IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 March 2023, accepted 18 March 2023, date of publication 22 March 2023, date of current version 27 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3260256

RESEARCH ARTICLE

FMDADM: A Multi-Layer DDoS Attack Detection and Mitigation Framework Using Machine Learning for Stateful SDN-Based IoT Networks

WALID I. KHEDR, AMEER E. GOUDA, AND EHAB R. MOHAMED
Department of Information Technology, Zagazig University, Zagazig 44519, Egypt
Corresponding author: Ameer E. Gouda (amir.gouda@zu.edu.eg)

ABSTRACT The absence of standards and the diverse nature of the Internet of Things (IoT) have made security and privacy concerns more acute. Attacks such as distributed denial of service (DDoS) are becoming increasingly widespread in IoT, and the need for ways to stop them is growing. The use of newly formed Software-Defined Networking (SDN) significantly lowers the computational burden on IoT network nodes and makes it possible to perform more security measurements. This paper proposes an SDN-based, four-module DDoS attack detection and mitigation framework for IoT networks called FMDADM. The proposed FMDADM framework comprises four main modules and five-tier architecture. The first module implements an early detection process based on the average drop rate (ADR) principle using a 32-packet window size. The second module uses a novel double-check mapping function (DCMF), that aids in earlier attack detection at the data plane level. The third module is an ML-based detection application comprising four phases: data preprocessing, feature extraction, training and testing, and classification. This module detects DDoS attacks using only seven features: two selected and five newly computed features. The last module introduces an attack mitigation process. We applied the proposed framework to three test cases: one single-node attack test case and two multi-node attack test cases, all with real IoT traffic generated and deployed in Mininet-IoT. The proposed FMDADM framework efficiently detects DDoS attacks at high and low rates, can discriminate between attack traffic and flash crowds, and protects both local and remote IoT nodes by preventing infection from propagating to the ISP level. The FMDADM outperformed most existing cutting-edge approaches across ten different evaluation criteria. According to the experimental results, FMDADM achieved the following accuracy, precision, F-measure, recall, specificity, negative predictive value, false positive rate, false detection rate, false negative rate, and average detection time benchmarks:- 99.79%, 99.43%, 99.77%, 99.79%, 99.95%, 00.21%, 00.91%, 00.23%, and 2.64 μs, respectively.

INDEX TERMS DDoS, detection, IoT, machine learning, mitigation, network security, SDN, SD-IoT.

I. INTRODUCTION

The diffusion and integration of the Internet of Things (IoT) into several critical industries, including transportation, healthcare, energy, and agriculture, has become undeniable. IoT is a revolutionary technology that links numerous nodes via wireless technologies to automatically send and receive data. The IoT systems have transformed conventional systems into intelligent, economical, and scalable systems. The heterogeneous nature of IoT networks is a major challenge [1]. This is because various IoT applications have distinct network requirements that must be met to operate the system optimally [2]. In addition to the benefits of IoT services, we have recently noticed their negative consequences on network security. According to Sarker et al. [3], IoT nodes may be vulnerable to malware outbreaks that spread surreptitiously among unprotected nodes to form an enormous number of IoT botnets. An IoT network's nodes are vulnerable to various attacks that attempt to obstruct the services offered by the IoT or control the entire network. Among these attack types, distributed denial of service (DDoS) attacks can be the IoT system's most challenging security risk [4]. When a

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

DDoS attack is launched against an IoT network, the network immediately begins to allocate resources to handle these requests. Any new requests would be rejected even if they came from a legitimate user, which would prevent the IoT network from providing its services as intended [5]. The most straightforward mitigation strategy is to develop cutting-edge security solutions that safeguard the IoT networks. The main obstacles in deploying any attack detection strategy are the limited power, processing, network bandwidth, and storage capacity resources over different IoT layers [6]. To avoid security breaches, it is critical to protect each layer of the IoT environment. IoT may be attacked at three different layers: the device layer, where data are gathered; the network layer, where data are transported for processing; and the cloud layer, where data are saved [7]. The proposed framework focuses on IoT network layer security.

In recent years, a range of technologies, systems, and approaches have been proposed for solving security issues in IoT. Software-defined networking (SDN) and machine learning (ML) technologies have attracted the interest of researchers to address different IoT security concerns [8]. On the one hand, a new technique known as ''Stateful SDN'' has expanded the basic functions of OpenFlow, the most widely used protocol for communication between data and control planes [9], by adding the ability to apply multiple match-action rules depending on the distinct states detected in the switch's SDN flow tables [10], [11], [12]. This feature gives the switch the ability to respond to events at the packet-level. The switch can take appropriate action if the results of the packet analysis agree with the switch rules listed in the flow tables [13].

Meanwhile, academics have developed many ML-based approaches and strategies for identifying DDoS attacks in SD-IoT. ML-based approaches have yielded good results in detecting DDoS attacks in SD-IoT networks [14]. Building a learned parameter model used to accurately forecast attacks requires training the system on both normal and attack behaviors. A substantial amount of data is produced by an IoT network. Choosing the most pertinent features of a dataset for model training and testing remains a challenging task. Using a large number of features in ML models increases both the cost and time complexity [15]. However, the inclusion of unrelated features renders the model less effective in detecting attacks. The construction of an effective ML-based DDoS detection model depends on the packet feature engineering approach, which is crucial [16], [17]. Considering these factors, we propose a new framework called FMDADM, which comprises four phases: data preprocessing, feature extraction, training and testing, and classification. The proposed FMDADM uses only five new computed features to detect an attack. By using fewer features, attacks can be detected more quickly. A variety of ML models for traffic classification, including Support Vector Machine (SVM), k-Nearest Neighbor (kNN), Gaussian Naive Bayes (GNB), Binomial Logistic Regression (BLR), Decision Tree (DT), and Random Forest (RF) classifiers, are used to construct the proposed detection model.

The research questions of our proposed framework can be summarized as follows:

1. Which ML-based DDoS attack detection model is the most suitable for stateful SDN-enabled IoT networks?
2. Which ML-based DDoS attack detection algorithm is best for multi-node attacks in SDN-enabled IoT networks?
3. How can a remote server be protected from botnet attacks over an IoT LAN network?

Whereas the main contributions are as follows:

1. The proposed FMDADM framework employs feature engineering to identify DDoS attacks using only five new computed features. The model can successfully overcome the over-fitting problem and provides a good fit as a result.
2. The second detection module, which uses a novel proposed mapping function called DCMF, provides two crucial features:- (a) detecting the attack at the data plane level before overwhelming the controller, and (b) discriminating between attack traffic and flash crowds. As a result, the controller has an extra layer of protection against the attack.
3. A small 32-packet window size is used for feature extraction in the third detection module. The three detection modules resulted in a reduction in the amount of time needed for training, testing, and detection.
4. FMDADM effectively detects DDoS in multi-node attack scenarios. This is a crucial area of strength for the proposed framework because it is generally known that conventional defenses fall short in the face of these attack scenarios.
5. FMDADM protects both local and remote IoT nodes by preventing infections from spreading to the ISP level. By protecting the controller and remote nodes in this form, we can stop DDoS attacks before they reach the Internet.
6. FMDADM uses actual IoT traffic features to build the detection model. Most literature-based studies deal with either simulated traffic or network traffic that is not extracted from actual IoT networks.
7. According to the experimental results, FMDADM outperformed most of the existing cutting-edge approaches across ten different evaluation criteria.

The remainder of this paper is organized as follows. Section II provides a summary of related research on the subject. Section III elaborates the proposed detection framework. The experimental results of the proposed framework are presented in Section IV. Section V provides a summary of the original contributions and discusses future work.

II. RELATED WORKS

Recently, relevant AI-based studies have been presented for DDoS attack detection in Software-Defined IoT (SD-IoT) networks, where SDN was used to improve the security

aspects of IoT networks. A DDoS detection solution for an SDN-based IoT network, called LEDEM, was suggested in [18]. The main issue with this model is its lack of adaptability, as LEDEM uses only one classification method and is incapable of dealing with various types of DDoS attacks. Yin et al. [19] proposed a broad architecture for the SD-IoT. The proposed SD-IoT architecture analyzes IoT network traffic and detects DDoS attacks based on the network attributes. The SD-IoT is further constrained by insufficient ML-based categorization algorithms. Xie et al. [20] employed traffic-flow patterns to identify DDoS attacks. This solution shows efficient DDoS information detection with a comparably low overhead compared with other approaches. However, this solution falls short when dealing with heavy network traffic, necessitating the implementation of a more sophisticated security solution. In [21], a new SVM-based security mechanism for IoT networks was proposed. Their model uses learning algorithms for both observing and reacting agents. Their proposed method achieved a general accuracy rate of 99.71% for anomaly identification. The authors of [22] suggested a feed-forward neural network model for attack detection in IIoT networks. The proposed model performed well in terms of accuracy; however, the dataset was not designed for the IIoT domain.

Ullah and Mahmoud [23] developed a new anomaly-based detection system for IoT networks. They devised a multi-class classification technique using a convolutional neural network (CNN) algorithm. This classification technique was admirably performed. However, ML approaches are preferred in intrusion detection systems (IDSs) for implementing highly secure capabilities [24]. The authors of [24] examined several ML models to conduct both binary and multiclass classification. They concluded that, compared with other classifiers, the XGBoost technique produced higher performance outcomes. Another CNN-based DDoS attack detection system for IoT networks, which restricts attacks at the source end, was proposed by the authors of [25]. They evaluated the proposed CNN using the freely accessible dataset CIC-DDoS2019 [26]. Two test cases were used to obtain the performance results; however, this dataset was insufficient for analyzing the behavior of IoT network traffic. Using a recurrent neural network (RNN), Yousuf and Mir [27] proposed an algorithm called DALCNN. DALCNN employs OpenDayLight (ODL) as a suitable SDN controller to address the problem of identifying DDoS attacks in IoT. The gap in DALCNN is that the RNN algorithm was trained using the NSL-KDD dataset, which is unfortunately inadequate for IoT network traffic characteristics. Another detection mechanism for abnormalities in IoT environments was proposed by Alanazi and Aljuhani [28]. The proposed mechanism uses several ML techniques for feature selection and an ensemble learning approach for traffic classification. Numerous constraints were considered in this study. For instance, detection accuracy may be affected by employing custom datasets instead of real-time IIoT traffic. Another problem is that obsolete datasets are restricted to particular types of cyber-attacks and cannot recognize contemporary attack scenarios.

Another RNN-based deep learning solution termed ''Deep Defense'' was proposed by Yuan et al. [29] for detecting DDoS attacks in IoT. This solution employs a series of consecutive network packets to extract low-level features to discriminate between normal and attack packets. However, the authors still need to test their model under numerous real-time scenarios. A Deep Neural Network (DNN) approach was proposed in [30] to identify DDoS attack in SDN scenarios. The tests results show that the Deep IDS system is a workable solution with a low network load. The proposed approach does not affect the functionality of the POX controller. Furthermore, the authors needed to improve the model to obtain better detection rates with low false-alarm rates across multiple OpenFlow Controllers. Zhang et al. [31] presented a technique for detecting low rate (LR) DoS attacks based on the Power Spectral Density (PSD). In [31], an SVM model was used to extract features from the KDD99 dataset. The PSD entropy limit values were separately set for the normal and attack groups. There was a trade-off between the accuracy and detection rates in the proposed IDS, which was not AI-based. A hybrid model developed by ElSayed et al. [32] utilizes a CNN and RF to determine whether the incoming flow to an SDN is normal. Empirical analysis revealed that the model performed well on a variety of publicly accessible datasets. However, the proposed model lacks support for feature reduction. In this context, Silveira et al. [33] evaluated the effectiveness of their proposed IDS in identifying LR DoS attacks in an SD-IoT system using a freely accessible CIC DoS 2017 dataset [34]. Unfortunately, this dataset is not ideal choice to employ because it has no relation to IoT network traffic features. Tang et al. proposed a similar LR DoS detection technique employing AdaBoost [35]. The authors selected a group of 28 traffic flow features to train and test the classifier effectiveness. The last three LR DoS detection techniques [32], [33], and [35] presented above are signature-based and may not accurately identify new attacks. In another attempt to provide a solution for identifying LR DoS attacks in SD-IoT, the authors of [36] introduced a FeedForward-Convolutional Neural Network (FFCNN), an AI-based anomaly detection method. FFCNN detects LR DoS attacks by combining a FeedForward Neural Network (FFNN) with a CNN. The drawback of this method is that it uses the CIC DoS 2017 dataset, which was not primarily extracted from real or simulated IoT networks. Motivated by the research gaps identified in the literature, we propose the FMDADM framework, which we explain in detail in the following sections.

