Class							
		Normal	ICMP	TCP	UDP	TP	FN
Actual Class	Normal	99.9%	0.09%	0.16%	0.23%	99.6%	0.4%
	ICMP	7.6%	92.0%	1.70%		90.7%	9.2%
	TCP	12.9%	1.19%	85.8%	0.45%	86.0%	14.7%
	UDP	0.48%	0.37%		99.5%	99.7%	0.6%

Table 9 Comparison of the Proposed Model with existing similar works in the literature

References	Dataset	Model	Accuracy
[32]	NSL-KDD DATASET (2009)	SMO	95.9%
		J48	92.7%
		NB	83.9%
[33]	Dataset they created	SVM	96.3%
[35]	Authors own dataset	SVM	95.3%
[37]	LongTail	C4.5	86.2%
		Bayesian Network	91.7%
		DT	88.2%
		Naive Bayes	87.9%
[44]	Authors own dataset	SOM	94.0%
		LVQ	
[45]	CAIDA DDoS 2007	SVM	93.0%
[46]	Authors own dataset	SVM	92.5%
		KNN	96.3%
		ANN	92.1%
		NB	95.1%
Proposed Model	DDOS attack SDN Dataset	XGBoost	97.2%

The coming flow data is copied to another module while the controller processes it for analysis.

6 Discussion

Based on the experiment, it has been made to recognize that ZEEK and SNORT can both work together flawlessly without impeding each other while functioning. SNORT alerts for an attack while ZEEK works quietly in the background spoofing the network for any irregularities. ZEEK is a powerful software for putting down every network abnormality or normality participating in the network. Based upon previous understanding, people only believe SNORT is an effective software for reducing DDoS attacks however based on our experiment, it has been made understood by the experimentation that ZEEK is a good software. It can aid in putting down any vulnerability in a network. With the two IDS interacting, it has been realized that there becomes a much more secure network from DDoS attack, and also it happened, it can be traced to where it originated from. Nothing passes by ZEEK.

Making use of both Snort and Zeek did provide their interesting sides. For Snort, the program does a much more hostile check of the inbound packets if it is set with certain policies as well as assesses. For Zeek, instead of doing a hostile check of each packet, it will certainly log all of the packets as they flow through and also enter them into a log that the individuals can evaluate as well as will certainly additionally flag the particular action that it deems as malicious. After the experimentation, this research study is limited to the specifications that were utilized, the tools for penetration testing, the variety of packets flooded, and simply a type of DDoS attack. Running these programs does offer an appropriate NIDS/NIPS system for a network as it can examine and also keep an eye on the traffic flowing to as well as from the internet. Yet, it was noticed that the system used was heating up based on the simulation and also the working of the 3 VMs being executed.

As SDN is getting appealed each day, its need is likewise rising. With the numerous advantages, there exist countless challenges. Security can be a barrier in SDN. The separation of the control and data planes becomes a point of vulnerability for attackers. DDoS assault is one of the most common

threats in SDN, which is why we developed a solution to improve SDN security. We recommend more research be done on enhancing the network from a prominent attack like DDoS and how we can enhance more IDS together to mitigate such attacks.

The inspection module now watches and categorizes any TCP SYN requests that are established TCP handshakes after the two components of the software have been initialized, while also keeping track of broken connections and source-side connection timeouts. Once a connection request (i.e., the first SYN packet) is received at the data layer, the OVS checks its flow table for a matching TCP ACK forwarding rule or asks the SDN controller for information. Keep in mind that the OVS device also checks to see if the nature of the received packet is a TCP SYN (before verifying that the TCP handshaking session has ended) to recognize unwelcome packets.

The packet inspection process is launched at this point by the initial state, which also sets all the input and global variables used in the inspection and mitigation modules. A string matching operation is performed when a malicious TCP SYN event has been identified in the Inspection state, and then an alert is generated. After that, the system enters the preventive state to lessen the attack if the current threshold value for observed events is exceeded. To update the current detection threshold, the system also uses the threshold update function.

The proposed ML-based model employs a variety of methods to produce a verifiable outcome. Although using a different dataset, the suggested model was compared to various already-existing models for precise literature and comparison. Yet, the dataset utilized is regarded as one of the most potent benchmark datasets. On the dataset, some data statistics, cleaning, and verification processes are carried out. These procedures are essential to ensuring that the learning process is efficient and free of obstacles like over- or under-fitting issues. At this point, the proposed model is ensured to contain unified data, increasing the value of the data and assisting decision-making. To evaluate XGBoost's performance using various performance metrics, further investigation and comparisons are then conducted.

