Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Low-rate Denial of Service attack detection method based on time-frequency characteristics

Yu Fu[1], Xueyuan Duan[1,2,3*], Kun Wang[1,4] and Bin Li[1]

Abstract
In this paper, a real cloud computing platform-oriented Low-rate Denial of Service (LDoS) attack detection method based on time-frequency characteristics of traffic data is proposed. All the traffic data flowing through the Web server is acquired by the collection and storage system, the original traffic data is divided into multiple flow segments by the preprocessing module, and the simple statistical features of several data packets in the flow are extracted by the analysis tool to form the detection sequence. The deep neural network is used to learn the potential time-frequency domain connection in the normal feature sequence and generate the reconstructed sequence. The discrimination module discriminates against the LDoS attack according to the difference between the reconstructed sequence and the input data in the time-frequency domain. The experimental results show that the proposed method can accurately detect the attack features in the stream segments in a very short time, and can achieve high detection accuracy for complex and diverse LDoS attacks. Because only the statistical characteristics of data packets are used, it is not necessary to analyze the data in the packets, which can be adapted to different network environments.

Keywords: Real network environment, Time-frequency characteristics, Low-rate denial of service, Attack detection

Introduction
The rapid development of Internet technology and scale not only changes people's way of life, but also provides a powerful boost to economic development and cultural dissemination, but also brings unprecedented challenges to network security. Due to the openness of internet protocols and services, all kinds of network intrusion attacks never stop. Among many threats, LDoS (low-rate denial-of-service) attack has been especially favored by network criminals and operators of black industry chains since they came out [1]. Different from the flood denial of service attack [2], LDoS attack usually uses a low-rate periodic attack stream to continuously destroy the target, and eventually paralyze it by gradually consuming the available resources of the target.

LDoS attacks are usually initiated by using the protocol vulnerabilities of the transport layer, network layer, and application layer [3]. For example, the LDoS attacks against the transmission control protocol (TCP) timeout retransmission mechanism trigger a large number of packet losses by sending short and high-speed attack pulses to the bottleneck link, forcing the network to start the timeout retransmission mechanism, resulting in a large number of useless data streams exhausting the network bandwidth [4]. For the LDoS attack in the application layer, the connection maintenance mechanism of HTTP is used. The attacker's server continuously sends multiple TCP synchronous (TCP SYN) connection requests but fails to complete the three-way handshake, and the server keeps sending out invalid synchronous data, eventually exhausting the server's allocatable resources, resulting in the failure to respond to normal

*Correspondence: duanxueyuan@xynu.edu.cn
[1] Department of Information Security, Naval University of Engineering, Wuhan 430033, China
Full list of author information is available at the end of the article

requests [5]. Hyper Text Transfer Protocol (HTTP) slow reading attack, the attacker initiates HTTP Get request, and limits the sending and receiving rate of the Web server by setting a small or almost zero value for the sending and receiving window, to occupy the server resources for a long time [6]. Individual LDoS attack flows are usually legitimate impulse network flows in form, and show the same basic characteristics as normal flows. The number of packets sent is small, and the average rate is low, which is generally only 10%–20% of the normal data flow [7]. It is often submerged in normal flows and difficult to be detected by ordinary firewalls. Therefore, the intensity of LDoS attack flows is much smaller than that of DoS attacks, and the fluctuation of network flows caused by it is not obvious. The Cloud computing environment has good adaptability to the change in network traffic, so LDoS attack in a cloud computing environment is not easy to be detected at the initial stage, and the attack effect will gradually appear only as the attack continues. While framework monitors SLA as a cloud monitoring service (SLA-Maas) was proposed by Afzal Badshah et al. [8], which could penalize those who are found in breach of terms and conditions enlisted in SLA, however, at this time, the resources of cloud service platform have been consumed in large quantities, and the user data is in danger of being stolen [9].

Researchers have been exploring new detection methods for LDoS attacks, but some of these methods rely too much on feature engineering, and the detection effect depends on the professional quality and work experience of operators [10, 11]; Although most detection methods are based on deep learning can automatically extract data features, they need enough training samples to complete the modeling [12, 13]. Most of the current detection methods are data stream level detection methods, which need to extract features from multiple packets of each stream. This tracking packet detection method will consume a lot of computing resources. Most of the experimental data come from public data sets or simulation platforms, and there are some defects, such as outdated attack scenes, incomplete traffic, and single traffic background.

To solve the above problems, this paper proposes a method to detect many LDoS attacks only by using the simple statistical characteristics of network traffic packets in the real network environment. This method takes the real network traffic as the background, uses the physical characteristics of LDoS attack pulse stream, takes the size and arrival interval of some data packets in the network stream as the detection data, and uses a neural network to carry out feature learning and attack judgment, thus realizing the end-to-end detection mode from the input of the original traffic to the output of the detection results. Because the normal traffic model is used, this method also has the ability to detect new unknown attacks.

In this paper, we are committed to designing a low-cost detection method for LDoS attacks in a cloud computing platform based on deep learning technology. We analyze the traffic characteristics of LDoS attacks caused by TCP SYN requests and HTTP Get policies, and propose a method to detect the statistical characteristics of packets in traffic segments. The contributions of this paper are as follows:

(1) Obtain traffic data from the original cloud computing platform, and use slicing technology to select the simple statistical features of the first few data packets in the stream segment to construct a sequence of traffic features for detection.
(2) We propose a hybrid detection model based on two-dimensional characteristics of the traffic. The self-encoder with a cyclic neural network structure and convolutional neural network with the residual structure is used to extract the time-frequency domain characteristics of traffic sequence, and the time-frequency domain characteristics are used to improve the detection accuracy.

Related work

In this section, we will review the related literature on machine learning and other techniques for detecting high-rate and low-rate DDoS attacks. Kuzmanovic and Knightly [14], perhaps the first scholars to study LDoS attacks, proposed a new type of low-rate TCP-directed DoS attack in 2003, which they also called "sledge-hammer" [15] (since it can penetrate the network core with a small amount of traffic and destroy any target on the network thus causing great damage). Since then, many detection methods of LDoS attacks have been proposed.

Attackers of LDoS initiates new connections at a high rate within a short period of time and then remains inactive for a long time, and the attack traffic shows high pulse rate and periodic characteristics [16]; while the network under attack shows the volatility of network traffic, which is mainly reflected in the characteristics of source and target IP addresses, time between new connections and traffic rate. Based on the above characteristics, LDoS attack detection methods based on traffic characteristics, machine learning and signal analysis have been proposed.

Traffic feature-based detection methods are usually used to detect abnormal fluctuation features of traffic in the attacked network, which include: queue length, connection duration, packet number, packet interval, packet size, and ACK sequence number extracted from the network [17]. Traffic refers to all messages with the same quintuple (source IP, destination IP, source port, destination port, and transmission layer protocol). Wu et al. [18] extracted the combined features of LDoS attack traffic, input them into a neural network classifier for LDoS attack detection, and verified the effectiveness of its method through a simulation platform and experimental network. Wu's team [10] also studied the multifractal characteristics of LDoS attack traffic [19], and proposed the multifractal trend analysis (MF-DFA) algorithm to estimate the singularity and burstiness of traffic under LDoS attack using HÖlder index, and the experimental results were consistent with the theoretical prediction. Zhang et al. [20] proposed an LDoS attack detection and filtering method based on the ratio of incoming packets to the total number of packets in the flow, according to the phenomenon that normal TCP flows send fewer packets during network congestion and attack flows send more packets, which demonstrated the feasibility of the method by analyzing the data in congested routers. However, these traffic feature-based detection methods suffer from two shortcomings: first, the studies and experiments are conducted in simulated environments or data sets without validation in real networks; Secondly, the extraction of features requires very high data processing expenses and takes a long time to consume, which is only suitable for processing offline data. These two shortcomings limit the application of feature-based detection methods in real-time online detection of LDoS attacks [21].