III. PROPOSED FRAMEWORK

The Internet connects many ISP networks, which in turn connect multiple LANs made up of heterogeneous nodes. These LANs are growing rapidly in size and volume, and IoT nodes may be vulnerable to malware outbreaks, making them potent sources of DDoS attacks. In the worst case, infected IoT

nodes (botnets) from multiple compromised ISP LANs may attack a remote server, causing severe resource scarcity on both the controller and the server. The proposed framework aims to prevent local DDoS attacks generated by IoT Botnets inside infected LANs from spreading to the ISP level. In this way, we can safeguard both the controller and remote nodes and block DDoS attacks from reaching the Internet. Figure 1 shows the five-layer architecture of the proposed FMDADM framework.

The proposed framework consisted of four modules: three detection modules and one mitigation module. The first module uses a small window size of 32 packets. This window size was then employed in the third detection module to achieve an earlier and more efficient attack detection. The second detection module presents a new mapping function to help detect DDoS attacks at the data plane level. The third module provides an ML-based detection application that is implemented and deployed at the controller level. The last module is a mitigation technique that operates at both switch and controller levels. Most related research uses a conventional, stateless approach to design detection and mitigation processes. The switch scans its flow table to match the incoming packets. If no match was found, the packets were treated as new and routed to the controller for processing. However, this approach lacks scalability and efficiency. In this paper, we make use of the aforementioned ''Stateful SDN,'' where the switches are given stateful packet analysis privileges by the SDN network. Therefore, when new packets arrive at the network, the switch considers the packets that have already been received in addition to the new packet features. The following subsections describe the proposed framework in detail.

A. THE FIRST DETECTION MODULE

The first detection module uses a 32-packet window size for early attack detection. Several factors must be considered when selecting the appropriate window size. One of these factors is the constrained number of fresh connections that may be established for each node in an IoT network. The number of nodes and switches of each controller within the network should also be considered. The third factor is that an attack within a frame of 32 packets is recognized more quickly than that of 50, 100, or even 500 packets [37]. The choice of 32-packet window is based on two principles: the average window entropy drop rate (AWEDR) and the test of significance (ToS). These two principles demonstrate that a 32-packet window is the best window size for detecting DDoS attacks. These principles are discussed in the following subsections.

1) WINDOW SIZE SELECTION BASED ON AWEDR

The window entropy drop rate (WEDR) is the rate at which the entropy value decreases in each window. There is a greater chance that an attack will be underway if the pace of the decrease is faster. For seven different attack rates from 20% to 80%, a 32-packet window size achieved the first acceptable average drop rate among the other ten window sizes. Equation (1) shows the calculation of the average window entropy drop rate (AWEDR) for a window size of n packets.

$$AWEDR = \frac{AWNE - AWAE}{AWNE} \times 100 \tag{1}$$

AWNE and AWAE represent the average window normal entropy and the average window attack entropy, respectively. Table 1 lists the AWEDR for the 11 window sizes when the attack rate was 20%. WS, NE, AE, (NE-AE), AP, and NP refer to the window size, normal entropy, attack entropy, difference between normal and attack entropy, number of packets in the attack case, and number of packets in the normal case, respectively. The AWEDR was calculated based on 840 experimental tests (120 runs for each attack rate) using two test cases: a single-node attack (attack packets targeting a single node) and a multi-node attack (attack packets targeting multiple nodes), both using real traffic generated in Mininet-IoT. Figure 2 shows the entropy drop rate in the single-node attack test case. The entropy drop rate increases concurrently with an increase in the attack rate. As shown in table 1, the number of attack packets targeting a single host for a 32-packet window size is seven times higher than the normal rate, with an average drop rate of 14.13%. The average drop rate in the 20, 24, and 28 window sizes was less than 10%, which was insufficient to pose a real threat to the controller. In the following subsection, we use a significance test to test this hypothesis. In table 1, the A_E threshold values are computed by running a series of experiments to determine how the attack affected the entropy value. Attack rates varying from 10% to 80% were launched and analyzed. In 120 experiments, attack rates of 10% and 15% yielded ADR of 7.08% and 9.32%, respectively.

These drop rates are insignificant and should not be considered when calculating the threshold. The attack rate of 20% was the lowest, which recorded an approved entropy drop rate value of 14.64% between the normal and attack entropies. Consequently, the A_E threshold was set to identify any attack that consumes 20% or more of the total network bandwidth. Table 2 presents the threshold value selection procedure. In table 2, ANTEPC, ANTEMC, ANTEDIFF, AATEPC, AATEMC, SD, and CI denote the maximum average normal

TABLE 1. Average window drop rate for different window sizes.

WS	N_E	A_E	N_E-A_E	A_P	N_P	AWEDR	Z-Score
20	1.501	1.377	0.124	4	1	08.20%	1.001
24	1.501	1.367	0.134	5	1	08.87%	1.023
28	1.501	1.363	0.138	6	1	09.17%	1.073
32	1.501	1.288	0.213	7	1	14.13 %	1.203
36	1.501	1.286	0.215	7	1	14.26%	1.225
40	1.501	1.285	0.216	8	1	14.37%	1.234
44	1.501	1.283	0.218	9	1	14.49%	1.262
48	1.501	1.282	0.219	10	1	14.57%	1.293
50	1.501	1.278	0.223	10	1	14.90%	1.301
100	1.501	1.271	0.230	20	1	15.30%	1.339
500	1.501	1.116	0.385	100	1	25.63%	1.389

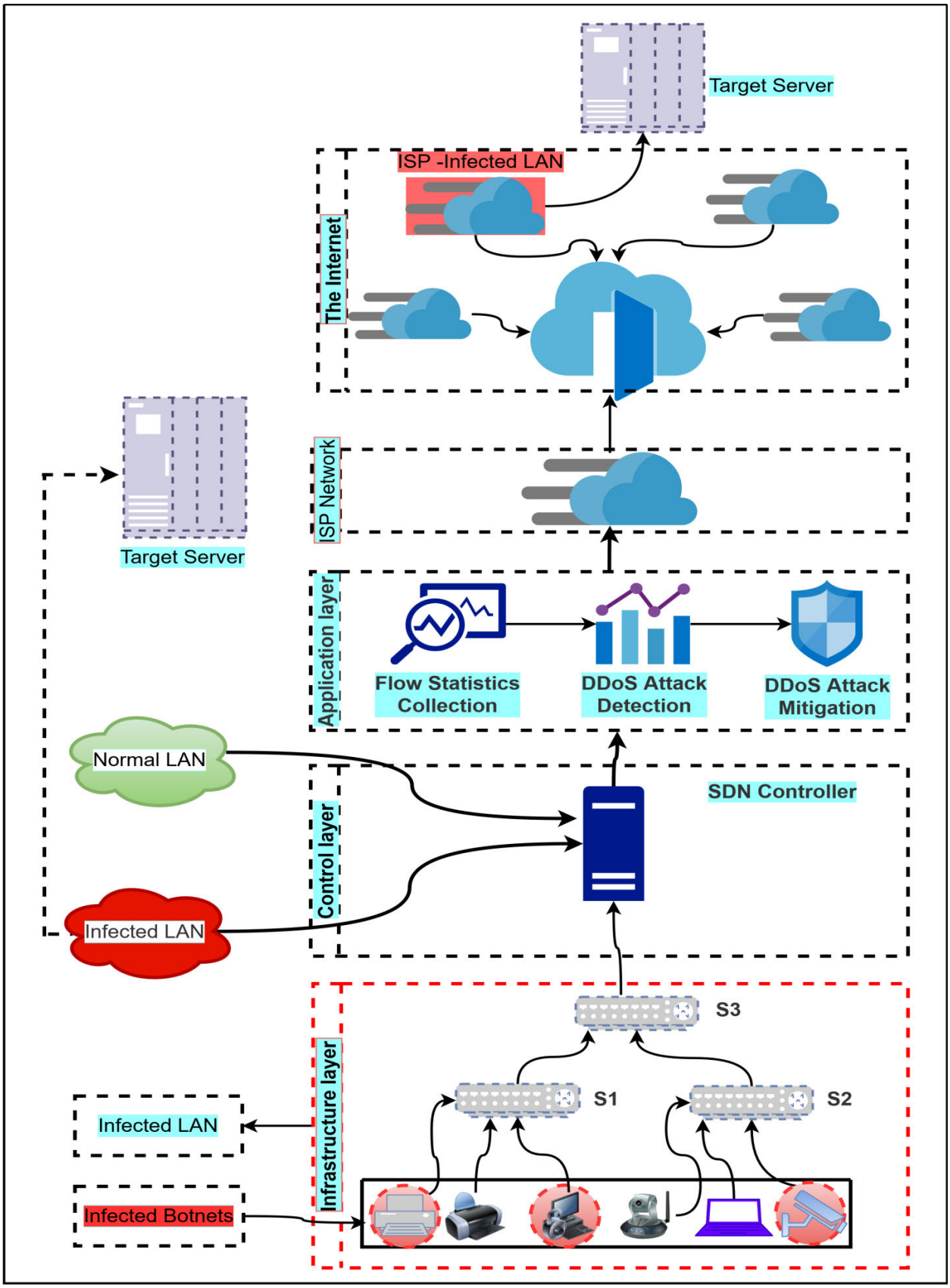

FIGURE 1. The general architecture of the proposed FMDADM framework.

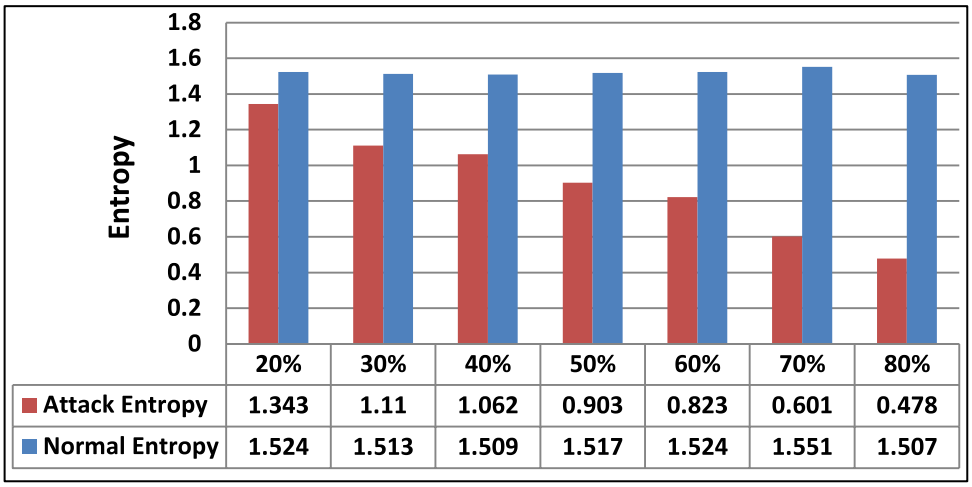

FIGURE 2. AEDR for different attack rates in a single-node attack test case.

TABLE 2. Entropy threshold value calculation.

Parameter	Normal State	Attack State
Entropy	1.564	1.343
SD	0.007	0.011
CI	± 0.0037	± 0.0049
ANTEPC	1.5675	-
ANTEMC	1.5605	-
AATEPC	-	1.3477
AATEMC	-	1.3383
ANTEDIFF	0.2128	
ADR	**14.64 %**	

entropy, the minimum average normal entropy, the difference between ANTEMC and ANTEPC, the maximum average attack entropy, the minimum average attack entropy, standard deviation, and confidence interval, respectively. Three steps were executed to obtain the A_E threshold as follows:

1. The least normal traffic entropy (LNTE) was calculated by subtracting CI from ANTEMC.
2. The highest attack traffic entropy (HATE) was calculated by adding CI to AATEPC.
3. Calculated the difference between HATE and LNTE.

Although the calculations stated above imply that the threshold may be worth 1.32, experiments show that this value is not optimal. After repeating the simulation 120 times, we found that a threshold value of 1.28 produced much fewer false positives (FP) and false negatives (FN). Moreover, it provides a clear-cut and detects any outright attack that dominates 20% or more of the entire traffic. It is worth noting that SDN can be adapted. In other words, the threshold value can be changed at any moment based on the packet tracking and monitoring of the controller. This threshold change is intended to limit the number of FP and FN, based on different network conditions. Consequently, in the third detection module, we used an adaptive entropy threshold as a computed feature. This threshold varies based on the attack situation to enhance the detection accuracy.