The switch will finally go from the idle state to the inspection state, which will begin traffic inspection before forwarding it to the prospective target after a flow has been received at the relevant OVS port. At startup, it is anticipated that the network administrator will also define or enter a default threshold value. The IDPS system also supports a secure mode option, which enables it to perform a deeper deep packet (payload) inspection utilizing exact string matching when dealing with sensitive data. Therefore, the IDPS will only carry out partial string matching if the switch is situated in a trustworthy area.

The proposed framework has the disadvantages of being only appropriate for flooding DDoS attacks being unable to recognize non-volumetric attacks like low-rate DDoS attacks, and only taking into account attacks against the control layer. To ensure that the system is always up to current, it will be preferable to conduct real-time training in the future while investigating spoofed DDoS attacks with actual SDN traffic. Another single-feature authentication and detection method would be to use the gradient of the number of packets within a monitor window. Future research will diversify the attacks used and evaluate the classification abilities of feature selection methods and ML-based models.

7 Conclusions

The SNORT startup establishes pre-defined ports and rules by default. Packages were targeted in the direction of the SDN controller after an efficient flood of DDoS hits from the penetration tool utilized. The guidelines can be created in SNORT under the local. rules file in the/ and so on/ SNORT/ rules directory, where alert types can be inserted based on traffic entering the service or applications. The RYU controller was utilized, which used TCP port 80 and could be attacked using a TCP-SYN Flood attack. As previously stated, with other DDoS attacks on controllers, an effective attack could bring the controller down. Rules area, there was an attempt of a DDoS attack utilizing numerous tools, and the snort console reveals the notifies on incoming traffic on SDN Controller over Port number 80 signals are created in which the possible DDoS attack from 192.168.0.180. ZEEK also contributed to improving the safety of the SDN network. Necessary policies were created to offer the appropriate logs required for assessment. ZEEK was doing behind-the-scenes while SORT was worrying about the attacks with alerts. ZEEK gave a more descriptive evaluation of the assault, from when it was simply an ICMP ping to when it became a TCP-SYN attack. ZEEK automatically develops folders that suit the suitable attack found. ZEEK produced weird log documents indicating that it had seen an attack or obtained a weird packet. ZEEK created a file called reporter, which gives the analysis of all the packets that went down as well as the percentages. This makes ZEEK a reputable spoofing or sniffing system for attaining a guaranteed network. Identifying the attack is the first and primary step before the network administrator can resolve the attack. In this research study, a DDoS detection system is carried out by using SNORT IDS (Intrusion Detection System) on the RYU controller. ZEEK was introduced to improve and examine the efficiency of the DDoS detection tool. There are numerous engineers presently working on the security of SDN. This research study will assist in offering a brand-new direction for engineers. The findings from the proposed model demonstrate that

while maintaining service availability for authorized users, we can severely restrict attacker capability. When evaluating cost–benefit analysis for attack mitigation, this is especially helpful. Future work will examine the unsupervised ML model, which carefully examines all unknown traffic. In the future, executing more IDS for the same TCP-SYN flooding or UDP flooding to include a DDoS attack, a DDoS prevention framework can be a study point and likewise have a real-life scenario to deal with.

Acknowledgements The authors thank the anonymous reviewers for the useful comments, which helped to improve the quality of the manuscript.

Funding The work of Agbotiname Lucky Imoize is supported by the Nigerian Petroleum Technology Development Fund (PTDF) and the German Academic Exchange Service (DAAD) through the Nigerian-German Postgraduate Program under Grant 57473408.

Data availability statement The data will be made available upon reasonable request by the corresponding author.

Declarations

Conflict of interest The authors declare no conflict of interest related to this work.