Detection methods based on machine learning/deep learning usually combine machine learning with other algorithms. Zhang et al. [22] combined principal component analysis (PCA) with support vector machine (SVM) models to detect attacks, using PCA to filter out noise interference and extract the main features of TCP flows, which were used to train SVM models. Yan et al. [23] extracted the mean, variance and entropy and other features to train improved logistic regression models to detect LDoS attacks. Pérez et al. [24] proposed a framework for detecting LDoS attacks in SDN environment which helps to implement various machine learning models such as decision tree, representative tree, random tree, random forest, multilayer perceptron (MLP) and support vector machine for LDoS attack detection. TANG et al. [16] used deep learning technology to calculate the reconstruction error of traffic sequence and realized the detection of LDoS attacks. However, such detection methods based on machine learning or deep learning are all flow-based detection, and there is also a large computational cost in characteristic extraction. Moreover, the characteristic design depends on manual experience, so the attack traffic of unknown characteristics cannot be detected efficiently. Many experiments have demonstrated the feasibility of using signal analysis methods to detect LDoS attacks in network traffic. Du Zhen et al. [25] used wavelet analysis to extract traffic data features, and SVM to complete anomaly separation in mixed traffic. Agrawal et al. [26] used a power spectral density based approach to identify low-speed LDoS attacks in cloud environments, using data collected in the time domain, using Fourier transform to the frequency domain, and calculating the power spectral density values, and if the power spectral density is concentrated in the low frequency band, it is classified as an attack. Brynielsson [27] tried to detect LDoS attacks on application servers using spectral analysis based on the continuous connection feature in the HTTP protocol. Although spectrum classification is effective in detecting LDoS attacks, the signal conversion will cause high computational overhead and high false alarm rate and undetected rate. In the time domain, Wu and Tang et al. [28] combined Hilbert spectrum and Pearson correlation coefficient to detect LDDoS attack packets within a small-scale detection window. Swamiet al [29]. proposed an adaptive detection scheme based on advanced entropy (AEB) that enables the detection of unknown attacks, and although these methods are simple and fast, they all require that the volatility of the traffic data being detected in the time domain must not be excessive, otherwise the detection effect is severely degraded, which is obviously incompatible with the real network situation.

To clearly express the characteristics of each method, we listed the above methods in a table to compare the detection methods. Table 1 is a comparative analysis of the detection methods. The comparative analysis shows that each method has certain advantages.

In summary, the existing LDoS attack detection methods mainly have three problems: First, the data used in the research is out of touch with the current real network environment, which shows that the background of LDoS attack stream generated by the simulator is too monotonous, which is inconsistent with the complicated and changeable current situation of the actual network environment; However, some public data sets collected in the real network environment generally have the problem of outdated attack scenarios. For example, although the CICIDS2017 data set contains several types of LDoS attack traffic, the forwarding rate of 150 packets per second is far from the current attack rate of only transmitting more than a dozen packets per second or even less. Secondly, it is necessary to track the traffic packets during detection, which will result in high calculation expenses and long time delay, and it is difficult to meet the demand for real-time online detection. Third, it is necessary to analyze the information of the complete data stream to

Table 1 Qualitative comparison of detection against LDoS attacks

Category	Detection method	Applicable attack	Positive rate	False positive rate	Speed	Implementation Complexity	Experimental environment
Detection methods Based on features	ACK [16]	LDoS/DDoS	High	Low	High	Low	Virtual machine
	Combined features [17]	LDoS	High	Low	Low	Medium	NS-2
	MF-DFA [9]	LDoS	High	Low	Low	High	Simulation platform
	CPR [19]	LDoS& LDDoS	High	Low	Low	Medium	NS-2
Detection methods Based on ML/DL	PCA-SVM [21]	LDoS	High	Low	Medium	Medium	NS-2
	improved logistic regression [22]	LDoS	High	Low	Low	High	NS-2
	SDN-based [23]	LDoS	Low	Low	High	High	Mininet virtual machine
	NAS-AE [15]	LDoS					NS-2
Detection methods Based on Signal analysis	wavelet analysis [24]	LDoS	Low	High	Low	Low	Simulation platform
	PSD [25]	LDoS&LDDoS	Low	High	High	Low	Real cloud environment
	Hilbert-Huang Transform [27]	LDoS&LDDoS	High	Low	Low	High	NS-2
	spectral analysis [26]	LDoS&DDoS	Low	Low	Medium	High	LoRDAS simulator

get the final detection result, which lacks the ability of early detection. Early detection of network anomalies and timely disposal of network threats has always been the goal pursued in the field of network security.

Data acquisition and design

Data acquisition

Storage system for network traffic acquisition

Due to the limitations of experiments using web simulators or publicly available datasets, the data used in this study were obtained from real Web service networks, preprocessed, and constructed for model training and performance evaluation. For obtaining all the traffic exchanged in the cloud server, a set of high-performance traffic collection and storage system is built. Wireshark is used as the traffic collection software, and the large-capacity disk array is used to permanently store data; In addition, to alleviate the speed difference between traffic acquisition and disk storage, a high-speed RAM cache set is used to absorb the instantaneous peak data of the traffic to ensure the integrity of the collected information. Figure 1 shows the basic architecture of the network traffic collection system. Traffic acquisition and storage system is deployed on the switch closest to the cloud server, and the cloud server traffic is acquired by port mirroring. Wireshark collects all the traffic flowing through the cloud service from the mirror port and temporarily stores it in different RAM in batches. According to the

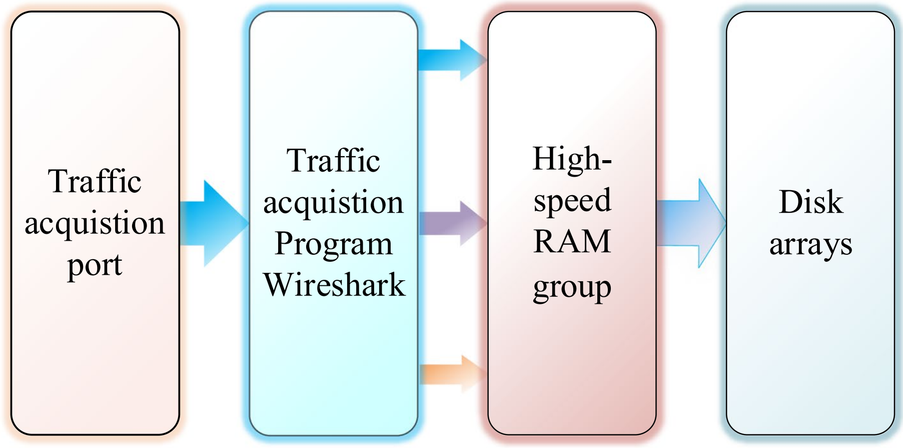

Fig. 1 Network traffic acquisition and storage system architecture diagram

"first come, first out" order, the RAM traffic data is input into the disk array for permanent storage.

