Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.030183
Article

Tech Science Press

DoS Attack Detection Based on Deep Factorization Machine in SDN

Jing Wang[1], Xiangyu Lei[1], Qisheng Jiang[1], Osama Alfarraj[2], Amr Tolba[2] and Gwang-jun Kim[3,*]

[1]School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha, 410114, China
[2]Computer Science Department, Community College, King Saud University, Riyadh, 11437, Saudi Arabia
[3]Department of Computer Engineering, Chonnam National University, Gwangju, 61186, Korea
*Corresponding Author: Gwang-jun Kim. Email: kgj@chonnam.ac.kr
Received: 20 March 2022; Accepted: 04 May 2022

Abstract: Software-Defined Network (SDN) decouples the control plane of network devices from the data plane. While alleviating the problems presented in traditional network architectures, it also brings potential security risks, particularly network Denial-of-Service (DoS) attacks. While many research efforts have been devoted to identifying new features for DoS attack detection, detection methods are less accurate in detecting DoS attacks against client hosts due to the high stealth of such attacks. To solve this problem, a new method of DoS attack detection based on Deep Factorization Machine (DeepFM) is proposed in SDN. Firstly, we select the Growth Rate of Max Matched Packets (GRMMP) in SDN as detection feature. Then, the DeepFM algorithm is used to extract features from flow rules and classify them into dense and discrete features to detect DoS attacks. After training, the model can be used to infer whether SDN is under DoS attacks, and a DeepFM-based detection method for DoS attacks against client host is implemented. Simulation results show that our method can effectively detect DoS attacks in SDN. Compared with the K-Nearest Neighbor (K-NN), Artificial Neural Network (ANN) models, Support Vector Machine (SVM) and Random Forest models, our proposed method outperforms in accuracy, precision and F1 values.

Keywords: Software-defined network; denial-of-service attacks; deep factorization machine; GRMMP

1 Introduction

The Software-Defined Network (SDN) is an emerging network control paradigm has gained widespread acceptance as a management platform. It has recently gained popularity as a reasult of its effectiveness in management activities, where security concerns are most paramount [1, 2]. In SDN, the switches are only responsible for forwarding, and controls is handled by a centralized controller. The architecture of SDN provides new approaches for configuring and managing the network, which improves network programmability. However, SDN can easily become a target for Denial-of-Service (DoS) attacks. For example, DoS attacks on the data transport layer can affect the awareness of the physical world and control execution [3]. The DoS attacks against client host can be harmful to targeted user host in SDN.

Specifically, DoS attacks are launched directly by a single host or server and come directly to consume resources such as computing, bandwidth and storage of the victim host. DoS attacks launched on client host have characteristics such as stealth and persistence, which make them difficult to be detected. Therefore, the study of DoS attack detection and defense for SDN has become increasingly important in the field of network security.

There are several effective DoS attack detection solutions for traditional networks, however for SDN, the comparable DoS attack detection is rare [4]. Avant-Guard introduced a SYN proxy-based module to verify the legitimacy of each flow based on TCP handshake [5]. Wang et al. suggested that DoS attacks should be detected by monitoring the CPU and memory usage of switch buffers and controllers [6]. Shin et al. used OpenFlow technology as a traffic conditioning tool to monitor the traffic in cloud networks [7,8]. Giotis and Mousavi proposed an entropy-based anomaly detection method [9,10]. Tang et al. proposed a low-rate DoS attack detection method combining tow-step cluster analysis and UTR analysis [11]. Cui et al. proposed a real-time enhanced anti-DDoS strategy [12].

The first problem of feature selection for DoS attack detection is not obvious. In this paper, we directly address the characteristics of the flow rules by sampling the Growth Rate of Max Matched Packets (GRMMP). Since a DoS attack against client host is a high-speed single-source IP attack, there must be a high-speed data stream in the network under attack. The number of matching packets in the flow rules corresponding to this data stream always increases at a relatively steady rate. Therefore, GRMMP is used as the detection feature, which increases fine-grained detection and yields reliable results. The second problem is the low detection accuracy of existing solutions. Therefore, this paper introduces the feature combination mechanism using the DeepFM algorithm. The feature combination detection method can establish the correlation between each feature sample, make the feature samples of updated parameters richer and improve the detection rate. To demonstrate the advantages of the detection method, we compare the method described in this work to an the existing detection method for the DoS attacks against client host and prove the reliability and superiority of the detection method.

There are problems of low accuracy, inconspicuous features and lack of datasets for DoS attack detection in SDN against client host. To solve the above challenges, in this paper, we propose a DoS attack detection method based on DeepFM with GRMMP. The main contributions of this paper can be summarized as follows:

- We designed a DeepFM model to train the flow table data, analyzed the variation law of flow table features, and extracted effective flow table features. DoS attack is detected by a classifier. If DoS attack is detected, an alert message will be sent to the users, and disable relevant flow table rules to alleviate the attack;
- We extracted GRMMP of DoS attack detection flow table as a feature, which can accurately reflect the behavior of DoS attacks. Compared with the traditional detection method based on data layer, this method is more suitable for the cooperation of SDN and controller;
- Through extensive experiments, 140,000 flow entries were obtained, which overcame the problem of lack of publicly available flow table datasets and improved the training performance.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents an overview of DoS attacks in SDN and detection models. Section 4 describes DoS detection based on flow table features. Section 5 shows the experimental results and analysis. And the last section summarizes our work.

2 Related Works

Although SDN has benefits in traffic control and network management, it is prone to DoS attacks. How to defend against such attacks in SDN has attracted a lot of attention. There are currently two types of DoS attacks against SDN: threshold-based detection methods and feature-based detection methods.