2) WINDOW SIZE SELECTION BASED ON TOS

The test of significance (ToS) determines the validity of a hypothesis by comparing the two averages from different population groups. First, we computed the two average entropy values under normal and attack conditions. Then, we computed the Z-Score to test the significance of the difference between the two conditions to determine whether the difference was acceptable. This test was performed for different window sizes to determine the optimal window size. The Z-Score in (2) was computed as follows:

$$Z - Score = \frac{|N_E - A_E|}{\sqrt{\frac{\sigma^2 n}{n}} + \sqrt{\frac{\sigma^2 a}{a}}} \tag{2}$$

where (σ_n, σ_a) are the normal and attack entropy standard deviations, respectively; and (n, a) is the number of samples. We fixed $(n = a = 7)$, where we had seven attack rates varying from 20% to 80%. A 32-packet window yielded the least acceptable variation in findings. Consequently, we selected it as the optimal detection window size and employed it to calculate the features for the third module of the proposed framework.

B. THE SECOND DETECTION MODULE

The second module proposes a new double-check mapping function called DCMF, which is used to provide early attack detection at the switch level. The proposed DCMF uses state monitoring (OpenState extension) to offer Stateful SDN functionalities across an Open Virtual Switch (OVS). The DCMF has two state tables: State Table 1, which contains the source IP address entries, and State Table 2, which contains the source MAC address entries. The DCMF function verifies that each source MAC address is associated with only one source IP address during a given time frame. DCMF uses

one-to-one mapping to perform the detection process. The initial state values for the entries in Tables 1 and 2 are 0. If the state value for any state table entry is more than one, it signifies that either the source IP or the source MAC is counterfeit, and an attack is detected. In this manner, the DCMF can prevent attack traffic from overwhelming the controller by blocking any traffic with forged source IP/MAC addresses at the data plane. Algorithm 1 shows the four FMDADM module algorithms including the DCMF function. The DCMF process flow is illustrated in figure 3. The only legitimate request is the final entry into the state tables. This function effectively distinguishes between attack traffic and flash crowds. The third detection module leverages DCMF as a calculated binary feature with values of 0 and 1. A DCMF value of 1 denotes attack traffic status, whereas a value of 0 denotes normal traffic.

C. THE THIRD DETECTION MODULE

The third detection module operates at the application layer, which is equivalent to the cloud layer of IoT architecture. The proposed module watches the switches at regular intervals of 1 s, and the information collected from the switches was used to feed the module. The third module consists of four phases: data preprocessing, feature extraction, training and testing, and classification, which are explained in the following subsections.

1) PREPROCESSING PHASE

Preparing raw data to be acceptable for a machine learning model is known as data preprocessing. The three most crucial steps in the data preprocessing phase are 1) missing values replacement, 2) encoding categorical data, and 3) feature scaling. The first step starts by computing the mean of the column or row containing any missing values, and placing the mean in place of the missing value. For this purpose, we used the *Scikit-learn* library in *Spyder* using *Anaconda*. In the second step, we used the *sklearn* library's *LabelEncoder* class to encode categorical data. This class converts variables into digits. In the final step, feature scaling is performed in two ways: standardization and normalization. We employed the standardization procedure in our proposed strategy to scale the features. We used *StandardScaler* class from the *sklearn.preprocessing* library to standardize the independent variables in the range of 0 to 1. The new scaled features are computed in (3) as follows:

$$Y = \frac{x - mean\,(x)}{d} \tag{3}$$

where Y is the new scaled value, x the original feature value, and d the standard deviation.

2) FEATURE EXTRACTION PHASE

Feature engineering is the process of selecting and transforming the most important features from the original data when building a machine learning predictive model. Feature engineering comprises two processes: feature selection and

Algorithm 1 The Proposed FMDADM Framework Algorithm

Inputs: T_IoT, Com_F, Tr_S, S_te1, S_te2, and TN
T_IoT ← IoT Network Traffic
Com_F ← Computed Features
A_IoT ← Attack IoT Traffic
N_IoT ← Normal IoT Traffic
IoT_S ← IoT Traffic State (Normal: 0, Attack: 1)
Pkt_n ← Packet Count
WS ← Window Size
Tr_S ← Training Set
TN ← Trees Number in Forest (F)
Tr_S $^{(i)}$ ← A single sample from Tr_S
Dst_IPN ← Destination IP Count
Dst_Add_Table ← Destination IP Address Table
A_E ← Attack Entropy Threshold
S_t ← State Table
S_te1 ← State Table 1 Entry
S_te2 ← State Table 2 Entry
Src_IP ← Source IP Address
Dst_IP ← Destination IP Address
Output: Classified IoT Traffic as Normal or Attack

Window Initialization
Pkt_n = 0, WS = 32, Dst_IPN = 0, and A_E Threshold = 1.28
CPWE Feature Calculation
for each (Packet_in):
  if (Dst_IP in Dst_Add_Table):
    Add Dst_IP to Dst_IPN
  else
    Add Dst_IP to Dst_Add_Table
  if (Pkt_n % WS = 0):
    Calculate CPWE
  else
    Pkt_n = Pkt_n + 1
end for
FMDADM Second Detection Module
for each (S_t):
  if (S_te1_value > 1):
    Src_IP → forged
    DCMF_State = 1
  else if (S_te2_value > 1):
    Src_MAC → forged
    DCMF_State = 1
  else
    DCMF_State = 0
    Send DCMF_State → Controller
end for
FMDADM Third Detection Module
Function RF(Tr_S, Com_F)
 for i ∈ (1,…,F) do:
  RTL_i ← Random Learn (Tr_S $^{(i)}$, Com_F)
  L ← L ∪ (RTL_i)
  if (IoT_S == 1):
    Attack Traffic
  else
    Normal Traffic
 end for
 return L
 end Function
FMDADM Mitigation Module
for each (T_IoT∈A_IoT):
  Send FlowMod → Switch
  Add new flow entry → Switch
  Set Drop_All rule → Switch
end for

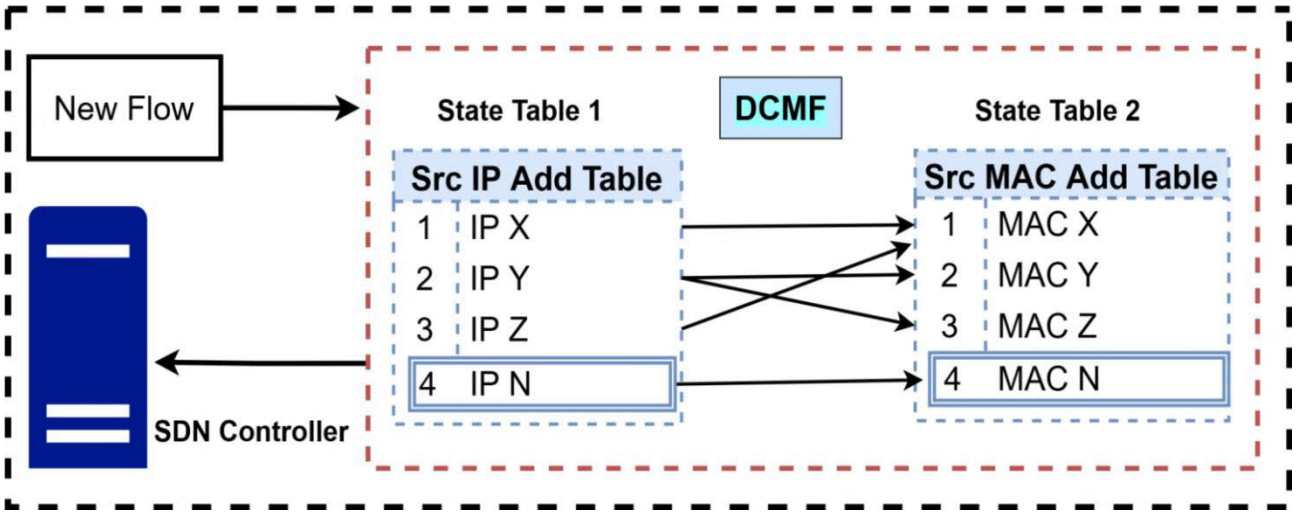

FIGURE 3. The proposed DCMF mapping function.

feature extraction. Whereas feature extraction generates new features, feature selection only selects a subset of the initial feature set. Both processes are required for the classification phase. Both the selected and extracted features are critical for shrinking large datasets, improving detection accuracy, and speeding up predictions. To achieve the early and accurate detection of attacks, we propose five new computed features. The five proposed new features are the computed particular window entropy (CPWE), computed packet rate feature (CPRF), received flow packets standard deviation (RFPSD), received flow bytes standard deviation (RFBSD), and computed flow entry rate (CFER). The features and their computations are discussed in the following subsections.

3) TRAINING AND TESTING PHASE

In the third proposed module, the dataset was split into training and testing sets. To split the dataset, we imported the *train_test_split()* function from *sklearn.model-selection* library into *Anaconda's Spyder IDE*. The model was then fitted to the training dataset using SVM, GNB, kNN, BLR, DT, and RF algorithms. After completing the training task, the trained classification models used the testing set to predict the final results of DDoS detection. In addition, a confusion matrix was set up to save and evaluate the results of the trained models. Finally, *joblib* from *sklearn.externals* is imported to save the trained classification models.

4) CLASSIFICATION PHASE

In this phase, the detection results from an actual inspected dataset are predicted using trained classification models. The computed features are merged to determine whether an interactive flow is attack or normal traffic. When a DDoS attack occurs, both CPRF and CFER features increase rapidly over time, whereas CPWE, RFPSD, and RFBSD feature values decrease. The following subsections describe the classification models used to create the proposed model.

5) ATTACK MITIGATION PHASE

The proposed mitigation module receives instructions from the two detection modules. When a malicious traffic flow is identified, the mitigation module protects the IoT nodes by adding extra high-priority flow rules to switch flow tables that match the attack packet rules. The switch receives a *FlowMod* message from the controller to add a new flow table entry and then drops all packets originating from the attack source. This reduces and mitigates the impact of the attack. Figure 5 illustrates this process and depicts a general flowchart of the proposed framework.

D. THE CLASSIFICATION MODELS

1) SUPPORT VECTOR MACHINE (SVM)

The SVM algorithm is a supervised machine learning two-class classification model that can be used to determine whether a particular traffic flow is regular or malicious. The SVM model executes the learning by supplying a sample set to each class. The algorithm defines a hyperplane that distinguishes between the two classes to perform the classification process. The two-dimensional line that splits the hyperplane into two classes is known as the classifier's decision line, where each class is assigned a unique location. After graphically depicting the points, the next task is to distinguish them using a line known as the decision boundary.