Attack traffic acquisition

In order not to affect the normal operation of the network, the attack traffic cannot be transmitted on the Web server network, but can only be generated and acquired in a physically isolated network. Therefore, an attack traffic collection LAN is designed, and its topology is shown in Fig. 2. Attackers 1 and 2 run different attack programs according to the set plan to launch LDoS attacks on the webserver to generate attack traffic. Meanwhile, normal users also run programs that normally visit the web server according to the set rules to simulate the traffic generated by normal users. The traffic collection and storage system collects and stores all traffic flowing through the server from the mirror port of SDN.

The LAN consists of five hosts, four of which are installed with Linux and one with Windows. One Linux host runs an OpenSwitch as an OpenFlow switch, and a Pox controller as an OpenFlow controller to create an SDN environment; the southbound interface is a TCP channel with a bandwidth of 1Gbps and the OpenFlow v1.3 protocol is used for communication between the switch and the controller. 1 Linux host acts as a separate web server; 1 Linux host as a separate cloud server; 1 Linux host as a normal user access cloud server; a Linux host and a Windows host perform LDoS attacks on a cloud server by running different attacks. The client and server machines are connected to the OpenFlow switch, and the specific configuration of each host is shown in Table 2.

The network traffic collection storage system captures all traffic flowing through the server from the mirror port of a company's cloud server. Four instances of the

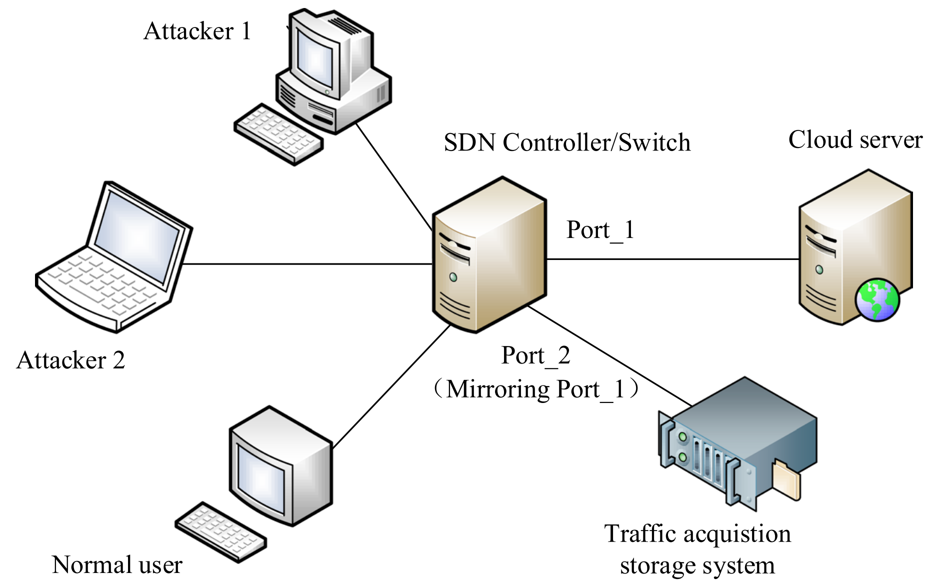

Fig. 2 Attack traffic acquisition network topology diagram

Table 2 LAN host configuration to generate LDoS attack traffic

Host Name	CPU model (frequency)	Memory capacity and frequency	System Version (Kernel)	Running programs
SDN Controllers/Switches	Intel Core i7–8750(2.21GHz)	32GB(2666MHz)	Ubuntu 18.04 LTS(Linux 4.15.0)	OpenSwitch/Pox controller
Servers	Intel Core i7-12700H(2.3GHz)	32GB(4800MHz)	Ubuntu Server 18.04 LTS(Linux 5.2.4)	Apache 2 (Apache for Ubuntu 18)
Attacking host 1	Intel Core i7–8750(2.21GHz)	32GB(2666MHz)	Windows 10(21H2)	Pwnloris (Httpbog)
Attacking host 2	AMD Ryzen R7-4800H(2.9GHz)	16GB(3200MHz)	Ubuntu 18.04 LTS(Linux 4.15.0)	Slowloris/ Torshammer/Hping/ Slowhttptest
General users	Intel Core i7-8565U(1.80GHz)	16GB(2133MHz)	Ubuntu 16.04 LTS(Linux 4.4.34)	Python program for accessing the server normally

client Python program running continuously on the normal user machine communicate with two instances of the server program with different ports to generate normal HTTP traffic of different sizes and different arrival times. The attack traffic is generated by the attack programs running on each attacking host respectively, and the types of attack traffic include various low-rate TCP SYN attacks, HTTP slow read attacks, etc. In order to implement SYN low-rate attacks in small time intervals, we design each attack for 50 seconds, and then sleep for 100 seconds, and the attack occurs only in the first 0.1 seconds of each second attack mode, each attack program running net time of 60 minutes. Figure 3 gives a schematic diagram of the time cycle of the attack program running and sleep.

The slow-read attacks generated by the Slowhttptest program do not use the pulse-shaped attack pattern. By excluding the traffic data of the dormant period from the collected traffic, we can get six types of attack traffic with normal traffic as background, all of which are 60 minutes long. Compared to the pure attack traffic without background, the traffic we designed is more complex and more difficult to detect.

Normal traffic collection

Figure 4 is a simplified network topology diagram containing the traffic acquisition system. The network traffic acquisition storage system is connected to mirror port of cloud server to capture all the traffic data of the server's external interaction. We select from the acquired traffic data for 1200 consecutive minutes without network abnormalities as normal traffic, because this collection period the network can provide normal services, indicating that there is no obvious abnormality in the network, has ruled out any denial-of-service attacks, perhaps there are a small number of other types of attacks, but this attack traffic is very small, the impact on our detection of LDoS attacks can be ignored.

Detection data design

Since the data used in this paper are all from real networks, they need to be pre-processed, feature data selected and data set divided before they can be used as experimental data for training and detection.

Data pre-processing

The main purpose is to remove some defective and redundant data; in addition, some traffic feature values are in the form of text, which need to be operated numerically, and the normalization of each feature value is also needed to improve the operation efficiency. Since the features selected in this study do not involve the specific content of the traffic package, and the data pre-processing is not the focus of this paper, so it is not described in detail.