Threshold-based detection methods: Detect one or more traffic metrics, such as traffic flow, maximum entropy, and packet delay, by monitoring the network in real-time. Once the traffic index exceeds the set threshold, the network may be attacked. In [10] Mousavi et al. suggested a method for detecting early attacks based on destination address entropy. The method detects an attack if the entropy value falls below a predetermined threshold. Chou et al. analyzed the correlation among links in the network by measuring the time difference of the round-trip time of each Link Layer Discovery Protocol (LLDP) frame, and judged whether there was an attack in the network [13]. Liu et al. used the generalized entropy method to detect the flow on the switch, and found the abnormal switch according to whether the generalized entropy value exceeded the threshold, and then used the neural network for further analysis [14]. Prasath et al. proposed a Bayesian algorithm-based detection method for training on historical network attack data [15]. This method is able to predict the attacking host and prevent it from continuing connection. These entropy-based methods have the advantage of flexible configuration and ease of computation. However, such detection usually relies on only a few metrics, and the results require very precise thresholds. The threshold varies with the network situation, so it is easy to affect the probability of accurate detection.

Feature-based detection methods: The goal is to establish a classifier that can distinguish between normal and attack flow. Usually, statistical analysis, neural network, support vector machine and other methods are used to process the attack characteristics, and then establish a detection model. Kirutika et al. proposed a monitoring system based on the behavior of the controller, which calculated the probability of being attacked by monitoring the behavior of the controller, and used the random forest algorithm to improve the accuracy of the intrusion detection system [16]. To distinguish attacks from normal traffic, Wang et al. proposed the SDN Security Guard (SGuard) Architecture to identify attacks from normal traffic, [17]. The classification module consists of a data collector, a feature extractor, and a Self-Organizing Map (SOM) classifier. They evaluate the performance of SGuard in Mininet and the results show that SGuard is lightweight and efficient. Cheng et al. proposed a DDoS attack detection method based on network flow grayscale matrix feature via multi-scale convolutional neural network (CNN) [18,19]. Meti et al. utilized a support vector machine (SVM) classifier and a Neural Network (NN) classifier to detect suspicious and harmful connections [20]. The experimental results show that the support vector machine technique is a preferable solution for implementing SDN intrusion detection. Sahil et al. proposed a hybrid anomaly detection system based on deep learning [21]. The system combines RBM and SVM to reduce dimension, and classifies normal traffic and abnormal traffic in SDN. However, instead of a flow-based dataset, KDD'99 was employed.

Compared with threshold-based detection methods, feature-based detection methods have higher detection rates and lower false detection rates. However, the previous studies have used SDN-related technologies to solve traditional network security threats, rather than focusing on security issues in the new network paradigm. Therefore, this paper proposes an SDN detection method based on GRMMP. We deploy the detection method on the controller, and obtain the switch flow table information through the controller. Then, features are extracted from the flow table as the input of DeepFM. Finally, according to the output of DeepFM, the network traffic is divided into attack traffic and normal traffic.

3 DoS Attacks and Detection Models in SDN

This section first introduces the principles of SDN and OpenFlow, and provides an in-depth analysis of how SDN works. Then, the principles of DoS attacks against client host are introduced and the impact on SDNs is discussed. Finally, we build a system detection model based on the above analysis and discussion.

3.1 SDN and OpenFlow Flow Table

OpenFlow is an SDN standard communication protocol for routers, switches and controllers for media. The basic idea is to deploy network applications without designing new devices, that is, to maintain hardware independence [22]. The SDN controller can control the underlying network components by the OpenFlow protocol [23]. In SDN, the flow table is the key for OpenFlow to control the network through the control layer. There are multiple flow tables in the switch. When receiving a packet, the switch hands it off to the flow table for processing. There are multiple flow entries in the flow table. Packets are matched sequentially from the table, with sequence numbers backward to 0. The header field of the flow entry is used to detect the packet match. If the input match is successful, the counter is updated and the flow entry action is performed; otherwise, the switch reports to the controller via Packet-In that a new forwarding rule is installed for the mismatched packets. Each forwarding route creates a corresponding rule in the flow table.

3.2 DoS Attack Against Client Host

Due to the separation of control and forwarding in SDN, the forwarding of client packets is handled directly by the switch at the data layer. The controller is transparent to the client. Traditional DoS attacks (such as Land attacks and UDP Flooding) are launched directly by a single host or server that generates only a flow table rule without generating a large number of Packet_in messages to directly consume the resources of the victim host, mostly flooding attacks.

The attacker floods packets to the victim host. When the packet enters the OpenFlow switch, it will look for the flow entry of the victim host to match in the existing flow table sequentially. After a successful match, the packet will be forwarded to the corresponding port through, which will reach the victim host and eventually causes DoS attack on the victim host. When the attack rate is high to a certain extent, it will exhaust the resources of the OpenFlow switch. The OpenFlow switch may also deny service, which will cause all hosts connected to the switch fail to function properly.

3.3 System Detection Model

The programmable nature of SDN allows us to easily deploy detection methods on the controller. The framework of our scheme consists of two main aspects: feature extraction and attack detection. The first step is to extract appropriate network traffic features: the growth in the number of packet matches per sampling interval (i.e., packet match growth rate) is used as a feature to detect DoS attack against client host. Flow table data is trained based on the DeepFM model: firstly, the changing pattern of flow table features is analyzed to extract valid flow table features, then DoS attacks are detected by a classifier for classification, and finally, the system sends alert messages to users and disables relevant flow table rules to mitigate DoS attacks. An overview of the system framework is shown in Fig. 1.

Feature extraction: The raw network traffic data contains a large amount of information, which makes it challenging to analyze DoS attacks directly. We choose GRMMP as feature because of the presence of high-speed data streams in the attack. DeepFM can determine whether DoS attacks disrupt normal traffic, thus showing the characteristics of DoS attack. Detection method based on GRMMP: features are selected from the traffic sequence and DeepFM model for detection of DoS attack.