2) K-NEAREST NEIGHBOR (KNN)

The KNN algorithm is a supervised learning technique that is used to address classification problems. KNN assesses the possibility that input data points from a particular training set fall into any of the two classes, depending on which data points are nearest to it. To use the KNN, the data points are first converted into numerical values. Using Euclidean distance, the classifier calculates the difference between the data points. The k dataset with the least distance was used

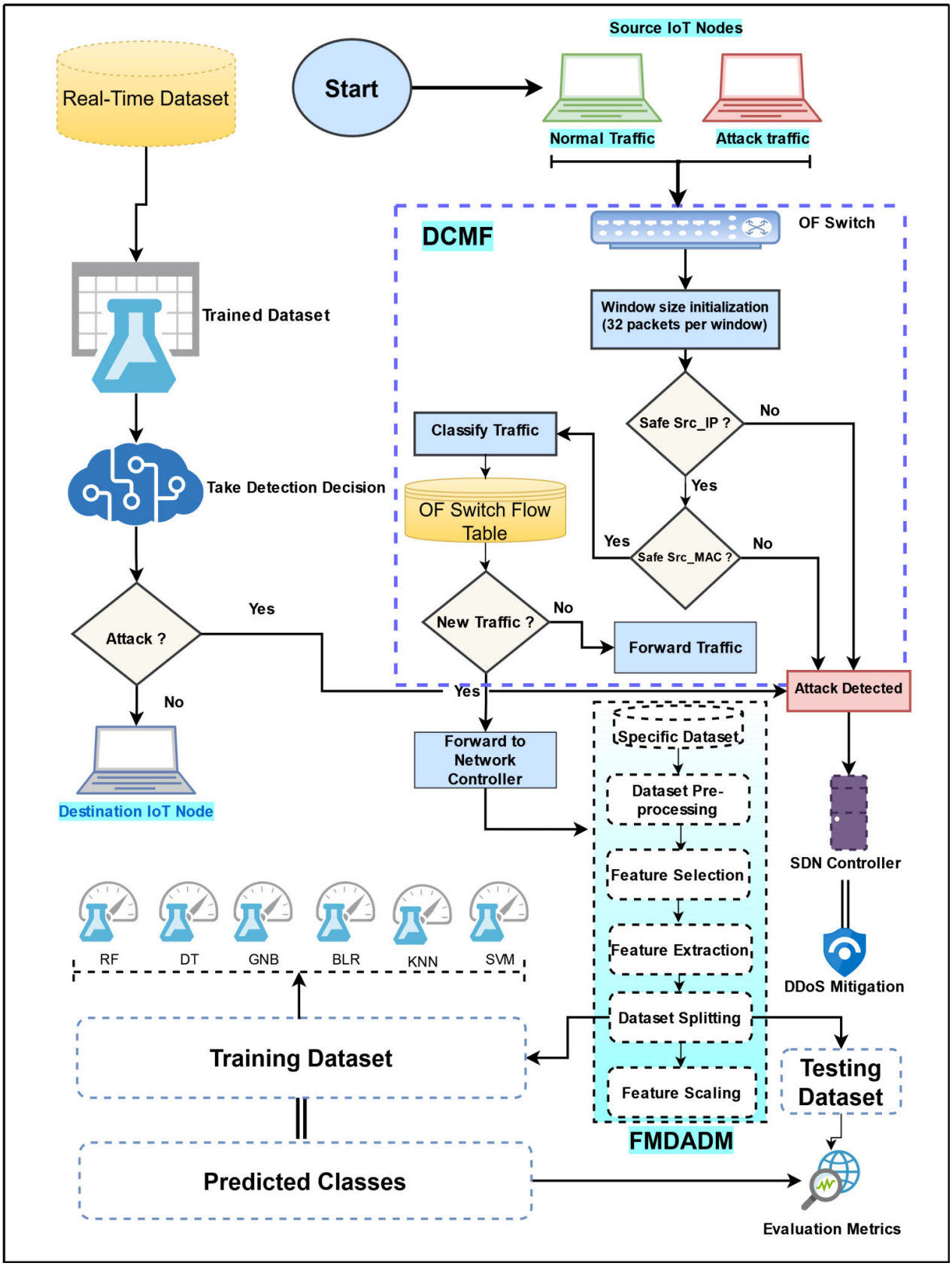

FIGURE 4. Flowchart of the proposed FMDADM framework.

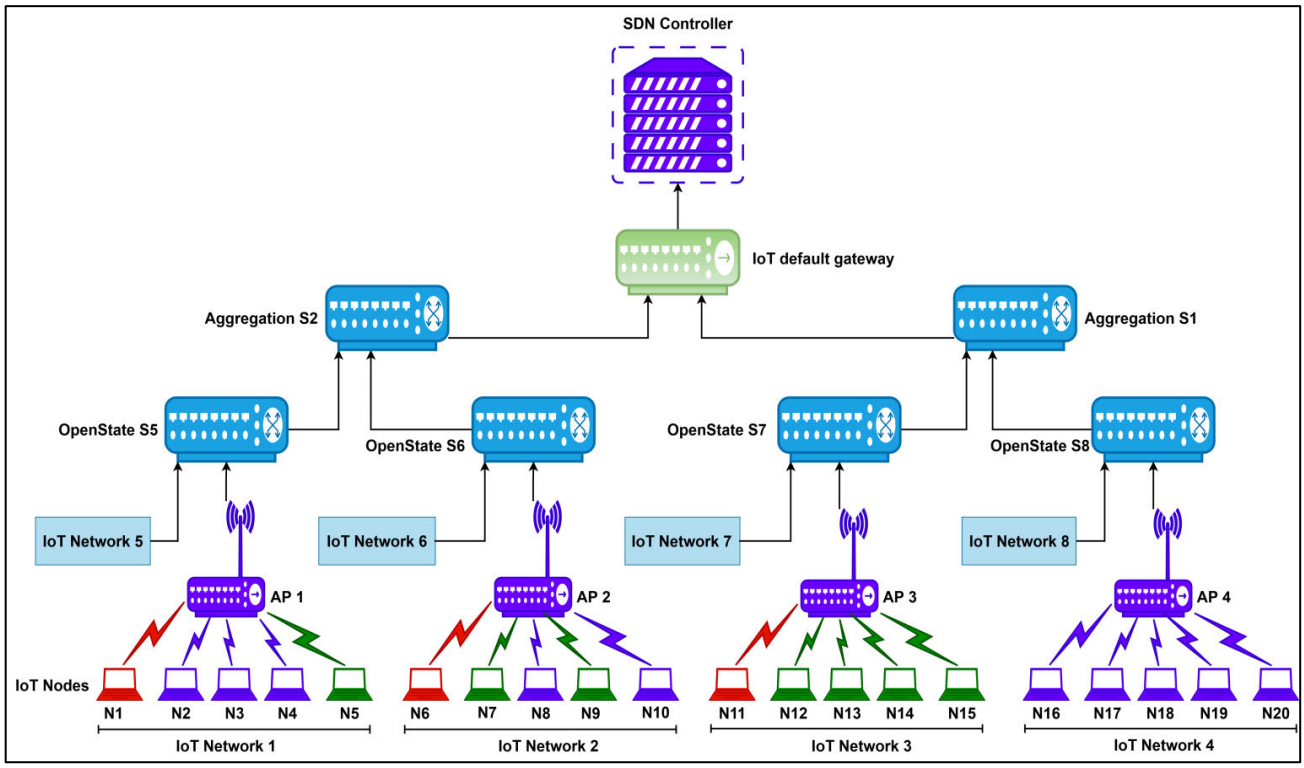

FIGURE 5. The general test architecture for the proposed detection framework.

as the classification set. Subsequently, an item is classified based on the majority of votes of its neighbors.

3) GAUSSIAN NAIVE BAYES (GNB)

Naive Bayes (NB) employs the independent variable comparison principle to ascertain the links between these independent variables. This is easy to build, because the algorithm does not evaluate the parameters. This enables it to operate on extremely large datasets. Moreover, the GNB model employs the probability of estimating continuous predictive features. Where Z→ [0, 1] is the result of employing F→ [F1, F2, F3,……, Fn] independent features.

4) BINOMIAL LOGISTIC REGRESSION (BLR)

One of the most well-known machine learning algorithms within the category of supervised learning is logistic regression (LR). LR uses a collection of independent variables to predict the category of the dependent variable. In BLR, the dependent variables can fall into one of two categories:- 0 or 1, normal or attack, etc. After preprocessing and splitting the dataset, we fitted the model to the training set by importing the *LogisticRegression* class from the *Sklearn* package. After the model was successfully trained, we forecasted the dependent variable class and determined the accuracy of the expected outcome by analyzing the confusion matrix.

5) DECISION TREE (DT)

The DT is a popular classification and prediction technique that uses supervised learning. The three primary components of DT are the root node, which symbolizes the decision; the branches that grow out of the root, which represent the various alternatives; and the leaf nodes, which represent the potential outcomes. These components form a tree when integrated. The primary problem is choosing the best property for both root nodes and sub-nodes.

An approach known as the attribute selection measure (ASM) was developed to address these issues. The (Information Gain) and the (Gini Index) are the two main ASM techniques. This study employs information gain by calculating the variation in entropy before and after splitting, and determines the defilement in class features.

6) RANDOM FOREST (RF)

The Random forest algorithm is an ensemble learning approach with applications in regression, classification, and other areas. During training, the RF builds many decision trees and produces a classification or regression for the mean prediction of all classes. A total of 1000 trees made up the RF classifier of the proposed model, and the minimum number of leaf nodes was 1.

E. THE COMPUTED FEATURES

1) COMPUTED PARTICULAR WINDOW ENTROPY (CPWE)

This feature refers to the entropy of a 32-packet window. To extract this feature, we created a function called *DstIpCollect.py* in Python and added it to the controller. This function collects destination IP addresses in a 32-packet window. Every new packet received by the controller was stored in an

array table. The entropy of the window was calculated when the array (*packet_count)* reached 32. The window entropy was computed under both normal and attack conditions.

2) COMPUTED PACKET RATE FEATURE (CPRF)

Packet rate is an extracted feature that indicates the number of data packets sent per second. The packet rate feature is calculated by dividing the packet per flow by the monitoring interval. The Packet per flow (PPF) is another computed feature that measures the total number of packets in a single flow at any given time. The packet rate feature is calculated using (4) as follows:

$$CPRF = \frac{FP_{Count}}{T * M_{Int}} \tag{4}$$

where FP_{Count} represents the total number of packets in a single flow, T represents the time frame used to compute the packets per flow and M_{Int} is the monitoring interval.

3) RECEIVED FLOW PACKETS STANDARD DEVIATION (RFPSD)

The packet per flow (PPF) feature is a calculated feature that equals the packet count (number of packets) for a single flow. The standard deviation of the packets per flow was proposed as a computed feature. This parameter is highly correlated with the likelihood of DDoS. An attacker broadcasts many small packets. This metric will be significantly decreased because these packets will have a far smaller standard deviation than the normal data packets. The RFPSD feature in (5) is computed for a 32-packet window size as follows:

$$RFPSD = \sqrt{\left(\left(\frac{1}{f}\right) * \sum_{i=1}^{f} (n - AvgP)\right)} \tag{5}$$

where f represents the number of live flows, n represents the number of packets in each live flow at a given time, and $AvgP$ represents the average number of packets across all the flows over a specified time.

4) RECEIVED FLOW BYTES STANDARD DEVIATION (RFBSD)

The byte per flow feature is a calculated feature that refers to the number of bytes transferred into a single flow. Similar to RFPSD, this metric has a significant link with the frequency of DDoS attacks, and its predicted value is smaller under attack conditions than that under regular traffic conditions. The RFBSD feature in (6) is computed as follows:

$$RFBSD = \sqrt{\left(\left(\frac{1}{f}\right) * \sum_{i=1}^{f} (n - AvgB)\right)} \tag{6}$$

where f represents the number of live flows, n represents the number of packets in each live flow at a given time, and $AvgB$ is the average number of bytes across all the flows over a specified time.

5) COMPUTED FLOW ENTRY RATE (CFER)

The flow entry rate (FER) is defined as the total number of flows that enter the OpenFlow switch during a given monitoring interval. When a DDoS attack occurs, the number of flow entries rapidly increases for a certain amount of time. Consequently, this feature is crucial in identifying DDoS attacks. The CFER feature in (7) is the computed FER for a 32-packet window size, and is computed as follows:

$$CFER = \frac{F_{Count}}{M_{Int}} \tag{7}$$

where F_{Count} *represents* the total number of flows during a given time and M_{Int} is the monitoring interval.

IV. EXPERIMENTAL RESULTS

The experiments were conducted on an Ubuntu server 20.04 LTS virtual machine running on an Intel core i5-1135G7 processor with 12 GB of RAM and a Microsoft Windows 10 host operating system. To develop and run the IoT test topologies, the following software tools were employed: VMware Workstation 12 Pro, Mininet-IoT, sFlow-test, and sFlow Mininet dashboard. Figure 6 shows the overall SDN-based IoT network topology architecture, which includes one POX controller, six switches, four access points (AP), one IoT default gateway, and 40 IoT nodes. We employed three different test cases to evaluate the performance of the proposed detection framework:- 1) single-node attack test case (SNATC), 2) two-node attack test case (TNATC), and 3) four-node attack test case (FNATC). In the first case, the attacker only targets one victim node. In the second case, the attack traffic is launched simultaneously against two target nodes. The last scenario involves sending attack packets to four target nodes at the same time.

Python scripts were developed using *Spyder* (a research development environment that is part of *Anaconda)* to create the proposed detection module. This module imports the *Numpy* and *Pandas* Python libraries to preprocess the dataset, *default-timer* and *date-time* modules for performing real-time IoT network topology experiments, and *matplotlib* library for visual analytics.