Traffic feature selection

We know that the general model of pulsed LDoS attack traffic can be represented by a triad of parameters (R,

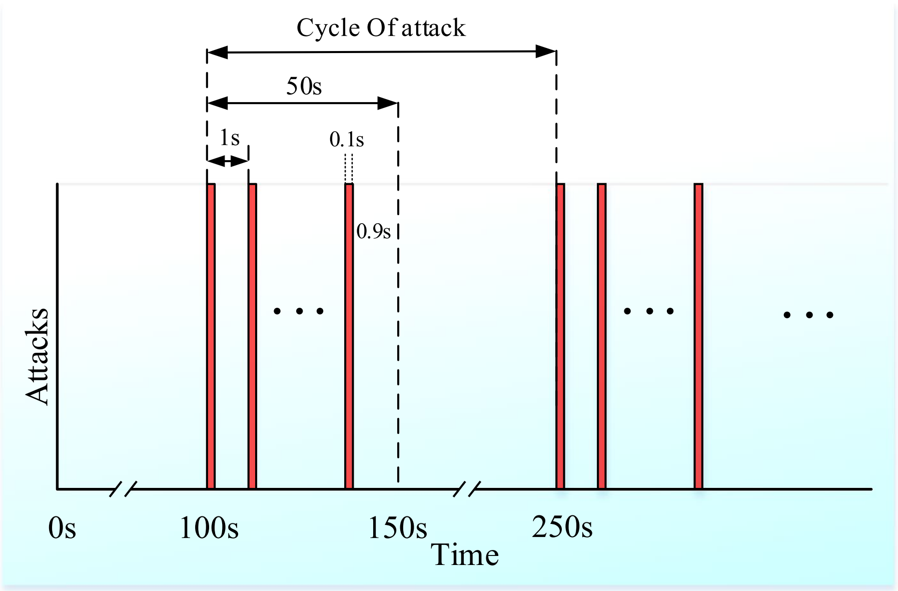

Fig. 3 Schematic diagram of the operation cycle of the attack program

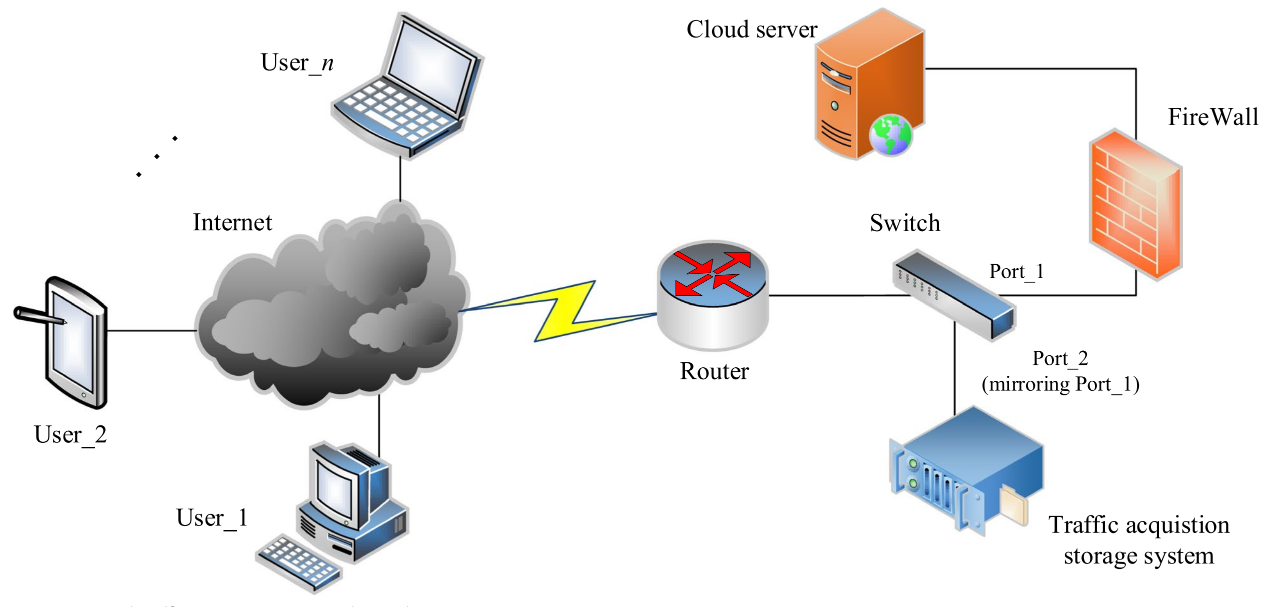

Fig. 4 Normal traffic acquisition network topology

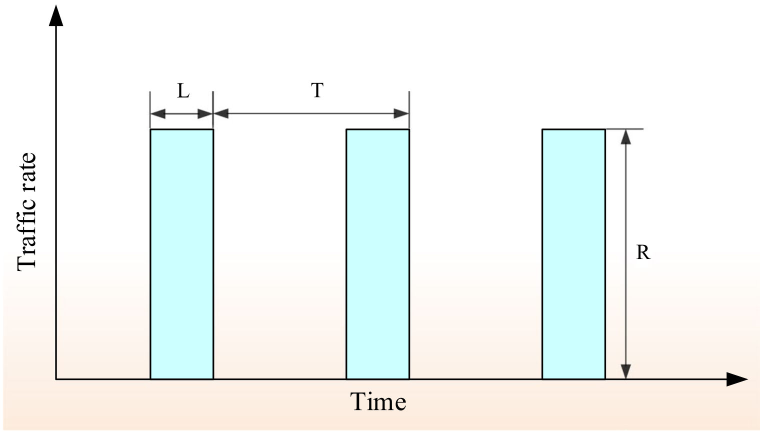

Fig. 5 General model of LDoS attack traffic

L, T), as shown in Fig. 5, where R is the rate of each attack pulse, usually expressed in terms of packets transmitted per unit of time, the rate of LDoS attacks is low and does not occupy the full network bandwidth, usually only 10%–20% of normal network traffic; L is the duration of each attack pulse, and T is the attack cycle.

The traditional approach to detection is to detect LDoS attack traffic by extracting the appropriate characteristics of individual streams, which usually incurs high network overhead because of the need to track the packets of the stream. Streams are all packets with the same five elements (source IP, destination IP, source port, destination port, and traditional layer protocol). From the results of the references [12, 30], it can be found that the size and arrival interval of packets within the first 2 seconds within a stream are very important for detecting DDoS. Therefore, the input sample data attempted to be constructed in this paper is 2-dimensional array consisting of two features of fixed time steps within the stream (the arrival interval time of the first number of packets in the statistical stream and the packet size). Each packet arrival is a time step, so to ensure that the set time step covers a time greater than 2 seconds, for the constructed single sample can be expressed as:

$$x_i = \begin{bmatrix} t_i^1, t_i^2, \cdots, t_i^n \\ l_i^1, l_i^2, \cdots, l_i^n \end{bmatrix}, \quad t_i^j, l_i^j \in \mathbb{R} \ (j \in [1, n]) \tag{1}$$

where t_i^j is the interval between the arrival of the *j*th packet and the *j-1*th packet in the *i*th sample, and l_i^j is the size of the *j*th packet.

Data set construction and partitioning

After pre-processing and feature selection, the stream data is cut into one stream segment every 10 seconds, so 1200 minutes of normal traffic data can get 7200 segments of normal stream segments, and label them as normal samples. Each attack flow sample time is 60 minutes, so each attack flow is cut into 360 flow segments, and a total of 2160 attack flow segments can be obtained and labeled as abnormal samples. In addition, in order to facilitate the aggregation of the detection results of different modules, it is also necessary to add a unique identifier for each stream segment.

This model uses normal traffic data for training and validation. So 40% of normal traffic segments are randomly selected as training sets, 30% of traffic segments are used as verification sets, and the remaining 30% are used to construct test sets.