References

1. Al-Thaedan, A., Shakir, Z., Mjhool, A. Y., Alsabah, R., Al-Sabbagh, A., Salah, M., & Zec, J. (2023). Downlink throughput prediction using machine learning models on 4G-LTE networks. International Journal of Information Technology, 1–7.
2. Hong, S., Xu, L., Wang, H., & Gu, G. (2015, February). Poisoning network visibility in software-defined networks: New attacks and countermeasures. In Ndss (Vol. 15, pp. 8–11).
3. Wang, R., Jia, Z., & Ju, L. (2015, August). An entropy-based distributed DDoS detection mechanism in software-defined networking. In 2015 IEEE Trustcom/BigDataSE/ISPA (Vol. 1, pp. 310–317). IEEE.
4. Rawat, R., Chakrawarti, R. K., Raj, A., Mani, G., Chidambarathanu, K., & Bhardwaj, R. (2023). Association rule learning for threat analysis using traffic analysis and packet filtering approach. International Journal of Information Technology, 1–11.
5. Tonkal Ö, Polat H, Başaran E, Cömert Z, Kocaoğlu R (2021) Machine learning approach equipped with neighbourhood component analysis for ddos attack detection in software-defined networking. Electronics 10(11):1227
6. Biswas P, Samanta T (2021) Anomaly detection using ensemble random forest in wireless sensor network. Int J Inf Technol 13(5):2043–2052
7. Chin, T., Mountrouidou, X., Li, X., & Xiong, K. (2015, October). An SDN-supported collaborative approach for DDoS flooding detection and containment. In MILCOM 2015–2015 IEEE Military Communications Conference (pp. 659–664). IEEE.
8. Hnamte, V., & Hussain, J. (2023). An efficient DDoS attack detection mechanism in SDN environment. International Journal of Information Technology, 1–14.
9. Ahuja N, Singal G, Mukhopadhyay D, Kumar N (2021) Automated DDOS attack detection in software defined networking. J Netw Comput Appl 187:103108
10. Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015, February). Sphinx: detecting security attacks in software-defined networks. In Ndss (Vol. 15, pp. 8–11).
11. Arunkumar M, Kumar KA (2023) GOSVM: Gannet optimization based support vector machine for malicious attack detection in cloud environment. Int J Inf Technol 15(3):1653–1660
12. Valdovinos, I. A., Pérez-Díaz, J. A., Choo, K. K. R., & Botero, J. F. (2021). Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. Journal of Network and Computer Applications, 187(May). https://doi.org/10.1016/j.jnca.2021.103093
13. Badotra S, Panda SN (2021) SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking. Clust Comput 24(1):501–513. https://doi.org/10.1007/s10586-020-03133-y
14. Ahuja, N., Singal, G., & Mukhopadhyay, D. (2020). DDOS attack SDN dataset. Mendeley Data, 1.
15. AbdulRaheem, M., Oladipo, I. D., González-Briones, A., Awotunde, J. B., Tomori, A. R., & Jimoh, R. G. (2022). An efficient lightweight speck technique for edge-IoT-based smart healthcare systems. In 5G IoT and Edge Computing for Smart Healthcare (pp. 139–162). Academic Press.
16. Shao, E. (2019). Encoding IP address as a feature for network intrusion detection (Doctoral dissertation, Purdue University Graduate School).
17. Chin, T., Xiong, K., & Rahouti, M. (2018). SDN-based kernel modular countermeasure for intrusion detection. In Security and Privacy in Communication Networks: 13th International Conference, SecureComm 2017, Niagara Falls, ON, Canada, October 22–25, 2017, Proceedings 13 (pp. 270–290). Springer International Publishing.
18. Scott-Hayward S, Natarajan S, Sezer S (2015) A survey of security in software defined networks. IEEE Communications Surveys & Tutorials 18(1):623–654
19. Li, Y., Serrano, M., Chin, T., Xiong, K., & Lin, J. (2019, July). A Software-defined Networking-based Detection and Mitigation Approach against KRACK. In ICETE (2) (pp. 244–251).
20. Chin T, Xiong K, Hu C (2018) Phishlimiter: A phishing detection and mitigation approach using software-defined networking. IEEE Access 6:42516–42531
21. Curtis, A. R., Kim, W., & Yalagandula, P. (2011, April). Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In 2011 Proceedings IEEE INFOCOM (pp. 1629–1637). IEEE.
22. Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013, November). Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 413–424).
23. Tian, Y., Tran, V., & Kuerban, M. (2019, January). DoS attack mitigation strategies on SDN controller. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0701–0707). IEEE.
24. Wu, G., Li, Z., & Yao, L. (2018, December). DoS mitigation mechanism based on non-cooperative repeated game for SDN. In 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS) (pp. 612–619). IEEE.
25. Shang, G., Zhe, P., Bin, X., Aiqun, H., & Kui, R. (2017, May). FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks. In IEEE INFOCOM 2017-IEEE Conference on Computer Communications (pp. 1–9). IEEE.