Feature-based detection: We choose DeepFM to implement detection. Combining the advantages of factorization machines and neural networks in feature learning, DeepFM can classify six attributes of traffic table items into dense and discrete features. We aim to distinguish DoS attack states from legitimate traffic by DeepFM models based on different GRMMPs. Among other things, DoS attacks often result in a large number of small packets (e.g., UDP and TCP packets) flooding the buffers or forwarding queues of network devices, which results in packet loss and may lead to a reduction in average packet size.

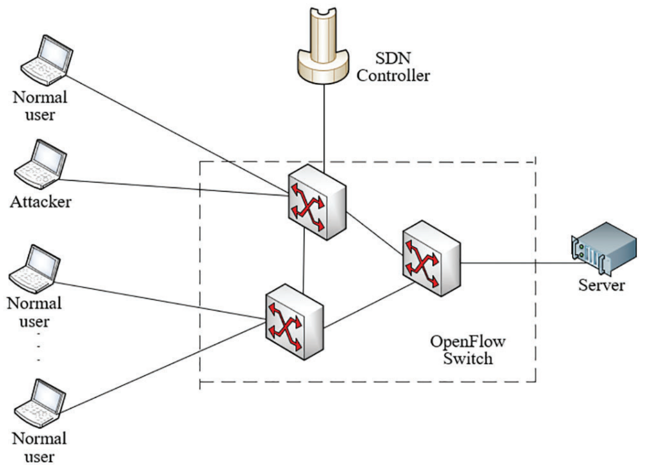

Figure 1: Detection model of attack against client host

In this paper, a detection method based on DeepFM is proposed. This method reduces parameters by designing shallow network structures or by recursive connections [24]. An attempt is made to use GRMMP as feature for detection. The purpose is to achieve more accurate detection and limit the occurrence of false positives as much as possible.

Therefore, the overall workflow diagram of the attack detection system is shown in Fig. 2. Firstly, a DoS attack experiment is set up to obtain enough data for DeepFM model training, and then the trained model is deployed on the server. The user needs first to perform some simple configurations, such as user alert message, sampling interval, etc. and then start the system.

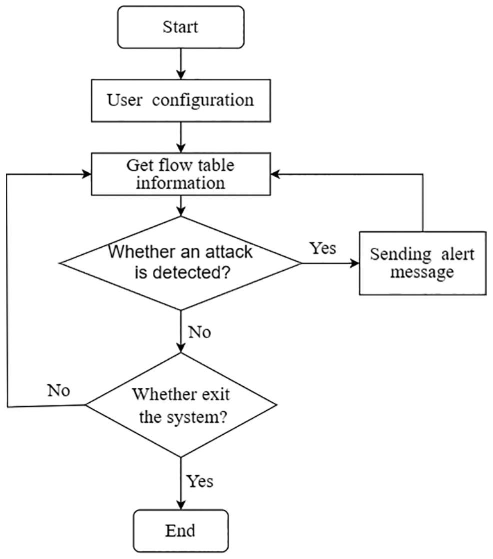

Figure 2: System workflow diagram

When the system samples the flow table information in SDN, the model detects DoS attacks based on the configuration. If a DoS attack is detected, the system will send an alert message and disable the relevant flow table rules to mitigate the attack. The algorithm of the DoS attack detection system framework is shown in Algorithm 1.

Algorithm 1: Framework of our DoS attack detection system.

```
Require:
    Deployment model;
    Start the server;
    Get flow table information;
Ensure:
    Calculate the maximum packet matching growth rate, $V_n$;
    Determine the threshold, $S$;
1:  if $V_n > S$ then
2:      Detection model  flow table information
3:      Start DoS attack detection
4:  else
5:      return to continue to obtain f low table information
6:  end if
7:  while Continue testing do
8:      if An attack is detected then
9:          Send alert
10:     else
11:         return to continue to obtain flow table information
12:     end if
13: end while
```

4 Our Proposed DoS Attack Detection based on Flow Table Features

After the features are extracted, DeepFM detects DoS attacks based on the above features. If a DoS attack is detected, an alert message is sent to the user and the relevant flow table rules are disabled to mitigating the attack. After feature extraction, the detector performs DoS attack detection based on the selected features: (1) Too few features will easily lead to inaccurate detection, while too many features will increase overhead. (2) Representation of features means that extracted features are significantly different from normal flow features. When we study DoS attacks against client host, the flow table features in the switch are very obvious. We detect DoS attacks according to the features of the flow table. In this section, we extract GRMMP features from SDN flow table and design a detection model using DeepFM classifier, which can accurately identify attack traffic and normal traffic.

4.1 DeepFM

Guo Huifeng et al. [25] proposed the DeepFM CTR estimation methodology in 2017. Only the combination of two features is extracted in the classic FM model. The calculation becomes more complicated and the result becomes poorer as the FM model extracts more combinations of features. Deep FM is proposed for extracting both shallow and deep features from data. Deep FM is proposed. The following Eq. (1) can be used to express DeepFM's output:

$$y = sigmoid\ (y_{FM} + y_{DNN}) \tag{1}$$

where $y \in (0, 1)$ is the output of the whole model. y_{FM} is the output of FM part, and y_{DNN} is the output of DNN part. The FM part shares the same input as the DNN. Since the input data is often very sparse, the original input data must first be learned through the embedding layer. The high-dimensional sparse vector is converted into a low-dimensional dense vector. The dense vector with the calculated dimensionality is used as the common input of the FM part and the DNN part.