To assess the effectiveness of the ML classifiers, we employed the following measures: accuracy (ACC), precision or positive predictive value (PPV), F1-score (F1), recall or sensitivity or true positive rate (TPR), specificity (SPC) or true negative rate (TNR), negative predictive value (NPV), fall-out or false-positive rate (FPR), false discovery rate (FDR), missor false-negative rate (FNR), and average detection time (ADT). These assessment measures were computed using the confusion matrix, except for detection time, which was calculated using a function added to the code. The performance of ML classifiers is critical for an accurate DDoS attack detection process. These metrics have the following mathematical definitions:

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} \tag{8}$$

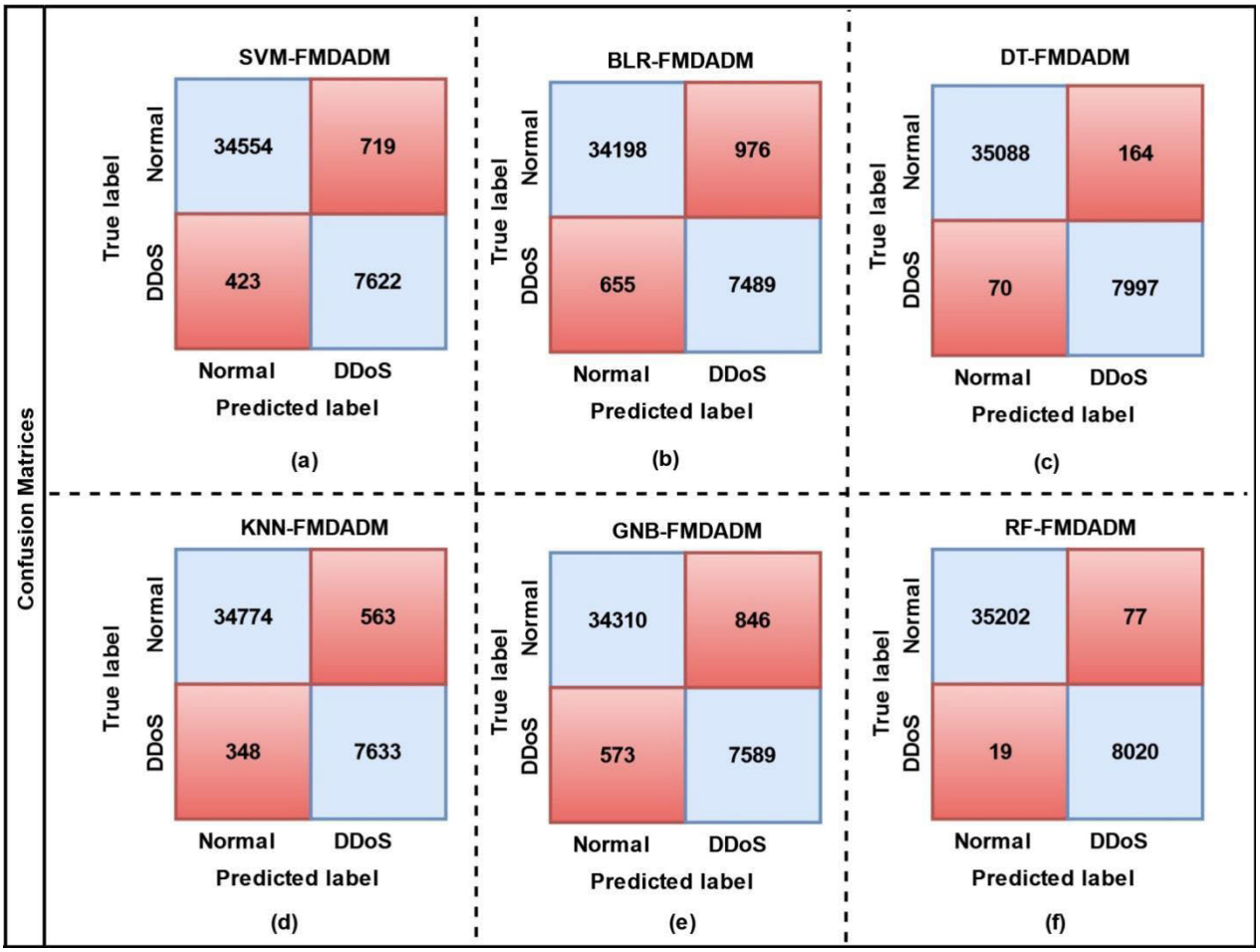

FIGURE 6. Confusion matrices of the tested classifiers based on the four-node attack test case.

$$PPV = \frac{TP}{FP + TP} \tag{9}$$

$$NPV = \frac{TN}{FN + TN} \tag{10}$$

$$TPR = \frac{TP}{FN + TP} \tag{11}$$

$$F1 = 2 * \frac{(PPV * TPR)}{(PPV + TPR)} \tag{12}$$

$$SPC = \frac{TN}{FP + TN} \tag{13}$$

$$FPR = \frac{FP}{TN + FP} \tag{14}$$

$$FDR = \frac{FP}{TP + FP} \tag{15}$$

$$FNR = \frac{FN}{TP + FN} \tag{16}$$

$$ADT = \frac{\sum_{i=1}^{n} DTR_x}{n} \tag{17}$$

where false positives (FP) are the number of real normal traffic cases that were incorrectly classified with the label "1" in the dataset; false negatives (FN) are the number of real attack traffic cases that were incorrectly classified with the label "0" in the dataset. True positives (TP) are the number of real attack traffic cases that were correctly classified with the label "1" as in the dataset; true negatives (TN) are the number of real normal traffic cases that were correctly classified with the label "0" as in the dataset. DTR_x refers to the detection time of a single test trial and n represents the total number of experimental tests.

TABLE 3. The five new computed features.

ATC	CPWE	CPRF	RFPSD	RFBSD	CFER	CLASS
SNATC	1.501	6	0.827	75.337	17	0
	1.288	14	0.227	435.38	39	1
TNATC	1.501	8	0.785	123.73	19	0
	1.308	36	0.324	571.02	43	1
FNATC	1.501	9	0.727	178.26	18	0
	1.348	78	0.397	695.30	68	1

A. DATASET DESCRIPTION

A crucial step in validating the proposed detection framework is to select the most suitable dataset that should include real-time IoT traffic. The Edge-IIoTset dataset [38], which is the

most contemporary and relevant dataset for DDoS attacks in IoT contexts, serves as the basis for our analysis and evaluation of the proposed detection framework. The two selected features were the source and destination ports. Table 3 lists the newly calculated features derived from the dataset. The last column represents the traffic type, with "1" indicating attack traffic and "0" indicating normal traffic. In table 3, the feature values of the two classes clearly distinguish between the normal and attack traffic. The term "ATC" stands for attack test case class and is divided into the aforementioned three classes: SNATC, TNATC, and FNATC. These three test cases are addressed in detail in the following sections.

B. EXPERIMENTAL TEST CASES

1) SINGLE-NODE ATTACK TEST CASE (SNATC)

In this experiment, we used the (IoT Network 1) topology depicted in figure 5. In this case, the network is managed by a single POX controller and the proposed detection and mitigation application runs on top of it. In this topology, N1 through N5 represent the IoT nodes and S5 designates the OpenState-enabled network switch. The IoT nodes are directly connected to the S5 switch through a wireless access point (AP 1). After that, the S5 switch is linked to the aggregation S2 switch. Three of the nodes, N2, N3, and N4, were legitimate nodes capable of exchanging regular traffic. N5 is the target of the attack generated by N1, which is an attacker. We begin the experiment by running the *pingall* command, which generates and exchanges ICMP packets between the nodes. This process aims to verify the connection between the linked nodes of the network and collect real-time features under normal traffic conditions. Subsequently, we route normal traffic between all the network nodes. Throughout the experiment, this connection remained open, resulting in a point where legitimate and attack traffic intermingled. Following this, we use the Mininet-IoT's xterm command to open the terminal of the attacker N1 node. We initiated attack traffic against N5 node using the *hping3* tool. The attack traffic rate was set to 20%, which means that attack packets accounted for 20% of all the traffic routed to the target node, whereas normal packets accounted for the remaining 80%.

2) TWO-NODE ATTACK TEST CASE (TNATC)

In this test case, the general network settings were the same as those used in the previous scenario. The sole distinction is the number of target nodes attacked concurrently. In this case, the attacker node (n6) from the (iot network 2) simultaneously sends attack packets to nodes n7 and n9. To achieve the highest possible attack detection accuracy, the proposed framework must be adapted to identify any differences in certain network features and deal with them. one property that can change from the previous scenario is the entropy. in this case, the entropy values increased relatively close to normal values. This is because the attack was distributed to more than one victim.

Thus, when calculating the CPWE feature in a particular window, it may give us normal results; therefore, other features must be considered to be able to detect the attack, even at low rates. We configured the CPWE feature value to three distinct levels to efficiently detect a DDoS attack in different test situations, with attack rates ranging from 20% to 80%. The (IoT Network 2) in figure 6 depicts the topology of this test case.

3) FOUR-NODE ATTACK TEST CASE (FNATC)

In this instance, the attack is simultaneously performed at four nodes at the same time. Nodes N12, N13, N14, and N15 from the (IoT Network 3) are simultaneously attacked by N11 node. The goal of this scenario is to assess the accuracy and efficiency of the proposed framework in dealing with various types and intensities of attack. Attack rates varying from 30 to 80% were employed in this scenario. An attack rate of 20% cannot be considered because it is distributed over four victims. Therefore, the number of packets received by each victim is within the expected normal range.

In this scenario, the CPRF and CFER features both increased rapidly over time, whereas the CPWE, RFPSD, and RFBSD feature values declined. Under the attack condition, the CPWE, CPRF, RFPSD, RFBSD, and CFER features had values of 1.348, 78, 0.397, 695.30, and 68, respectively, compared with 1.501, 9, 0.727, 178.26, and 18, respectively, for the normal condition. The disparity in the feature values between the two conditions clearly differentiates between normal and attack traffic.

C. PERFORMANCE ANALYSIS AND EVALUATION

We extensively simulated the proposed FMDADM framework using the trained BLR, GNB, SVM, kNN, DT, and RF models to evaluate their performance and select the optimal model. We refer to these models as: BLR-FMDADM, GNB-FMDADM, SVM-FMDADM, kNN-FMDADM, DT-FMDADM, and RF-FMDADM. We used the sFlow-RT to periodically record the performance metrics of the models. The three aforementioned test scenarios were executed 120 times with flow entries varying between 5000 and 30,000. Here, we present the results of the FNATC scenario. Because this is the most challenging detection situation, its results are sufficient for judging the proposed framework. The different evaluation metric results for FNATC are listed in table 4. These results were derived from the confusion matrices of the six classification models, as illustrated in figure 6. The "ADT" in table 4 stands for the average detection time across all simulations run for each model under the FNATC scenario, at a 20% attack rate. Of the models tested, RF-FMDADM had the lowest ADT and outperformed the other five ML models for all the metrics.

Table 4 shows that the RF-FMDADM model achieved the highest accuracy rate of 99.79%. DT-FMDADM and kNN-FMDADM came in second and third, with accuracy rates of 99.53% and 98.19%, respectively. The RF-FMDADM recall ratio of 99.77% indicates that out of

TABLE 4. Evaluation metrics results for the four-node attack test case using different classifiers.