26. Wu P, Yao L, Lin C, Wu G, Obaidat MS (2018) Fmd: A DoS mitigation scheme based on flow migration in software-defined networking. Int J Commun Syst 31(9):e3543
27. Wang T, Chen H, Qi C (2018) Mindos: A priority-based SDN safe-guard architecture for DoS attacks. IEICE Trans Inf Syst 101(10):2458–2464
28. Bharathi, N. A., Vetriselvi, V., & Parthasarathi, R. (2019). Mitigation of DoS in SDN using path randomization. In International Conference on Computer Networks and Communication Technologies: ICCNCT 2018 (pp. 229–239). Springer Singapore.
29. Wang, S., Chavez, K. G., & Kandeepan, S. (2017, May). SECO: SDN sEcure COntroller algorithm for detecting and defending denial of service attacks. In 2017 5th International Conference on Information and Communication Technology (ICoIC7) (pp. 1–6). IEEE.
30. Wang, S., Chandrasekharan, S., Gomez, K., Kandeepan, S., Al-Hourani, A., Asghar, M. R., ... & Zanna, P. (2018, April). SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks. In NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium (pp. 1–5). IEEE.
31. Zheng J, Li Q, Gu G, Cao J, Yau DK, Wu J (2018) Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis. IEEE Trans Inf Forensics Secur 13(7):1838–1853
32. Alshamrani, A., Chowdhary, A., Pisharody, S., Lu, D., & Huang, D. (2017, November). A defense system for defeating DDoS attacks in SDN based networks. In Proceedings of the 15th ACM international symposium on mobility management and wireless access (pp. 83–92).
33. Latah, M., & Toker, L. (2018). A novel intelligent approach for detecting DoS flooding attacks in software-defined networks. International Journal of Advances in Intelligent Informatics.
34. Li C, Wu Y, Yuan X, Sun Z, Wang W, Li X, Gong L (2018) Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN. Int J Commun Syst 31(5):e3497
35. Ye, J., Cheng, X., Zhu, J., Feng, L., & Song, L. (2018). A DDoS attack detection method based on SVM in software defined network. Security and Communication Networks, 2018.
36. Li, X., Yuan, D., Hu, H., Ran, J., & Li, S. (2015, December). DDoS detection in SDN switches using support vector machine classifier. In 2015 Joint International Mechanical, Electronic and Information Technology Conference (JIMET-15) (pp. 344–348). Atlantis Press.
37. Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., & Yang, B. (2016, November). Predicting network attack patterns in SDN using machine learning approach. In 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (pp. 167–172). IEEE.
38. Krishnan P, Duttagupta S, Achuthan K (2020) SDN/NFV security framework for fog-to-things computing infrastructure. Software - Practice and Experience 50(5):757–800. https://doi.org/10.1002/spe.2761
39. Shin, Seungwon. (2013). AVANT-GUARD : Scalable and Vigilant Switch Flow Management in Software-Defined Networks Categories and Subject Descriptors. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 413–424.
40. Jain, G., & Anubha. (2021). Application of SNORT and Wireshark in Network Traffic Analysis. IOP Conference Series: Materials Science and Engineering, 1119(1), 012007. https://doi.org/10.1088/1757-899x/1119/1/012007
41. Tan, H. C., Mohanraj, V., Chen, B., Mashima, D., Nan, S. K. S., & Yang, A. (2021, October). An iec 61850 mms traffic parser for customizable and efficient intrusion detection. In *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (pp. 194–200). IEEE.
42. Barbour, G., McDonald, A., & Mkuzangwe, N. (2021, June). Evasion of Port Scan Detection in Zeek and Snort and its Mitigation. In *ECCWS 2021 20th European Conference on Cyber Warfare and Security* (p. 25). Academic Conferences Inter Ltd.
43. Juarez, L. (2021). NIDS on a Budget. University of Hawai'i West O'ahu
44. Jankowski, D., & Amanowicz, M. (2016). On efficiency of selected machine learning algorithms for intrusion detection in software defined networks. *International Journal of Electronics and Telecommunications*, *62*(3).
45. Mowla NI, Doh I, Chae K (2018) CSDSM: Cognitive switch-based DDoS sensing and mitigation in SDN-driven CDNi word. Comput Sci Inf Syst 15(1):163–185
46. Polat H, Polat O, Cetin A (2020) Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models. Sustainability 12(3):1035