The FM part is a binary resolver. The calculation Eq. (2) is as follows:

$$y = \langle w,\ x\rangle + \sum_{j_1=1}^{d}\sum_{j_2=i+1}^{d}\langle v_i,\ v_j\rangle x_{j1}x_{j2} \tag{2}$$

where $<w, x>$ is the weighted representation of the first-order features, and the cross term is the weighted representation of the second-order features. The feed-forward neural network is used in the deep neural network. The input data cannot be directly fed into the neural network since the data processed by this model frequently contains a substantial amount of discrete data, which would result in a lower learning effect. As a result the input to this section is a low-dimensional dense vector processed by the embedding layer, which is expressed in Eq. (3) as follows:

$$a^{(0)} = [e^1,\ e^2,\ \ldots,\ e^m] \tag{3}$$

where e^1, e^2, …, e^m is the output of the i-th feature domain processed by the embedding layer. And then extract features using $a^{(0)}$ input as a neural network,

$$a^{(l+1)} = \sigma(W^{(l)}a^{(l)} + b^{(l)}) \tag{4}$$

where σ represents the activation function of the neural network hidden layer, $a^{(l)}$, $w^{(l)}$, $b^{(l)}$ represents the output, weight and bias parameter of the l-th hidden layer, respectively. Then use the output of the last layer as the output of the deep neural network part:

$$y_{DNN} = \sigma(w^{|H|+1}a^{|H|} + (b^{|H|+1})) \tag{5}$$

where H is the number of hidden layers. The benefits of FM and DNN sharing input include two aspects: (1) it is able to learn both low-order feature combinations and high-order feature combinations; (2) because it is an end-to-end model, no feature engineering is required.

4.2 Flow Table Characteristics

The flow table is one of the most important concepts in SDN and a carrier of packet forwarding rules. Each flow table consists of a number of flow rules, which each consisting mainly of matching fields, counters and action records [26]. Matching domains whose task is to synchronize data flows, including all network identifiers from the data link layer to the network layer to the transport layer. The action set relates to the packet forwarding policy used to match packets. Actions can be dropping packets, forwarding packets to the appropriate port, forwarding packets to the controller, etc. The counters are used to record some statistics about the flow rules, including duration, n_packets, n_bytes, idle_timeout and idle_age. See Tab. 1 for explanations of term.

Table 1: Flow table information and explanation of terms

Flow table information	Explanation of terms
duration	The duration of the flow rule, which refers to the time in seconds that the flow rule has survived in the flow table since it appeared.
n_packets	The total number of data flows matched by the flow rule, which refers to the total number of packets matched by the flow rule within this period.
n_bytes	The total size of data streams that the flow rule can match. During this time period, it refers to the total number of bytes of data streams handled by the flow rule.
idle_timeout	The idle timeout of the flow rule, the so-called soft timeout, refers to the longest time the flow rule can be in the unmatched state. Once the unmatched time exceeds the idle timeout, the flow rule will be automatically deleted, in seconds, the default state is 10 s.
idle_age	The time interval in seconds between the last time the flow rule was matched and the current moment in time.

The number of packet matches (n_packets) is a count field carried by each flow rule, which records the total number of packets matched by the flow rule. As the DoS attack against the client host is a high-rate single-source IP attack, there must be a high-rate data flow in the attacking network, and the number of matching packets in the corresponding flow rules of this flow will always increase at a relatively stable rate [27]. Therefore, the increasing number of packet matches per unit time (that is, packet match growth rate) can be used as a feature to detect the DoS attack against client host. Our definition of growth rate of max matched packets is shown in Eq. (6):

$$GRMMP = \frac{|MP_{i+1} - MP_i|}{\triangle t} \tag{6}$$

where $MP_i \quad = max(n\_packets_1,\ \ n\_packets_2,\ \ \ldots,\ \ n\_packets_n)$, which means the largest n_packet among n flow entries is stored in the current flow table. MP_{i+1} denotes the maximum n_packet of the flow table after a sampling interval of Δt. When an attacker launches a DoS attack, the number of matching packets of this attack flow entry increases sharply, resulting in a high value of GRMMP.

For each flow table, duration, n_packets and n_bytes have a continuous variation in time, while in_port, idle_age and actions are relatively discrete and less relevant. GRMMP is a features obtained by mathematical statistics of the entire flow table per second, which can obviously reflect the change of the entire flow table over time.

In this designed method, some fixed fields of each flow entry are directly featured, including duration, n_packet, n_byte, idle_age, in_port, and Actions. The first five features are numeric features that need no processing, while actions are character string features that need to be processed numerically.

We can train DoS attack detection based on DeepFM growth rate of max matched packets. The detailed training process based on DeepFM framework is shown in Algorithm 2. Step1: build the basic model. Step2: label and pre-process the data and encode the data separately. Step3: load the data. Step4: conduct DeepFM training; step5: save the model.

Algorithm 2: Training process of our DoS attack detection system.

```
Require:
    presetted training accuracy threshold, $A$;
    Accuracy during training,$T_n$;
1:  Build model
2:  data preprocessing
3:  Loading data
4:  Training with Data
5:  if $T_n > A$ then
6:      Save training model parameters
7:  else
8:      return to Loading data
9:  end if
```

4.3 Implementation of DeepFM Detection Algorithm

In our DeepFM model, the six attributes of the flow entries are divided into dense and discrete features. Duration, n_packet (number of matched packets), and n_byte (number of matched bytes) are regarded as continuous features. Consider idle_age (idle time), in_port (logical input port) and Actions as discrete features, and train them after feature coding.

DeepFM is a combination of Deep NN and FM using FM for the low-order combination of features and the Deep NN part for the high-order combination of features. The DeepFM model is shown in Fig. 3.