To construct the detection data set, one attack flow segment is randomly inserted into the normal flow each time, as shown in Fig. 6, so that six sets of synthetic network traffic sets can be obtained by combining different attack flows with normal flows, and the detection set of each attack flow contains 2520 flow segments of 420 minutes in length, with 2160 segments for the basic normal flow segment and 360 segments for the attack flow segment. Attack traffic accounts for 14.29% of the total traffic.

Considering the diversity of attack traffic in real networks, it is necessary to use data containing multiple attack flows to test the detection capability of our proposed method for multiple attacks. Therefore, 120 attack flow segments are randomly selected from each of the six attack flows, and then randomly inserted into the normal flows to generate an “All-United” synthetic flow set containing six attack flows, which contains a total of 2880 flow segments of 480 minutes in length, including 2160 normal flow segments and 720 attack flow segments. The traffic set contains a total of 2880 flow segments of 480 minutes in length, including 2160 normal flow segments and 720 attack flow segments. Attack traffic accounts for 25% of the total traffic. Table 3 shows the relevant information of each dataset.

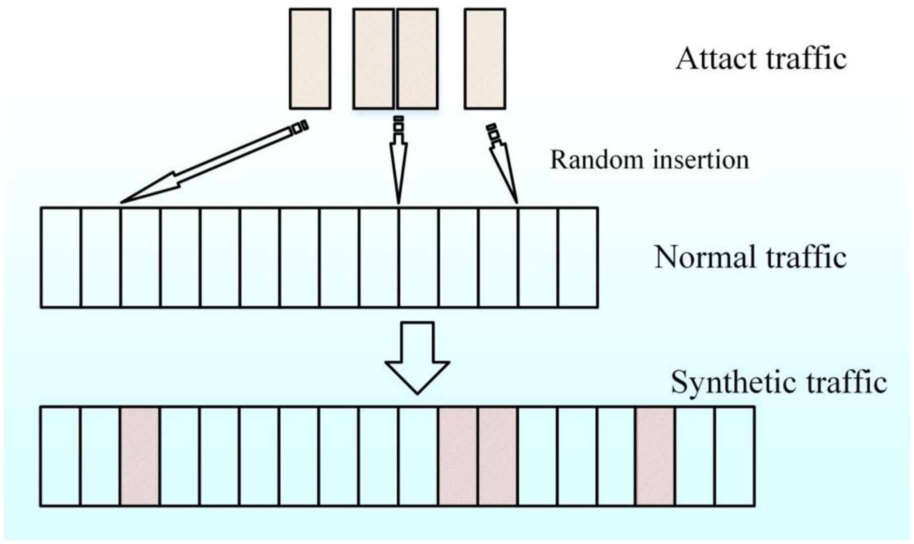

Fig. 6 Synthetic traffic generation schematic

Table 3 Information on each synthetic traffic dataset

Name	Total traffic time (minutes)	Number of normal flow segments (segments)	Number of attack stream segments (segments)	Anomaly ratio
Pwnloris	420	2160	360	0.142 9
Hping	420	2160	360	0.142 9
Torshammer	420	2160	360	0.142 9
Slowloris	420	2160	360	0.142 9
Httpbog	420	2160	360	0.142 9
Slowhttptest	420	2160	360	0.142 9
All-United	480	2160	720	0.25

TFD model

TFD model

Numerous experiments have shown that convolutional neural networks have the efficacy of signal filters, i.e., convolutional neural network structures can be used as extractors of short-time frequency domain features of network traffic [31]; while recurrent neural networks are better at extracting time-dependent information in sequences, i.e., time-domain features of network traffic [32]. Therefore, in order to extract the time-domain and frequency-domain features of the traffic, a reconstruction machine based on time-domain features and a reconstruction machine based on frequency-domain features are constructed in this study, and the time-domain and frequency-domain features in the input traffic are extracted by the two reconstruction machines, and then the reconstruction sequence of the input data is generated based on the extracted features. In order to make full use of the feature information of network traffic in time domain and frequency domain, the LDoS attack traffic anomaly detection model (Time domain & Frequency domain Based Detection, TFD) based on time and frequency domain features is proposed. Figure 7 shows the overall architecture of TFD, including three functional blocks of data preparation, data reconstruction, and attack determination.

Data preparation functional group

This group includes 2 modules of traffic acquisition and data pre-processing, which mainly capture the network traffic from the network interface and perform operations such as corresponding feature extraction and format conversion, converting it into the required form output for the next step of detection. According to the experimental findings, 16 time steps are sufficient to include the first 2 s of the flow, so the traffic data obtained from the real network, only the first 16 packets of information in the network flow studied, that is, the input data of the model is a 2-dimensional array containing 2 features of packet arrival interval and packet size for 16 time steps.

Data reconfiguration functional group

This group consists of three modules: reconstructor based on time-domain features, reconstructor based on frequency-domain features and reconstruction error calculation. This function group is the core component of TFD, which mainly completes the reconstruction of the input feature sequence in two dimensions, time domain and frequency domain, and calculates the deviation of the generated reconstruction sequence from the input sequence, i.e., reconstruction error, as the basis for the next attack determination.

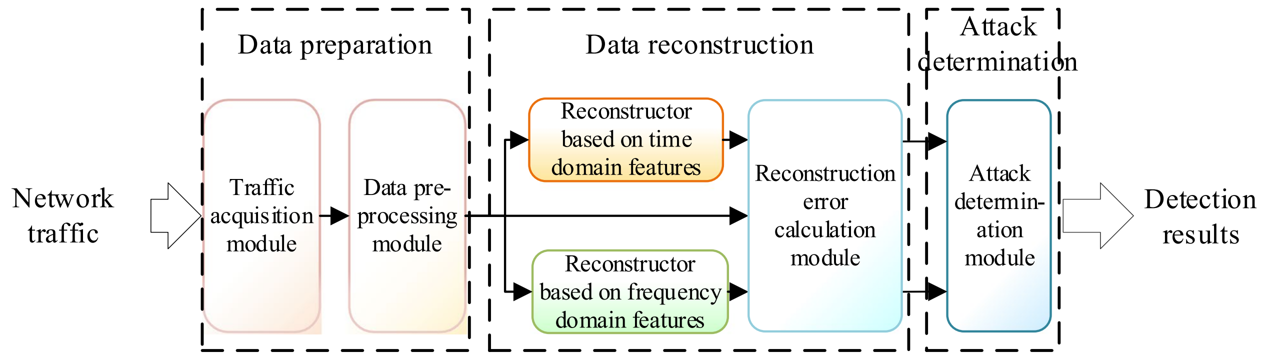
Network traffic
Data preparation
Traffic acquisition module
Data pre-processing module
Data reconstruction
Reconstructor based on time domain features
Reconstructor based on frequency domain features
Reconstruction error calculation module
Attack determination
Attack determin-ation module
Detection results

Fig. 7 Flowchart of LDoS attack traffic detection based on time-frequency features

Attack determination functional group

This group consists of the attack determination module, whose main task is to compare the reconstruction error in both time and frequency domain dimensions calculated in the previous stage with the corresponding threshold, and to determine the flow with a reconstruction error greater than the threshold as an attack flow.