Model	ACC	PPV	F1	TPR	SPC	NPV	FPR	FDR	FNR	ADT(µs)
BLR-FMDADM	95.84	88.13	89.52	90.95	97.02	97.78	02.98	11.87	09.05	7.94
GNB-FMDADM	96.94	90.33	92.27	94.29	97.57	98.61	02.43	09.67	05.71	19.36
SVM-FMDADM	97.45	92.53	93.46	94.42	98.17	98.66	01.83	07.47	05.58	39.17
kNN-FMDADM	98.19	94.61	95.31	96.03	98.70	99.06	01.30	05.39	03.97	4.23
DT-FMDADM	99.53	98.38	98.75	99.13	99.62	99.80	00.38	01.62	00.87	3.85
RF-FMDADM	99.79	99.09	99.43	99.77	99.79	99.95	00.21	00.91	00.23	2.31

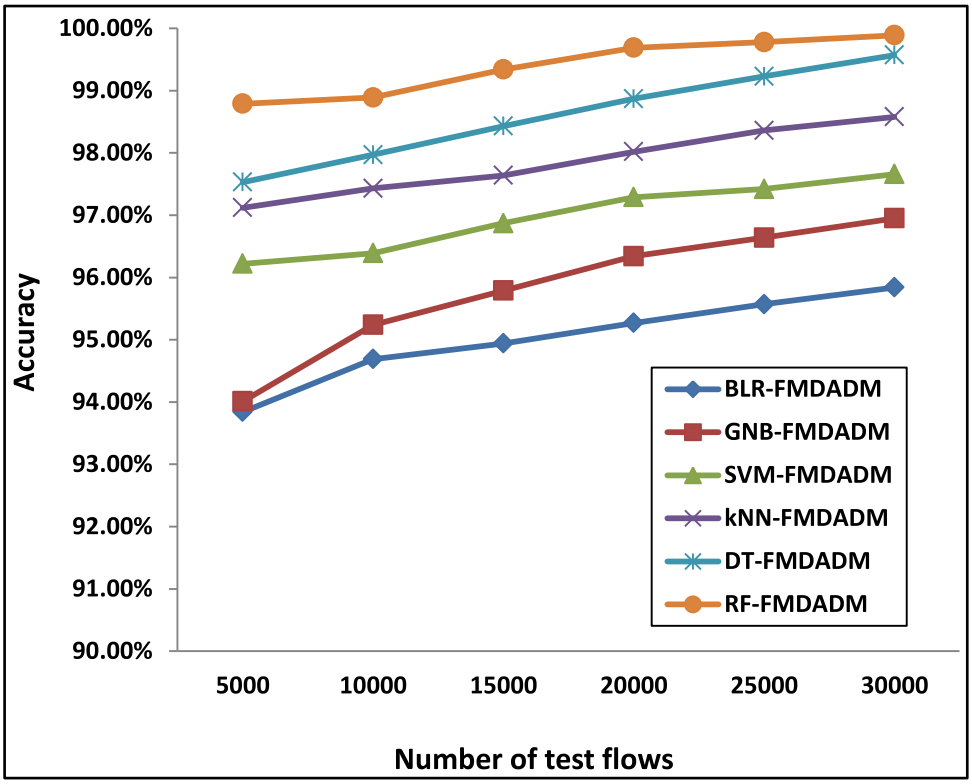

FIGURE 7. Adaptive accuracy results for the six ML models at different flow intensities.

5000 attack flows, the model would be able to identify 4989 flows as attacks. In line with this, among the ML models included in this study, RF-FMDADM provided the highest F1 score of 99.43%. Both GNB-FMDADM and SVM-FMDADM achieved similar results for all metrics except the average detection time. SVM-FMDADM recorded the longest ADT among the six models at 39.17 s. Compared with the other models, BLR-FMDADM achieved the lowest performance results, with scores of 95.84%, 88.13%, 89.52%, 90.95%, 97.02%, 97.78%, 02.98%, 11.87%, and 09.05% for ACC, PPV, F1, TPR, SPC, NPV, FPR, FDR, and FNR, respectively. In table 4, the RF-FMDADM precision value of 99.09% is the proportion of flows successfully identified as attacks across all real attack flows. Among all the models, RF-FMDADM had the lowest false-alarm rates, with an FPR of 0.21%, FDR of 0.91%, and FNR of 0.23%.

Based on these findings, the RF-FMDADM model was considered the best detection model, and its efficacy was evaluated by comparing it with other contemporary techniques. In terms of the adaptive detection accuracy, the proposed RF-FMDADM model outperformed the BLR-FMDADM, GNB-FMDADM, SVM-FMDADM, kNN-FMDADM, and DT-FMDADM models, as shown in figure 7. The number of test flows increased with simulation time, which enhanced the general performance of the RF-FMDADM framework. Following RF-FMDADM, the descending order of ML models that achieved better adaptive accuracy was as follows: DT-FMDADM, kNN-FMDADM, SVM-FMDADM, GNB-FMDADM, and BLR-FMDADM.

Figure 8 shows a Wireshark capture of the IO graph during the SNATC. We can observe how the attack overwhelmed the destination, with TCP requests reaching 2600 packets/s, causing the destination to cease responding to legitimate hosts. Figures 9 and 10 depict a graphical comparison of the total traffic transferred between IoT network nodes in both normal and attack situations, as displayed by the Mininet-IoT

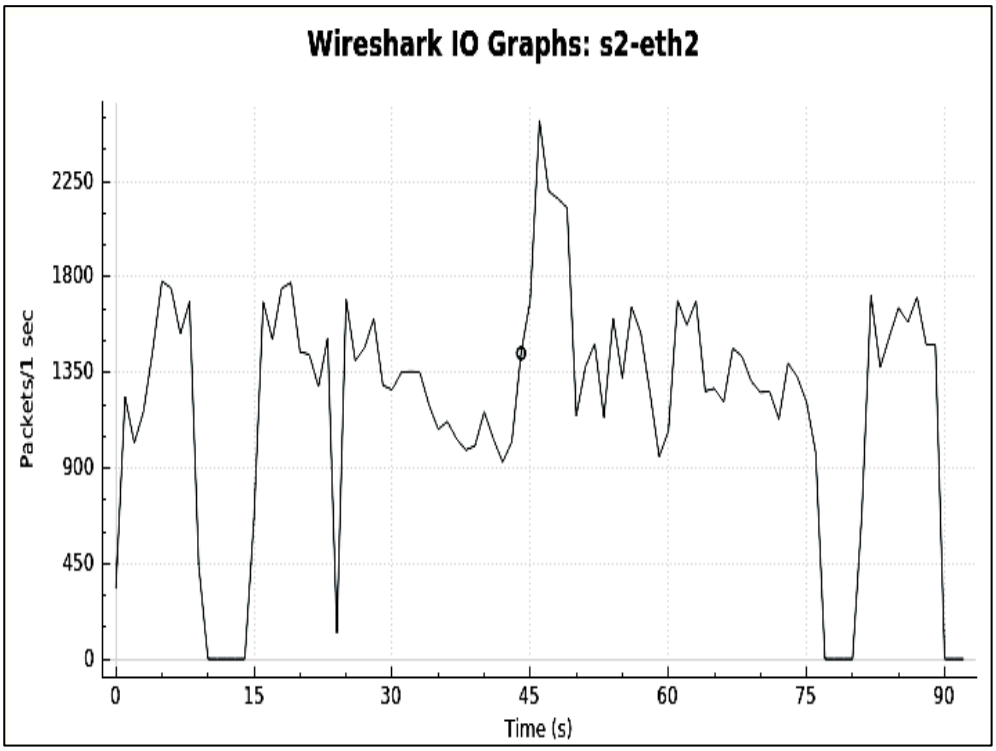

FIGURE 8. Wireshark capture of the IO Graph during SNATC scenario.

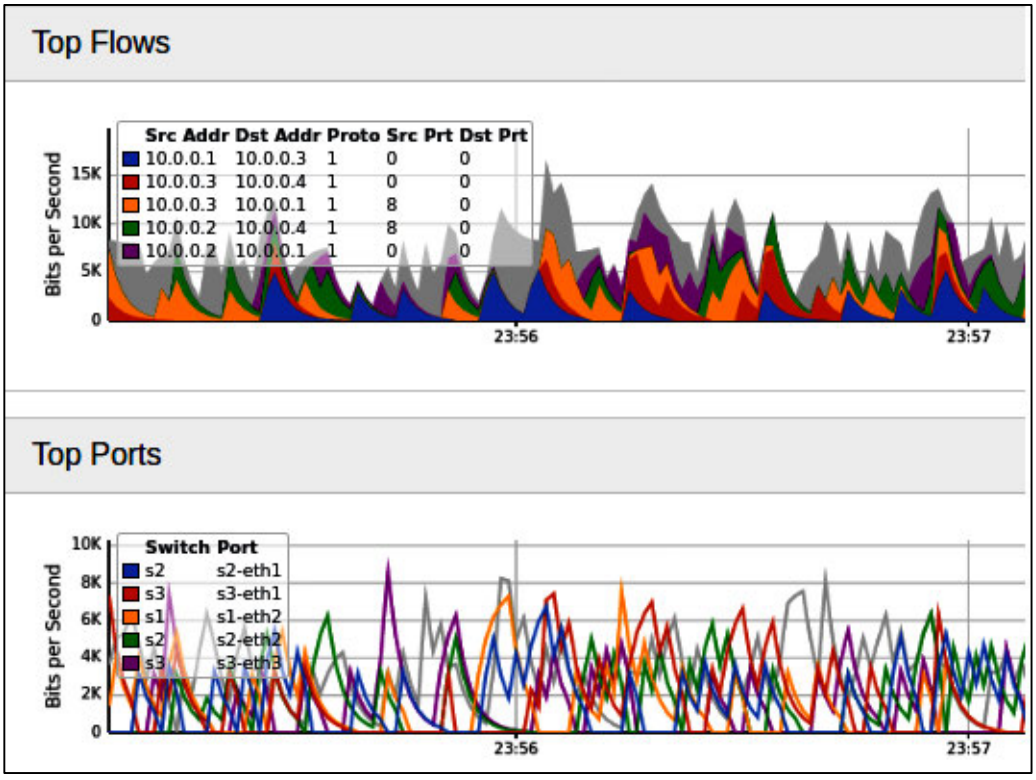

FIGURE 9. The IoT network state under the normal condition.

dashboard. There was significant variation in the overall traffic between the two scenarios. In figure 9, the traffic reaches a maximum of 15 kbps when all 40 nodes are linked to the network to exchange normal traffic. However, in the attack situation, the traffic exceeded 38 Mbps, as shown in figure 10.

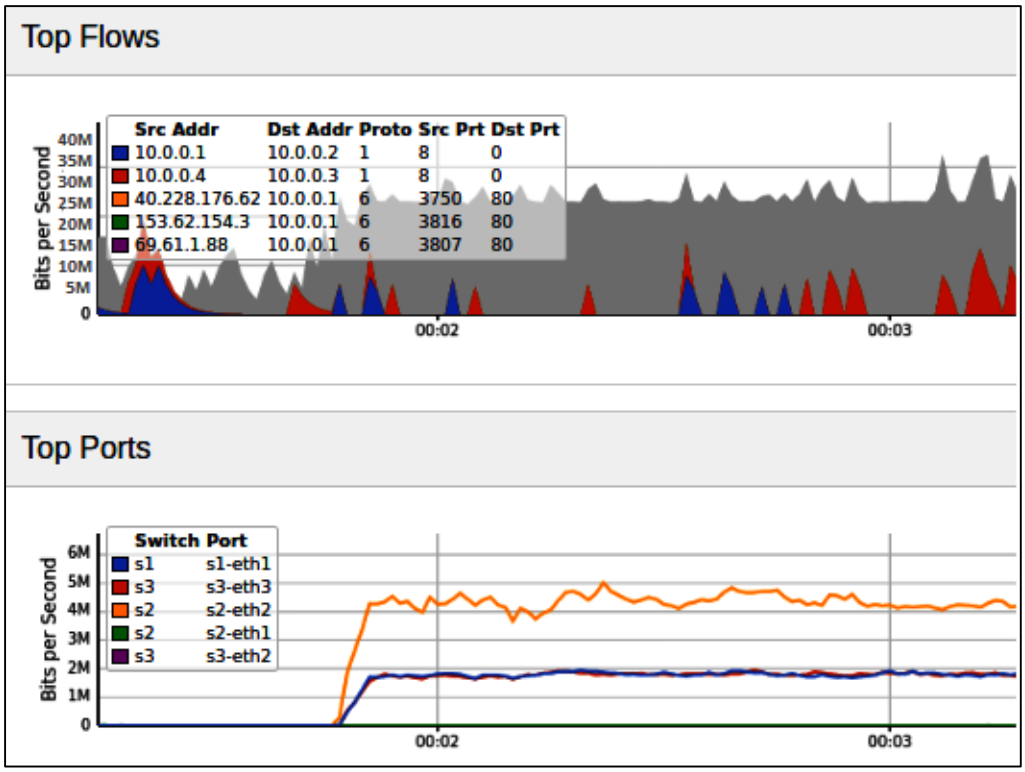

FIGURE 10. The IoT network state under the attack condition.

TABLE 5. Comparison of the proposed framework to existing ML approaches.