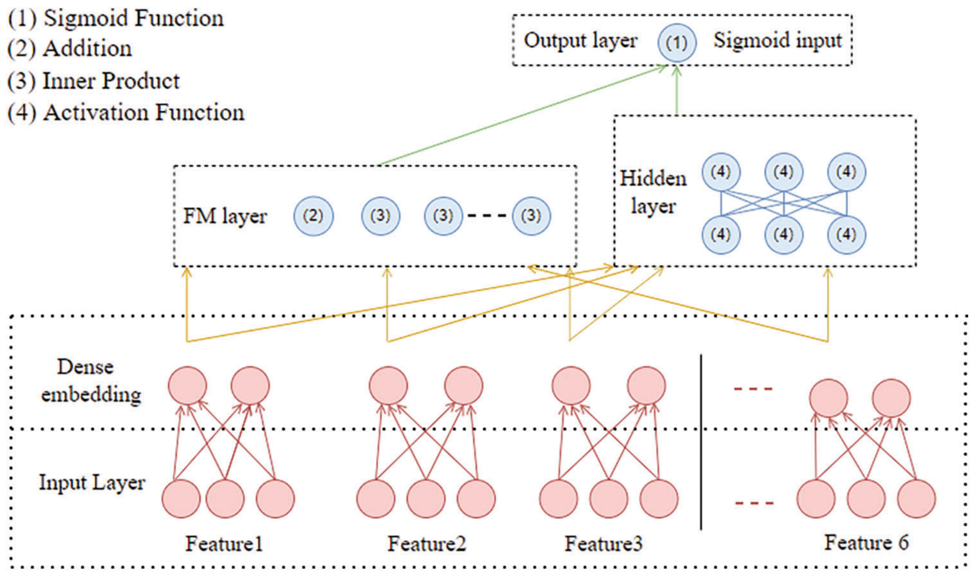

Figure 3: DeepFM structure

The FM algorithm was created to handle the problem of feature combining with sparse data. FM can be used to learn firstorder feature weights and second-order feature combination relations. The inner product of the assumed vectors corresponding to the two features is used to fit the weights of the feature combination relationship corresponding to the two features in the FM model, which can easily capture the second-order combined features. Large amounts of data and sparse features describe the algorithm. It is capable of achieving high performance and useful features.

DeepFM is able to obtain a reasonable prediction even when features and simultaneous occurrences are present in some test data. DeepFM can learn feature combination associations that never or seldom arise in the training set in the case of hign-dimensional sparse data. DeepFM's output is the product of an extra unit and a succession of multiplicative units.

Suppose the extracted flow rule feature sample is X = $\{x_1, x_2, \ldots, x_6\}$, x_1 to x_6 represent duration, n_packet, n_byte, idle_age, in_port, and actions, respectively. The general linear model prediction flow rule is expressed in Eq. (7):

$$y = w_0 + \sum_{i=1}^{n} w_i x_i \tag{7}$$

where n is the feature dimension. In this case, n = 6, x_i is the input feature sample, w_i is the modified weight, and w_0 is the initial weight. The general linear model considers each feature separately, without taking into account their relationship. In practice, however, a large number of features in network attacks are linked. In flow rule feature of the DoS attack against client host, the number of packets matched and the number of bytes matched are often correlated. In general, if the number of matches is higher, the number of bytes matched will also be higher. A polynomial model must be used to combine flow rule features, as indicated in Eq. (2).

The deep neural network of DeepFM is part of the forward neural network, which is used to learn the combination of higher-order features. The raw datasets against the client host DoS attack treat idle_age (idle time), in_port (logical input port), and actions (actions) as discrete features, which tend to be high-dimensional and sparse and lead to difficulties in training the parameters of deep neural networks. DeepFM solves this problemby compressing a high-dimensional sparse input vector into a low-dimensional dense real vector, which is then coupled to the neural network's hidden layer.

There are two particular points to note here in the embedding layer. (1) Although different domains contain different feature dimensions, the embedding vector for each domain has the same size. (2) The implicit vector in FM is shared with the neural network, i.e., it is used as the network weight in the embedding layer to compress the input into the embedding vector. DeepFM does not require training FM before using the parameters that initialize the embedding layer of the neural network. In DeepFM, the FM will be trained as part of the model, together with the other parameters. The input representation of the embedding layer is shown in Eq. (3).

DeepFM uses the same embedding vector for FM and deep neural networks, which has two benefits: (1) From the original data, the embedding vector can learn information about a combination of low and high order features. (2) DeepFM does not require feature engineering specifically designed to limit the structure of the neural network.

The output layer is the accumulation of the FM Layer results and the Hidden Layer results. The fusion of the low-order and high-order feature interactions is then nonlinearly transformed by Eq. (1) to obtain the predicted probabilistic output.

To reduce training time, layer normalization (LN) is used when deploying DeepFM. For each sample, the LN calculates the mean and variance of all neuron inputs in the sample layer. The mean-variance is used to compress the input into a distribution with a mean of 0 and a variance of 1. Then, to ensure the properties of the different neurons, a rescaling and re-panning variable is assigned to each neuron and is rescaled and translated in the previous normalization step for the input [28, 29].

The following is the procedure for detecting DoS attacks using the DeepFM algorithm: First, a six-tuple of features is retrieved and labeled. The parameters are then initialized, and the algorithm's loss function is a Sigmoid function using a stochastic gradient descent process. The technique can determine exactly which flow entry is caused by the attack stream, rather than merely when the attack was set, because the input samples are sampled for each flow entry. This allows for more fine-grained DoS attack detection.

5 Experimental Results and Analysis

In this section, we design experiments and evaluate the DeepFM algorithm. First, there are few publicly available datasets in the SDN, so a simulation experiment was designed. Then we simulate a DoS attack scenario and collect the flow table data from the experiment as our training dataset.

5.1 Experiment Set-up

The purpose of the simulation experiment is to obtain enough datasets for DeepFM training. The principle of the design of the experiment is to simulate the network communication environment and DoS attack process as much as possible. We use Mininet 2.3.2 to emulate SDN. Mininet 2.3.2 includes Open Virtual Switch (OVS) 2.5.0, which is an open source virtual machine and supports the OpenFlow protocol. This software can simulate SDN environment very well. Since the open source Hping3 tool can automatically mobilize APIs through TCL scripts and support customizing various parts of packets, we will use Hping3 to simulate normal data flows and DoS attack flows in DoS attack scenarios.

The experimental hardware is a Windows10 system computer, with Intel Core i7-9750h CPU @ 2.60 Ghz and 16 GB operating memory. The associated model and script implementations are developed using Python3. Its network topology is shown in Fig. 4.