Reconstructor based on time domain features

We proposed a reconstructor based on time-domain features, which uses LSTM auto-encoder to extract and reconstruct the features of the input traffic data. In order to fully extract the data features, a channel enhancement layer is set up between the encoder and decoder, and the specific structure is shown in Fig. 8.

The specific functions of each layer of the reconstructor based on time-domain features are as follows:

(1) **Encoder**

Layer 1, LSTM (32), reads the input data and outputs 32 features, each with 16 time steps.

Layer 2, LSTM (8), takes a 16×32 input from layer 1 and reduces the feature size to 8, and outputs a feature matrix of size 1×8.

Layer 3, the channel enhancement layer (16) copies the 1×8 feature matrix 16 times to form a 16×8 2-dimensional matrix as the input of the decoder layer, which can provide a richer feature representation for the decoder and is the bridge between the encoder and decoder.

(2) **Decoder**

The decoder builds layer 4 LSTM (8) and layer 5 LSTM (32) in the opposite order to the encoder, which are mirrors of layer 2 and layer 1, respectively.

Layer 6 is a fully connected layer that performs matrix multiplication between the output of layer 5 and its internal vector to generate a 16×2 output vector.

Define the encoder function in each layer as $\varphi : \mathcal{X} \to \mathcal{Z}$, which maps the input $x \in R^x = \mathcal{X}$ to $z \in R^z = \mathcal{Z}$, and the decoder layer with the function $\psi : \mathcal{Z} \to \mathcal{X}'$, which maps the input $z \in R^z = \mathcal{Z}$ to $x' \in R^x = \mathcal{X}'$.

Thus, the encoding and decoding process is expressed as:

$$z = \varphi^2\left(\varphi^1(x)\right) = \varphi^2 \circ \varphi^1(x) \tag{2}$$

$$z' = C_{boosted}^{16}(z) \tag{3}$$

$$x' = \psi^1\left(\psi^2(z')\right) = \psi^1 \circ \psi^2(z') \tag{4}$$

$$f_\theta(x) = x' = \psi^1 \circ \psi^2 \circ C_{boosted}^{16} \circ \varphi^2 \circ \varphi^1(x) \tag{5}$$

Where $\circ$ is the joint function, $f_\theta(x)$ denotes the function of the LSTM autoencoder defining the model, and θ is the parameters to be determined for each neuron, using the softsign activation function. The aim of our LSTM autoencoder fitting is to make the output fit the input as closely as possible, using the mean square error as the objective function:

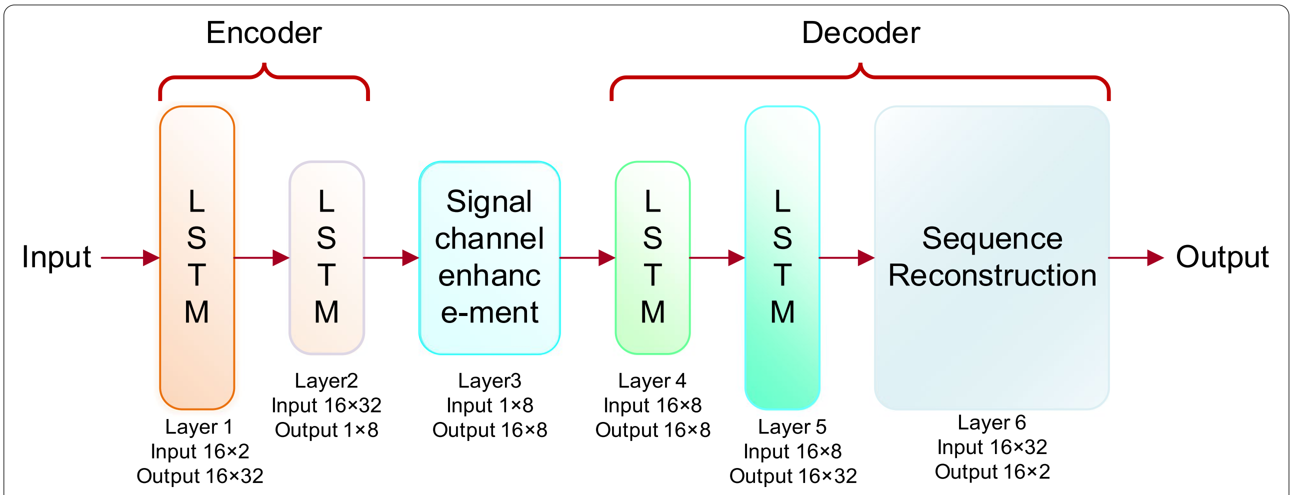
Encoder
Decoder
Input
LSTM
LSTM
Signal channel enhanc e-ment
LSTM
LSTM
Sequence Reconstruction
Output
Layer 1 Input 16×2 Output 16×32
Layer2 Input 16×32 Output 1×8
Layer3 Input 1×8 Output 16×8
Layer 4 Input 16×8 Output 16×8
Layer 5 Input 16×8 Output 16×32
Layer 6 Input 16×32 Output 16×2

Fig. 8 Reconstructor based on time domain features

$$Loss(\theta) = \frac{1}{m}\sum_{i=1}^{m}[x_i - f_\theta(x_i)]^2 \tag{6}$$

where x_i is the input original feature data and $f_\theta(x_i)$ is the output reconstructed data.

Using stochastic gradient descent training, the AdaGrad optimization algorithm is chosen, the learning rate in 0.000 05, the batch size in 16, and 100 complete training sessions of this reconstructor with the training set data. The training process is shown in Algorithm 1.

```
Hyperparameters:
$epochs = 100$          // Number of complete training
$batch\ size = 16$      //Batch size
$\alpha = 0.000\ 05$    // Learning Rate
Input:   Network traffic training data $X$ and label $Y$
Output:  Model Parameter $\theta$
  for $i$ in range($epochs$)
    $g_\theta = 0$          // Resetting gradient parameter
    for $j$ in range(len($X$))     //len($X$) is the number of training data
        $x_j \in X; y_j \in Y$
        $g_\theta \leftarrow \nabla_\theta(\frac{1}{m}\sum_{i=1}^{m}[y_j - f_\theta(x_j)]^2)$   // Update gradient parameter
        $\theta \leftarrow \theta + \alpha g \mathrm{AdaGrad}(\theta, g_\theta)$   // Update model parameter
    end for
  end for
return $\theta$
```

Algorithm 1 Training of reconstructor based on time domain featuresThe reconstruction error decreases with each complete training, and eventually, the reconstructor converges to a steady state in which additional training does not reduce the reconstruction error. The reconstruction error of the validation set data in the steady state is used as the threshold to distinguish normal traffic or attack traffic, namely the threshold for anomaly determination.

Reconstructor based on frequency domain features

We know that both Fourier transform and wavelet transform use convolutional operations to realize the conversion from time domain to frequency domain, therefore, we use a convolutional neural network to extract the frequency domain features in the network traffic. Figure 9 shows the structure of our proposed frequency domain feature-based reconstructor, which is built using a convolutional residual network.

We design a frequency domain feature-based reconstructor consisting of five convolutional modules for extracting features of the input data in layers, denoted as Conv1 to Conv5. They all use 1×2 convolutional kernels with a step size of 1 and a number of channels of 1. The number of convolutional kernels in each layer is 32, 26, 8, 4, and 2, respectively, using the ReLU activation function, due to the fact that only two features in the flow are operated on. Since only two features of the traffic are operated and the feature data are small, this model does not set up pooling layer and dropout layer, and introduces residual structure to prevent the performance degradation of the neural network.