Model	Evaluation Metrics							Hyperparameter Settings	
	ACC	PPV	F1	TPR	FPR	TT(s)	DT(µs)	Parameter	Selected value
MLP	92.0	99.7	64.70	47.90	N/A	25.77	1.11	Number of neurons/ layer Activation function No. of hidden layers	10 Sigmoid 2
SMO	95.7	96.0	96.30	96.60	06.00	N/A	N/A	Kernel Kernel coefficient	Rbf 10^{-4}
IBK	97.8	97.9	97.90	97.90	02.20	N/A	N/A	Number of neighbors K Neighbors weight	6 distance-based
J48	93.7	99.9	74.10	58.90	01.30	02.49	02.04	Splitting criterion Min. samples split	Entropy 2
SVM	97.6	99.7	91.60	84.70	N/A	223	139.08	Kernel Reg. parameter coefficient Kernel coefficient	Sigmoid 10^{3} 10^{-2}
REP	93.2	01.0	71.30	55.40	N/A	00.98	01.72	Splitting criterion Number of trees Min. samples leaf	Entropy 70 1
FFCNN	99.0	97.5	96.90	96.30	N/A	10.1	03.81	Number of neurons/ layer Activation function	64 Relu
kNN	98.8	N/A	N/A	98.47	00.97	N/A	N/A	Number of neighbors K Neighbors weight	3 Uniform
Proposed	99.8	99.0	99.43	99.77	00.21	00.97	2.64	Splitting criterion Number of trees Min. samples leaf Min. samples split Max features	Gini 1000 1 2 7

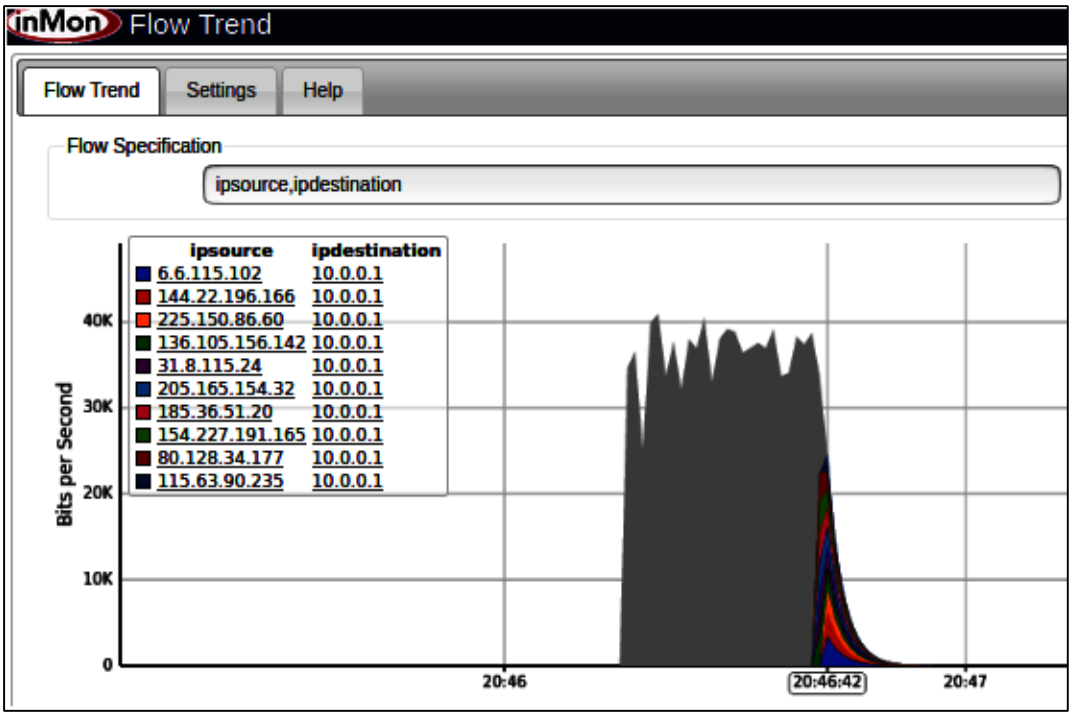

FIGURE 11. The proposed RF-FMDADM framework's detection and mitigation performance.

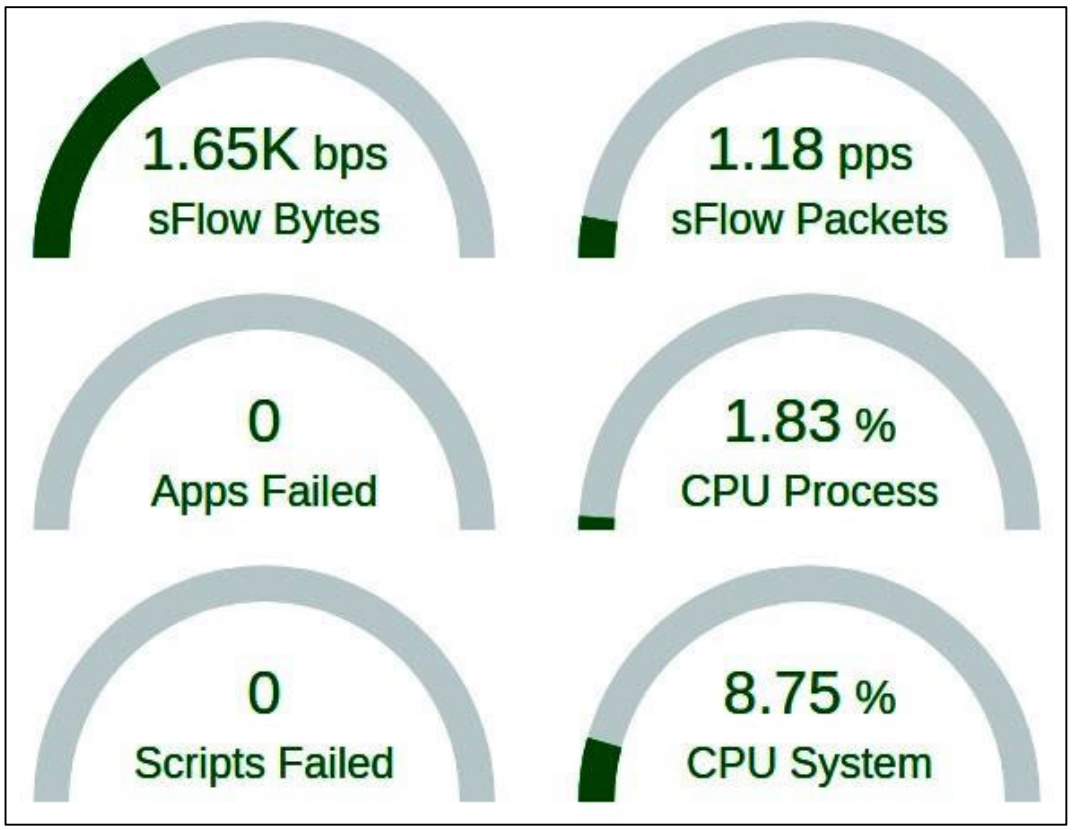

FIGURE 12. The IoT network resources after applying the mitigation strategy.

As seen in figure 10, the attack traffic is combined with benign traffic during the attack operation. The screenshot demonstrates that two legitimate source nodes with IP addresses 10.0.0.1 and 10.0.0.4, communicate regularly with the destination nodes 10.0.0.2 and 10.0.0.3. In contrast, three nodes using spoof IP addresses are targeting node 10.0.0.1.

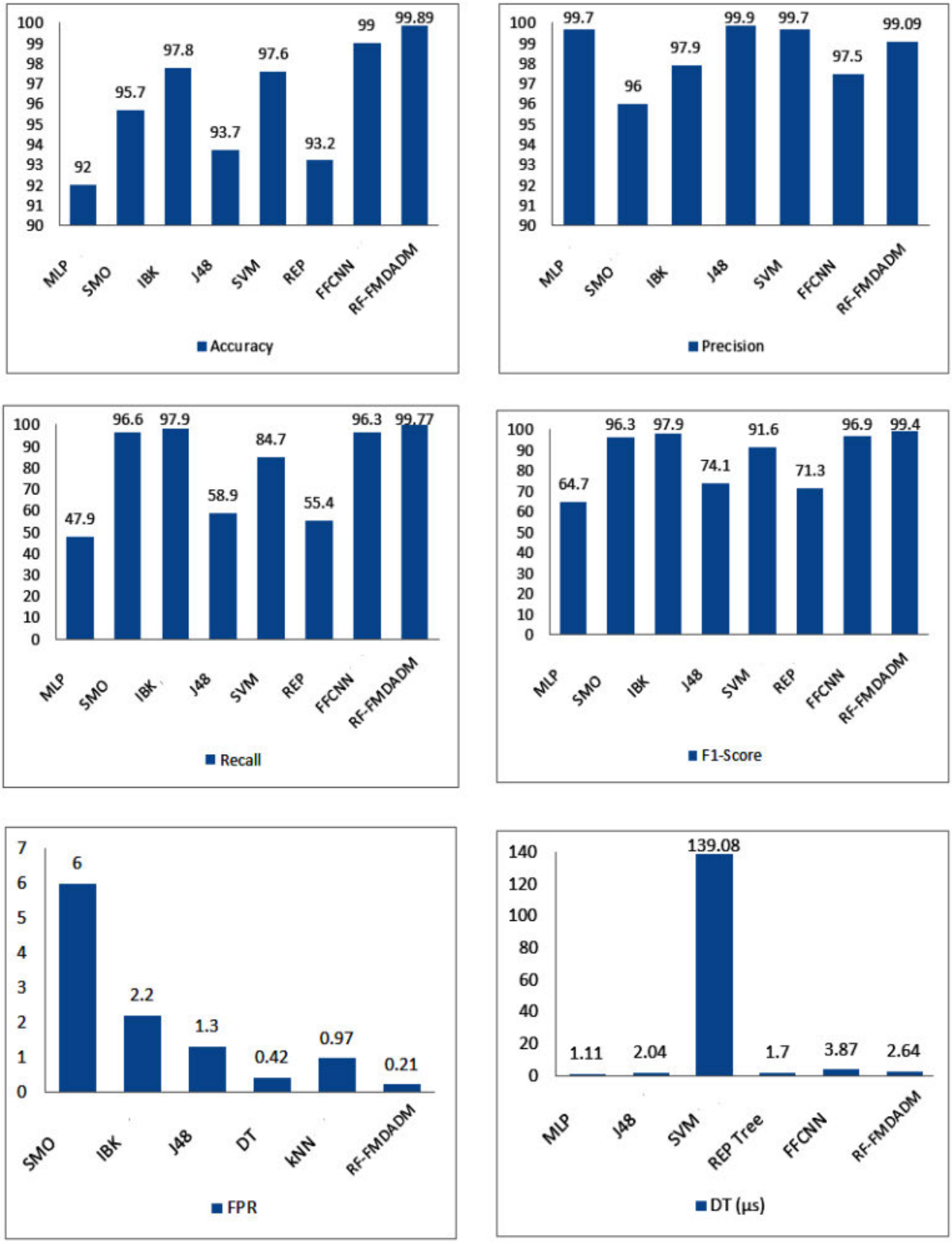

FIGURE 13. Graphical representation of the evaluation metrics for the compared ML methods.

This is an example of TCP flooding attack. The network bandwidth increased to 40 Mbps, representing a very high traffic load compared with the regular condition of a maximum of 10 Kbps. The included switches count the flows, verify the sequence number, compare the byte and packet flows to counters, and provide the required information regarding ingress and egress ports. With a very high packet rate of 6000 packets/s, the network bandwidth increased significantly to 40 Mbps. As the simulation runs longer, network traffic demand increases exponentially. When the proposed framework reduces the effects of the attack, the network traffic drops in chunks, demonstrating the efficacy of the RF-FMDADM deployment. In figure 8, high traffic flow peaks represent DDoS attack-related network traffic, while low dips represent the impact of the RF-FMDADM functionality, which efficiently restores regular traffic flow via OpenState switches by detecting and mitigating the attack.

The proposed RF-FMDADM detects the attack and returns the network performance to the pre-attack condition, as shown in figures 11 and 12. Figure 11 provides convincing evidence that the proposed RF-FMDADM effectively identifies and stops the attack, restoring the network to its pre-attack state. The network measurements demonstrate that the communicating nodes in figure 11 are legitimate and the attackers are blocked. The network was restored to its normal state at a maximum of four packets/s.

Table 5 compares the proposed RF-FMDADM model's performance metrics to those of the current ML approaches and includes the hyperparameter settings for all compared methods. The majority of the algorithms evaluated employ WEKA hyperparameter values. The ''TT'' abbreviation in table 5 stands for the length of time each approach required for training. It is clear that the proposed RF-FMDADM model performs better than the existing ML methods in terms of accuracy, F1-score, recall, and false positive rate, with scores of 99.89%, 99.43%, 99.77%, and 00.21%, respectively. In terms of accuracy, the FFCNN [36] approach comes second after the proposed RF-FMDADM approach, with an accuracy rate of 99.0%. With a precision rate of 99.9%, J48 [36] was the most precise among the nine techniques included in the comparison. However, compared to other approaches, it has a very low recall, F1-score, and accuracy rate of 58.90%, 74.10%, and 93.70%, respectively.