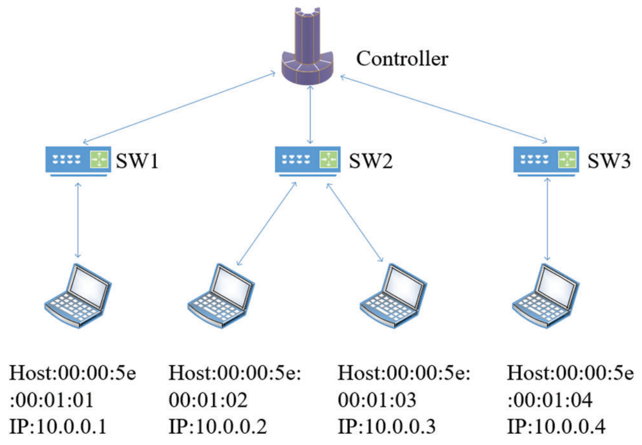

Figure 4: Network topology of DoS attack simulation

The IP addresses of the four hosts are 10.0.0.1 to 10.0.0.4, and the Mac addresses are 00:00:5e:00:01:01 to 00:00:5e:00:01:04. During our experimental attack, first a host is randomly selected as a victim of DoS attack, then a host is randomly selected as DoS attackers, and finally, the unselected hosts are used as normal users in the network, which only generate random normal traffic. In addition, normal traffic and attack traffic are described as follows:

The normal flow: Non-aggressive traffic, which is intended to mimic the normal network environment. Therefore, in our experiment, the nodes of each host will execute the command Hping3 to send normal traffic, and the target host will be random. To make the normal traffic closer to the reality, the sending rate of the normal traffic generated in this experiment will meet the requirements of Poisson distribution, normal distribution, uniform distribution, constant and on-off mode. The on-off mode is a burst traffic mode, which can simulate the burst traffic in the normal network. At the same time, the size of each packet is distributed normally, uniformly and constant.

Attack traffic: For DoS attack traffic, the attack host sends attack packets to the victim host at a constant and faster communication rate than normal traffic. Therefore, we use Hping3 to generate Flood DoS attacks.

In the experiment, we generated a total of 140,000 flow table data. For subsequent evaluation, we used 70% of the dataset for the training set, 20% for the test set, and 10% for the validation set.

5.2 Performance Specifications

To evaluate the ability of the proposed DeepFM to detect DoS attacks in an SDN environment, we use a common approach to verify the merits of the binary classification problem. The samples are divided into two categories, normal and abnormal traffic, and then the actual and predicted cases are calculated to classify the resulting training results into four cases: When the actual traffic is attack traffic and the detection is also attack traffic, we call it TP (True positive), when the actual traffic is normal traffic and is detected as normal traffic, we call it TN (True negative), when the actual traffic is attack traffic but is detected as normal traffic, we call it FN (False negative), when the actual traffic is normal traffic and the detection is attack traffic, it is called FP (False positive). In general, we consider TP and FN to be the result of correct detection, while the rest are detection errors. We will get the following parameters to evaluate the performance of DeepFM:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Precision represents the proportion of the number of samples correctly predicted as attack traffic to the total number of samples predicted as attack traffic.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

Recall refers to the proportion of the number of samples correctly predicted as attack traffic to the total number of actual attack traffic samples.

$$Accuracy = \frac{TN + TP}{FN + TP + FP + FN} \tag{10}$$

Accuracy refers to the proportion of the number of correct predictions in all samples.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{11}$$

In statistics, the F1 value representation is a metric for determining the accuracy of a binary model. It considers the precision and recall of classification models. With a maximum value of 1 and a minimum value of 0, the F1 score can be thought of as a harmonic average of model accuracy and recall.

5.3 Feature Analysis

Fig. 5 shows the sampling of GRMMP under the normal network, where we set Δt = 1 s extract the flow table and perform feature. The results of GRMMP under normal network are tested at 150 s cycle. From the figure, we can clearly see that the GRMMP under normal network has been kept in a low and stable range.

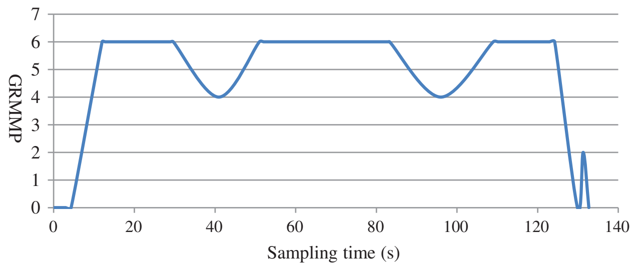

Figure 5: Number of GRMMP in normal sampling

Fig. 6 shows GRMMP sampling in the DoS attack against client host. We set Δt = 1 s extract the flow table and perform feature. The results of GRMMP under DoS attack against client host tested at 20 s cycle. From the graph, we can clearly see that GRMMP stays at a high value during DoS attack against the client host. Therefore, we can use GRMMP as a feature to identify DoS attack against client host.

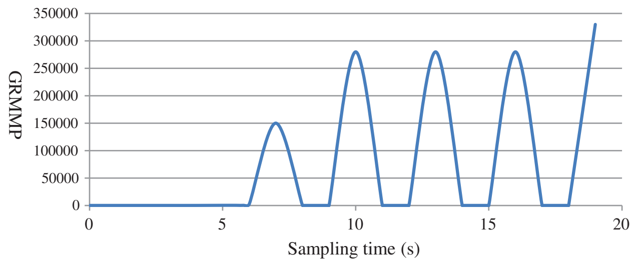

Figure 6: Number of GRMMP in DoS attack sampling

5.4 The Influence of Learning Rate and Hidden Vector Dimension

We try to optimize the model by altering the learning rate and the hidden vector dimension because the model's performance is dependent on the values of the start parameters. We selected four groups of different parameter combinations for the experiment. In the proposed experiment, a total of 22,604 pieces of data are used for testing, including 8908 attack samples and 13,696 normal samples. The evaluated values of DeepFM under different parameters are shown in Fig. 7, and the detection performance under different parameters is shown in Fig. 8.