The residual network is composed of a series of residual blocks, and as Andreas et al. [33] argue "direct mapping is the best choice", here a direct mapping of the inputs to the final design is used. Figure 10 illustrates a simple residual block structure, which can be represented as:

$$x_{l+1} = h(x_l) + \mathcal{F}(x_l, W_l) \tag{7}$$

Among them, $h(x_l)$ is the direct mapping part, which needs to be upgraded or downscaled using 1×1 convolution because the dimensions of x_l and x_{l+1} may be

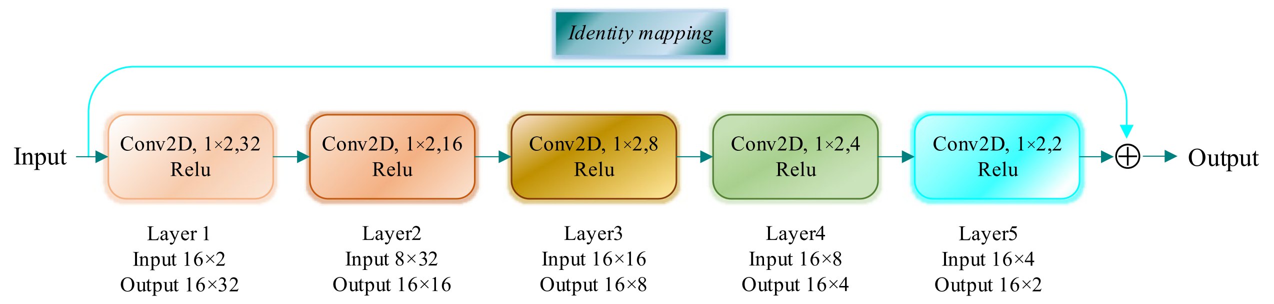
Identity mapping
Input
Conv2D, 1×2,32 Relu
Conv2D, 1×2,16 Relu
Conv2D, 1×2,8 Relu
Conv2D, 1×2,4 Relu
Conv2D, 1×2,2 Relu
Output
Layer 1 Input 16×2 Output 16×32
Layer2 Input 8×32 Output 16×16
Layer3 Input 16×16 Output 16×8
Layer4 Input 16×8 Output 16×4
Layer5 Input 16×4 Output 16×2

Fig. 9 Reconstructor model structure based on frequency domain features

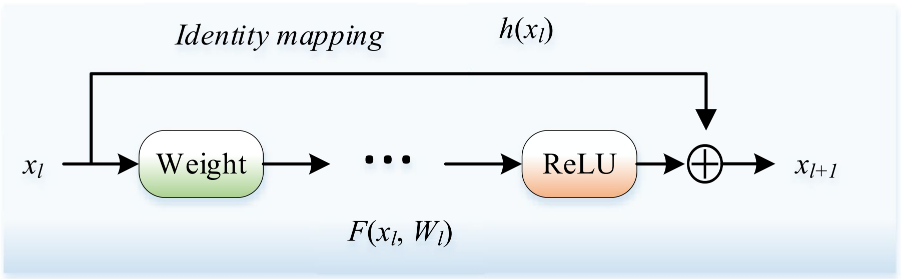

Fig. 10 Reconstructor model structure based on frequency domain features

different; F(x_l, W_l) is the residual part, which consists of multiple convolution operations.

As the input and output dimensions of the reconstructor are the same, it can be mapped directly without dimensional change, so the final reconstructed sequence Y for the input data X after each convolution operation and residual mapping can be expressed as

$$\mathcal{F}(x, W) = \phi^5\left(\phi^4\left(\phi^3\left(\phi^2\left(\phi^1(x)\right)\right)\right)\right) \tag{8}$$

$$g_\vartheta(x) = x' = x + \mathcal{F}(x, W) \tag{9}$$

Where $\phi^k(\cdot)$ denotes the operation of the kth convolutional module, $g_\vartheta(\cdot)$ denotes the function of the convolutional residual network definition model, ϑ denotes the pending parameters of each neuron, and the cross-entropy function is used as the loss function:

$$Loss(\vartheta) = -\frac{1}{n}\sum_{i=1}^{n}\left[x_i \log\left(g_\vartheta(x_i)\right) - (1 - x_i)\log\left(1 - g_\vartheta(x_i)\right)\right] \tag{10}$$

Where x_i is the input original feature data, $g_\vartheta(x_i)$ is the output reconstruction data, trained using stochastic gradient descent method, the optimization algorithm is Adam, the learning rate is taken as 0.000 01, and the batch size is 16. One hundred times of complete training is performed for this reconstructor using the training set data, and the training algorithm is analogous to Algorithm 1. The error of the model in steady state is calculated using the validation set data as the threshold for anomaly determination.

Abnormality determination

Using the normal data trained TFD, the reconstructor can better complete the reconstruction of the normal flow fragment, while for the attack flow fragment because it deviates from the characteristics of the normal TCP flow, the reconstructed sequence will produce a large deviation from the original input sequence, so this deviation can be compared with the reconstruction error threshold to determine whether the input is an attack flow or a normal flow. Since our purpose of detecting attack traffic is to ensure network security, we pay more attention to the recall rate (detection rate) of the attack traffic, and even the presence of appropriate false positives is tolerable, so when either of the reconstructed errors generated by the two reconstructors is greater than the corresponding reconstructed error threshold, the input is determined to be an attack traffic, that is:

$$Traffic = \begin{cases} attack & (r_T \geq R_T \text{ or } r_F \geq R_F) \\ normal & else \end{cases} \tag{11}$$

Where R_T, r_T is the reconstruction error threshold based on the time domain features and the reconstruction error value calculated using the data to be tested, and R_F, r_F is the reconstruction error threshold based on the frequency domain features and the reconstruction error value of the data to be tested.

Experiments and analysis of results

Experimental settings

The hardware and software configurations of the experimental platform for model training and detection are as follows: hardware: Intel Core i9-12900F, 128GBRAM(DDR5), NVIDIA RTX3090; software: Ubuntu 18.04LTS, CUDA11.2, Pytorch1.8.

Evaluation indicators

We design a fast detection method for LDoS attacks based on the TFD model with the aim of quickly discovering LDoS attack traffic from the traffic to be detected. As for the initiation phase of the attack and the type of the attack, they are not our focus, so the detection objective is finally converted into a binary classification problem. Normal traffic is defined as negative samples and attack traffic is defined as positive samples, and five metrics, Accuracy, Precision, Recall, False positive rate (FAR), and F1 value, are used to evaluate the performance of the TFD model, and these metrics are calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = TPR = \frac{TP}{TP + FN} \tag{14}$$

$$FPR = \frac{FP}{FP + TN} \tag{15}$$

$$F_1 = \frac{2 \times Precision \times \mathrm{Recall}}{Precision + \mathrm{Recall}} \tag{16}$$

Where TP, TN, FP, and FN indicate the interrelationship between the true and predicted results, and the specific meanings can be referred to the confusion matrix in Table 4.