Similarly, the REP Tree [36] achieved comparable results for all metrics to those of J48. In all metrics, SMO [39] and IBK [39] had comparable outcomes, except for FPR, where IBK had a lower FPR of 2.20% and SMO had a higher FPR of 6%. With scores of 98.47% and 00.97%, the kNN [40] approach was second only to the RF-FMDADM in terms of recall and FPR, respectively. However, the authors of [40] did not indicate the method's F1-score or its precision. As seen in table 5, The SVM [36] approach obtained the longest training and detection times among all methods of 223.86 s and 139.08 μs, respectively. MLP [36] had the quickest detection time among all approaches, but it had the lowest F1-score and recall rates of 64.70% and 47.90%, respectively. Figure 13 shows a graphical representation of the accuracy, precision, recall, F1-measure, DT, and FPR for the various ML methods included in the comparison.

V. CONCLUSION

In this paper, we introduced FMDADM, an ML-based DDoS detection, and mitigation framework for SDN-enabled IoT networks. The proposed framework comprises three detection modules and a mitigation module. The first module employs a 32-packet window size used for feature extraction in the third detection module. The second detection module introduces a novel mapping function called DCMF. The DCMF provides two crucial features:- (a) detecting the attack at the data plane level before overwhelming the controller, and (b) discriminating between attack traffic and flash crowds. As a result, the controller has an extra layer of protection against the attack. The third detection module employs feature engineering to identify DDoS attacks using only five new computed features. As a result, the model gives a good fit and can successfully handle the over-fitting problem. The three detection modules resulted in a reduction in the amount of time needed for training, testing, and detection.

We thoroughly tested the proposed framework by employing trained BLR, GNB, SVM, kNN, DT, and RF models to assess their performance outcomes and select the best model. The RF model performed best across all ten evaluation metrics. Three different test scenarios were used to evaluate the performance of the proposed framework. According to the experiments, the FMDADM can detect DDoS with high accuracy in multi-node attack scenarios. This is a crucial area of strength for the proposed framework because it is generally known that conventional defences fall short in the face of these attack scenarios.

The proposed framework is designed to prevent local DDoS attacks produced by IoT Botnets inside compromised LANs from propagating to the ISP level. By protecting the controller and remote nodes in this form, we can stop DDoS attacks before they can reach the Internet. The proposed FMDADM framework can effectively identify DDoS attacks at both high and low rates.

The experimental results show that the proposed framework performed better than most cutting-edge solutions currently available with the following benchmarks for accuracy, precision, F-measure, recall, specificity, negative predictive value, false positive rate, false detection rate, false negative rate, and average detection time: 99.79%, 99.09%, 99.43%, 99.77%, 99.79%, 99.95%, 00.21%, 00.91%, 00.23%, and 2.64 μs.

In the future, we plan to deploy and evaluate the proposed framework in a multi-controller SD-IoT environment. Additionally, we plan to test the proposed framework with other controllers to determine which one works best for the IoT network. In addition, we will test increasingly sophisticated test scenarios. Finally, the detection of more attack types on an SDN-based IoT network may be added to this study.

REFERENCES

[1] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, W. H. Alshoura, and H. Arshad, "The Internet of Things security: A survey encompassing unexplored areas and new insights," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102494, doi: 10.1016/j.cose.2021.102494.

[2] R. Ahmad and I. Alsmadi, "Machine learning approaches to IoT security: A systematic literature review," *Internet Things*, vol. 14, Jun. 2021, Art. no. 100365, doi: 10.1016/j.iot.2021.100365.

[3] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of Things (IoT) security intelligence: A comprehensive overview, machine learning solutions and research directions," *Mobile Netw. Appl.*, pp. 1–17, Mar. 2022, doi: 10.1007/s11036-022-01937-3.

[4] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for IoT security attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022, doi: 10.1109/JIOT.2021.3077803.

[5] M. Azrour, J. Mabrouki, A. Guezzaz, and A. Kanwal, "Internet of Things security: Challenges and key issues," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, Sep. 2021, doi: 10.1155/2021/5533843.

[6] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022, doi: 10.1109/JIOT.2021.3095077.

[7] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman, and M. Bilal, "Smart home security: Challenges, issues and solutions at different IoT layers," *J. Supercomput.*, vol. 77, no. 12, pp. 14053–14089, Dec. 2021, doi: 10.1007/s11227-021-03825-1.

[8] S. Siddiqui, S. Hameed, S. A. Shah, I. Ahmad, A. Aneiba, D. Draheim, and S. Dustdar, "Towards software-defined networking-based IoT frameworks: A systematic literature review, taxonomy, open challenges and prospects," *IEEE Access*, vol. 10, pp. 70850–70901, 2022, doi: 10.1109/ACCESS.2022.3188311.

[9] B. Isyaku, K. B. A. Bakar, F. A. Ghaleb, and A. Al-Nahari, "Dynamic routing and failure recovery approaches for efficient resource utilization in OpenFlow-SDN: A survey," *IEEE Access*, vol. 10, pp. 121791–121815, 2022, doi: 10.1109/ACCESS.2022.3222849.

[10] X. Zhang, L. Cui, K. Wei, F. P. Tso, Y. Ji, and W. Jia, "A survey on stateful data plane in software defined networks," *Comput. Netw.*, vol. 184, Jan. 2021, Art. no. 107597, doi: 10.1016/j.comnet.2020.107597.

[11] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of dos and DDoS attacks in IoT-based stateful SDN: An experimental approach," *Sensors*, vol. 20, no. 3, p. 816, 2020, doi: 10.3390/s20030816.

[12] A. Mahmood, C. Casetti, C. F. Chiasserini, P. Giaccone, and J. Härri, "Efficient caching through stateful SDN in named data networking," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 1, p. e3271, Jan. 2018, doi: 10.1002/ett.3271.

[13] F. Paolucci, F. Cugini, P. Castoldi, and T. Osinski, "Enhancing 5G SDN/NFV edge with p4 data plane programmability," *IEEE Netw.*, vol. 35, no. 3, pp. 154–160, May 2021, doi: 10.1109/MNET.021.1900599.

[14] A. Rahman, M. J. Islam, A. Montieri, M. K. Nasir, M. M. Reza, S. S. Band, A. Pescape, M. Hasan, M. Sookhak, and A. Mosavi, "SmartBlock-SDN: An optimized blockchain-SDN framework for resource management in IoT," *IEEE Access*, vol. 9, pp. 28361–28376, 2021, doi: 10.1109/ACCESS.2021.3058244.

[15] V. Ravi, R. Chaganti, and M. Alazab, "Deep learning feature fusion approach for an intrusion detection system in SDN-based IoT networks," *IEEE Internet Things Mag.*, vol. 5, no. 2, pp. 24–29, Jun. 2022, doi: 10.1109/IOTM.003.2200001.

[16] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez, and B. Rubinstein, "Machine learning in network anomaly detection: A survey," *IEEE Access*, vol. 9, pp. 152379–152396, 2021, doi: 10.1109/ACCESS.2021.3126834.

[17] M. A. Razib, D. Javeed, M. T. Khan, R. Alkanhel, and M. S. A. Muthanna, "Cyber threats detection in smart environments using SDN-enabled DNN-LSTM hybrid framework," *IEEE Access*, vol. 10, pp. 53015–53026, 2022, doi: 10.1109/ACCESS.2022.3172304.

[18] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3559–3570, Apr. 2020, doi: 10.1109/JIOT.2020.2973176.

[19] D. Yin, L. Zhang, and K. Yang, "A DDoS attack detection and mitigation with software-defined Internet of Things framework," *IEEE Access*, vol. 6, pp. 24694–24705, 2018, doi: 10.1109/ACCESS.2018.2831284.

[20] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019, doi: 10.1109/COMST.2018.2866942.

[21] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "A machine learning security framework for IoT systems," *IEEE Access*, vol. 8, pp. 114066–114077, 2020, doi: 10.1109/ACCESS.2020.2996214.

[22] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion detection in industrial Internet of Things network-based on deep learning model with rule-based feature selection," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–17, Sep. 2021, doi: 10.1155/2021/7154587.

[23] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021, doi: 10.1109/ACCESS.2021.3094024.

[24] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion detection system using machine learning for vehicular ad hoc networks based on ToN-IoT dataset," *IEEE Access*, vol. 9, pp. 142206–142217, 2021, doi: 10.1109/ACCESS.2021.3120626.

[25] M. V. O. de Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença Jr., "Near real-time security system applied to SDN environments in IoT networks using convolutional neural network," *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106738, doi: 10.1016/j.compeleceng.2020.106738.

[26] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8, doi: 10.1109/CCST.2019.8888419.

[27] O. Yousuf and R. N. Mir, "DDoS attack detection in Internet of Things using recurrent neural network," *Comput. Electr. Eng.*, vol. 101, Jul. 2022, Art. no. 108034, doi: 10.1016/j.compeleceng.2022.108034.

[28] M. Alanazi and A. Aljuhani, "Anomaly detection for Internet of Things cyberattacks," *Comput., Mater. Continua*, vol. 72, no. 1, pp. 261–279, 2022, doi: 10.32604/cmc.2022.024496.

[29] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2017, pp. 1–8, doi: 10.1109/SMARTCOMP.2017.7946998.

[30] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263, doi: 10.1109/WINCOM.2016.7777224.

[31] N. Zhang, F. Jaafar, and Y. Malik, "Low-rate DoS attack detection using PSD based entropy and machine learning," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 59–62, doi: 10.1109/CSCloud/EdgeCom.2019.00020.

[32] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103160, doi: 10.1016/j.jnca.2021.103160.

[33] F. A. Fernandes Silveira, F. Lima-Filho, F. S. Dantas Silva, A. de Medeiros Brito Junior, and L. F. Silveira, "Smart detection-IoT: A DDoS sensor system for Internet of Things," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Jul. 2020, pp. 343–348, doi: 10.1109/IWSSIP48289.2020.9145265.

[34] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on Web servers in the presence of sampling," *Comput. Netw.*, vol. 121, pp. 25–36, Jul. 2017, doi: 10.1016/j.comnet.2017.03.018.

[35] D. Tang, L. Tang, R. Dai, J. Chen, X. Li, and J. J. P. C. Rodrigues, "MF-AdaBoost: LDoS attack detection based on multi-features and improved adaboost," *Future Gener. Comput. Syst.*, vol. 106, pp. 347–359, May 2020, doi: 10.1016/j.future.2019.12.034.

[36] H. S. Ilango, M. Ma, and R. Su, "A feedforward–convolutional neural network to detect low-rate DoS in IoT," *Eng. Appl. Artif. Intell.*, vol. 114, Sep. 2022, Art. no. 105059, doi: 10.1016/j.engappai.2022.105059.

[37] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against software defined network controllers," *J. Netw. Syst. Manage.*, vol. 26, no. 3, pp. 573–591, Jul. 2018, doi: 10.1007/s10922-017-9432-1.

[38] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[39] S. Das, A. M. Mahfouz, D. Venugopal, and S. Shiva, "DDoS intrusion detection through machine learning ensemble," in *Proc. IEEE 19th Int. Conf. Softw. Quality, Rel. Secur. Companion (QRS-C)*, Jul. 2019, pp. 471–477, doi: 10.1109/QRS-C.2019.00090.
[40] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, "A new framework for DDoS attack detection and defense in SDN environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020, doi: 10.1109/ACCESS.2020.3021435.

WALID I. KHEDR received the Ph.D. degree in computer science from Ain Shams University, in 2009. He is currently a Professor of information technology with the Faculty of Computers and Informatics, Zagazig University. His current research interests include network security protocols, key management protocols, cloud security, and the Internet of Things security.

AMEER E. GOUDA received the B.Sc. and M.Sc. degrees from Zagazig University, Zagazig, Egypt, in 2012 and 2019, respectively. He is currently a Teaching Assistant with the Information Technology Department, Faculty of Computers and Informatics, Zagazig University. His current research interests include network security, machine learning, cloud computing, the Internet of Things, and SDN.

EHAB R. MOHAMED received the B.Sc., M.Sc., and Ph.D. degrees in communication from the Faculty of Engineering, Zagazig University. He is currently a Lecturer with the Information Technology Department, Faculty of Computers and Informatics, Zagazig University. His current research interests include optimization, computational intelligence, wireless networks, SDN, cloud computing, and multimedia.