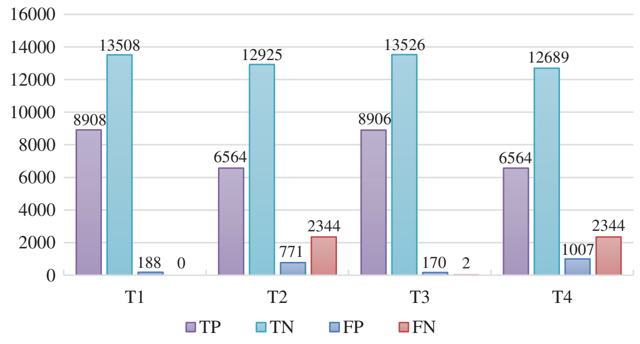

Figure 7: Evaluation values of deepFM under different parameters

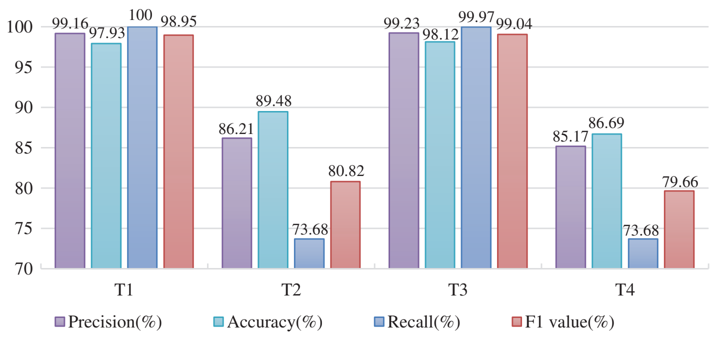

Figure 8: Detection performance under different parameters

In these figures, we take the learning rate γ = 0.001 and the hidden vector dimension K = 8 as type 1 (T1). The learning rate γ = 0.001, and the hidden vector dimension K = 16 as type 2 (T2). The learning rate γ = 0.01, and the hidden vector dimension K = 8 as type 3 (T3). The learning rate γ = 0.01, and the hidden vector dimension K = 16 as type 4 (T4). In Fig. 7, we can see the number of the four types of TP, TN, FP and FN.

In the experiment, the training time of T1 is 169.50 s, T2 is 173.02 s, T3 is 170.05 s, and T4 is 169.48 s. We can see from the experimental results in Fig. 8, when we reduce K from 16 to 8, all evaluation metrics are in an increasing trend by evaluating accuracy, recall, and precision and F1 values. T3 provides us the best results in terms of accuracy, precision, and F1 values, while T1 is slightly higher than T3 in terms of recall. Overall, the DeepFM-based DoS attack detection method has the best detection performance when we set the learning rate γ to 0.01 and the hidden vector dimension k to 8.

5.5 DeepFM Analysis

Accordingly, we compare DeepFM with the K-NN and ANN models mentioned in the literature [30] and the SVM and Random Forest models in the literature [31], and the results obtained are shown in Fig. 9. Also, we compared different models setting the same parameter values for each classifier to tune it to the optimal state. We find that the DeepFM classification model has the highest detection rate for k of 8 and γ of 0.01. The SVM classifier works best with 27 support vectors and 9 support vectors per class. The random forest classifies is best at a maximum depth of 50. K-NN and ANN models have the best detection performance at an attack rate of 400 packets per second. The results show that our algorithm outperforms other algorithms in terms of accuracy, precision, recall and F1 value.

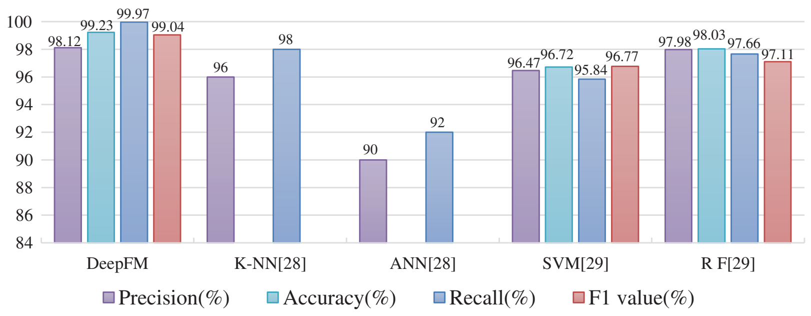

Figure 9: Detection performance of different methods

6 Conclusion

In this paper, a DoS attack detection method of growth rate of max matched packets based on DeepFM is proposed. By extracting packet matching growth rate from flow rules, flow entries are divided into dense features and discrete features. A series of experiments were conducted to obtain 140,000 flow table data for the DeepFM model training. Since the DeepFM model considers both the association between individual flow table features and the ability to extract associations between different flow tables, the DeepFM model can infer whether the network is under attack after training. It has been proven that the growth rate of max matched packets DoS attack detection method based on DeepFM has greater accuracy, precision, recall rate and F1 value by comparing the detection performance of different algorithms and existing DoS attack detection methods. Therefore, our method can detect DoS attacks timely and efficiently.

Funding Statement: This work was funded by the Researchers Supporting Project No. (RSP-2021/102) King Saud University, Riyadh, Saudi Arabia; This work was supported by the Research Project on Teaching Reform of General Colleges and Universities in Hunan Province (Grant No. HNJG-2020-0261), China.

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

References

[1] R. Amin, M. Hussain, M. Alhameed, S. M. Raza, F. Jeribi *et al.,* "Edge-computing with graph computation: A novel mechanism to handle network intrusion and address spoofing in SDN," *Computers, Materials & Continua*, vol. 65, no. 3, pp. 1869–1890, 2020.

[2] Q. Tang, K. Wang, Y. Song, F. Li and J. Park, "Waiting time minimized charging and discharging strategy based on mobile edge computing supported by software-defined network" *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6088-6101, 2020.

[3] J. Wang, W. Chen, Y. Ken, O. Alfarraj and L. Wang, "Blockchain based data storage mechanism in cyber physical system" *Journal of Internet Technology*, vol. 21, no. 6, pp. 1681-1689, 2020.

[4] Q. Yan, F. R. Yu, Q. Gong and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2016.

[5] S. Shin, V. Yegneswaran, P. Porras and G. F. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security*, Berlin, Germany, pp. 413–424, 2013.

[6] H. Wang, L. Xu and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in *IEEE/IFIP Int. Conf. on Dependable Systems and Networks IEEE*, Rio de Janeiro, Brazil, pp. 239–250, 2015.

[7] P. Porras, S. Shin, V. Yegneswaran, M. Fong and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proc. of the First Workshop on Hot Topics in Software Defined Networks*, Helsinki, Finland, pp. 121–126, 2012.

[8] S. Shin, P. Porras, V. Yegneswara, M. Fong and G. Gu, "FRESCO: Modular composable security services for software-defined networks," in *Proc. NDSS*, California, USA, pp. 1–25, 2013.

[9] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras and V. Maglaris, "Combining openFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, no. 7, pp. 122–136, 2014.

[10] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. ICNC*, Anaheim, CA, USA, pp. 77–81, 2015.

[11] D. Tang, R. Dai, L. Tang and X. Li, "Low-rate DoS attack detection based on two-step cluster analysis and UTR analysis." *Human-centric Computing and Information Sciences*, vol. 10, no. 6, pp. 92–104, 2020.

[12] Y. Cui, L. Yan, S. Li, H. Xing and W. Pan, "SD-anti-DDoS: Fast and efficient DDoS defense in software-defined networks," *Journal of Network and Computer Applications*, vol. 68, pp. 65–79, 2016.

[13] L. Chou, C. Liu, M. Lai, K. Chiu and W. Tsai, "Behavior anomaly detection in SDN control plane: A case study of topology discovery attacks," in *Proc. ICTC*, Jeju-si, Jeju-do, South Korea, pp. 357–362, 2019.

[14] Z. Liu, Y. He, W. Wang and B. Zhang, "DDoS attack detection scheme based on entropy and PSO-BP neural network in SDN," *China Communications*, vol. 16, no. 7, pp. 144–155, 2019.

[15] M. Prasath and B. Perumal, "Network attack prediction by random forest: Classification method," in *Proc. ICECA*, Coimbatore, TN, India, pp. 647–654, 2019.

[16] K. Kirutika, V. Vetriselvi, R. Parthasarathi and G. S. V. Rao, "Controller monitoring system in software-defined networks using random forest algorithm," in *Proc. ICCST*, Chennai, TN, India, pp. 1–6, 2019.

[17] T. Wang and H. Chen, "SGuard: A lightweight SDN safe-guard architecture for DoS attacks," *China Communications*, vol. 14, no. 6, pp. 113–125, 2017.

[18] J. Cheng, Y. Liu, X. Tang, V. S. Sheng, M. Li *et al.,* "DDOS attack detection via multi-scale convolutional neural network," *Computers, Materials & Continua*, vol. 62, no. 3, pp. 1317–1333, 2020.

[19] X. Zhu, K. Guo, H. Fang, L. Chen, S. Ren *et al.,* "Cross view capture for stereo image super-resolution," in *IEEE Transactions on Multimedia*, vol. 24, pp. 3074–3086, 2022.

[20] N. Meti, D. G. Narayan and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *Proc. ICACCI*, Silicon Valley, California, USA, pp. 1366–1371, 2017.

[21] G. Sahil, K. Kuljeet, K. Neeraj and R. Joel, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, 2019.

[22] X. Shi, Y. Li, H. Xie, T. Yang, L. Zhang *et al.,* "An openFlow-based load balancing strategy in SDN," *Computers, Materials & Continua*, vol. 62, no. 1, pp. 385–398, 2020.

[23] C. H. Lee and J. S. Park, "An SDN-based packet scheduling scheme for transmitting emergency data in mobile edge computing environments," *Human-centric Computing and Information Sciences*, vol. 11, no. 28, 2021. https://doi.org/10.22967/HCIS.2021.11.028.

[24] X. Zhu, K. Guo, S. Ren, B. Hu, M. Hu *et al.,* "Lightweight image super-resolution with expectation-maximization attention mechanism," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1273–1284, 2022.

[25] H. Guo, R. Tang, Y. Ye, Z. Li and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne, Australia, pp. 1725–1731, 2017.

[26] Q. Zuo, M. Chen, G. Zhao, C. Xing and G. Zhang, "Research on openFlow-based SDN technologies," *Journal of Software*, vol. 24, no. 5, pp. 1078–1097, 2013.

[27] J. Wang, "Research on DoS attack detection and defense methods for SDN," M.S. Thesis, Civil Aviation University of China, Tianjin, China, 2019.

[28] J. Wang, Y. Zou, P. Lei, R. Sherratt and L. Wang, "Research on recurrent neural network based crack opening prediction of concrete dam" Journal of Internet Technology, vol. 21, no. 4, pp. 1161–1169, 2020.

[29] S. Zhang, C. Zhao and F. Gao, "Incipient fault detection for multiphase batch processes with limited batches," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 103–117, 2019.

[30] S. Y. Khamaiseh, A. Al-Alaj and A. Warner, "FloodDetector: Detecting unknown DoS flooding attacks in SDN," in *Proc. ITIA*, Zhenjiang, Jiangsu, China, pp. 1–5, 2020.

[31] M. Yue, H. Wang, L. Liu and Z. Wu, "Detecting DoS attacks based on multi-features in SDN," *IEEE Access*, vol. 8, pp. 104688–104700, 2020.