Table 4 Confusion matrix

	Predictive Positive	**Predictive Negative**
True Positive	TP	FN
True Negative	FP	TN

TFD model training

The two reconstructors in the TFD model have the same input data, the training process is independent of each other, and both can output results independently, so the two reconstructors are trained separately. Figure 11 shows the loss function values when the two reconstructors are trained for 100 iterations on a training set of All-United traffic data.

It can be seen that the loss function values of the two reconstructors change slowly in the initial training phase, then suddenly and rapidly decrease, and stabilize after reaching a certain procedure. Relatively speaking, the time-domain-based reconstructor has a slight vibration in the loss function value at the initial training stage, but can decrease rapidly and enters a stable state first after about 50 iterations. The frequency domain-based reconstructor, on the other hand, has a stable but slow decaying loss function value at the beginning of training and stabilizes only after 70 iterations, and its steady-state loss value is smaller than that of the time-domain-based reconstructor. The reason why the time-domain-based reconstructor reaches the steady state first may be because the parameters of the model are less than those of the frequency-domain-based reconstructor model, which is easier to train and can complete the training faster, but oscillations may occur in the early training period.

In addition, because the number of samples in the dataset we use is relatively small, and the features of each sample are only a 16×2 2-dimensional matrix with few

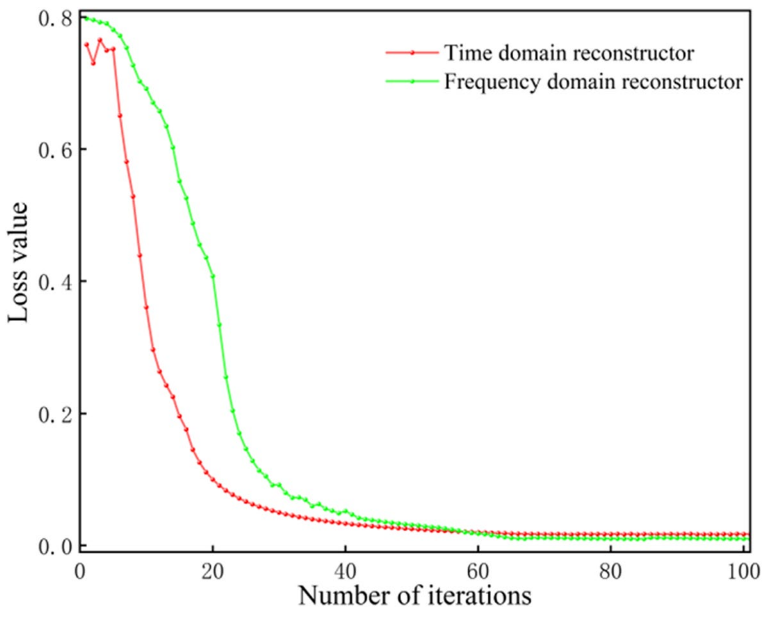

Fig. 11 Anomaly recall rate for each LDoS attack traffic detection

trainable parameters, the average time to complete one iteration of training on the training set for each attack traffic is no more than 7 seconds, which is converted to millisecond detection time for a single sample. In addition, since the detection target is a simple binary classification problem, the computation time to perform anomaly determination can be neglected, which means that our detection model can complete millisecond detection from data input to result output, and can be considered to be able to perform time-to-real detection of network traffic even with the addition of data collection and pre-processing time.

Classifier threshold setting

The error in the steady state of the two reconstructors by using the validation set as the threshold for anomaly determination during testing, combined with their performance during training, was defined as the steady state after 70 iterations, and the average of the errors generated from 71 to 100 training sessions was calculated as the threshold. In addition, to eliminate the chance in the operation, the average of the thresholds calculated for five times was taken as the final threshold of this reconstructor, as shown in Table 5.

Testing results

The testing experiment is divided into two phases, the first phase is to test each attack traffic individually to obtain the detection capability of the TFD model for each LDoS attack; the second phase is to test the All-United dataset consisting of multiple attack traffic to evaluate the detection capability of the model for complex attacks. A 5-fold cross-validation approach is used to calculate the mean value of the model's detection results for each dataset as the model's performance metric. The detection results for LDoS attacks alone are shown in Table 6.

As shown in Table 5, all the results are the average of five detection results, excluding the possible chance factors in computing. It can be seen that the proposed method achieves a recall rate of more than 95% for the detection of each attack traffic. Although the false alarm rate for the Slowhttptest attack exceeds 7%, several other attacks are kept at a low level with an overall false alarm rate of 3.88%, which is acceptable in the design of detection with security alerts as the goal. The best detection indicator for Pwnloris attacks, Pwnloris is actually an upgraded version of Slowloris with more obvious features.

Secondly, the detection of Hping attacks was also good, and the reason for this was analyzed, as the Hping program was originally used for flooding attacks of DDoS. In this experiment, for the effect of low-speed attacks, Hping is set to generate attacks only in the first 0.1 s interval per second, and the attack packet size is fixed, which makes the Hping attack traffic has a significant periodic change characteristic, and therefore is more easily detected.

The reason for the relatively poor detection of Slowhttptest attacks should be due to the Slowhttptest generation is a slow-read attack traffic, the time span itself is relatively large, and we choose data for the first 16 time steps in each 10-second stream segment features, resulting in the selection of some data just to deal with Slowhttptest attack traffic "silent period" so that no obvious data features, which adversely affects the detection.

The TFD model performs well in detecting individual attack traffic, but in order to study the ability of the

Table 6 Detection performance of the model for six LDoS attacks

Validation Sets	Accuracy	Recall	Precision	FAR	F1
Pwnloris	0.982 3	0.980 4	0.984 1	0.015 8	0.982 3
Hping	0.978 3	0.980 5	0.976 2	0.023 8	0.978 3
Torshammer	0.963 7	0.958 6	0.968 4	0.031 2	0.963 5
Slowloris	0.962 3	0.960 4	0.964 0	0.035 8	0.962 2
Httpbog	0.963 7	0.978 6	0.950 2	0.051 2	0.964 2
Slowhttptest	0.948 4	0.971 6	0.928 5	0.074 8	0.949 6
Average	0.966 5	0.971 7	0.962 0	0.038 8	0.966 7

Table 5 Detection domain values for each data set

Validation Sets	Time Domain Reconstructor	Frequency Domain Reconstructor	
Pwnloris	0.017 6	0.011 2	Take the average of 5 times verification results
Hping	0.018 6	0.011 6	
Torshammer	0.016 3	0.018 9	
Slowloris	0.016 8	0.016 6	
Httpbog	0.073 5	0.018 8	
Slowhttptest	0.082 3	0.063 8	
All-United	0.017 4	0.010 9	

proposed method to cope with complex attacks, it is also necessary to test the All-United data set containing multiple attack traffic, and after training the model using the data in the training set, the threshold value is derived using the validation set data, and the same 5-fold cross-validation method is used for the test set data, with the arithmetic of each result The average value is the final result, as shown in Fig. 12 where the average accuracy is 0.935 8, the average precision 0.936 3, the average recall 0.940 7, the average false alarm 0.059 2, and the average F1 value 0.938 4.

Figure 13 shows the recall rate of the TFD model for each LDoS attack type, and it can be seen that the recall rate of detecting one LDoS attack alone can reach more

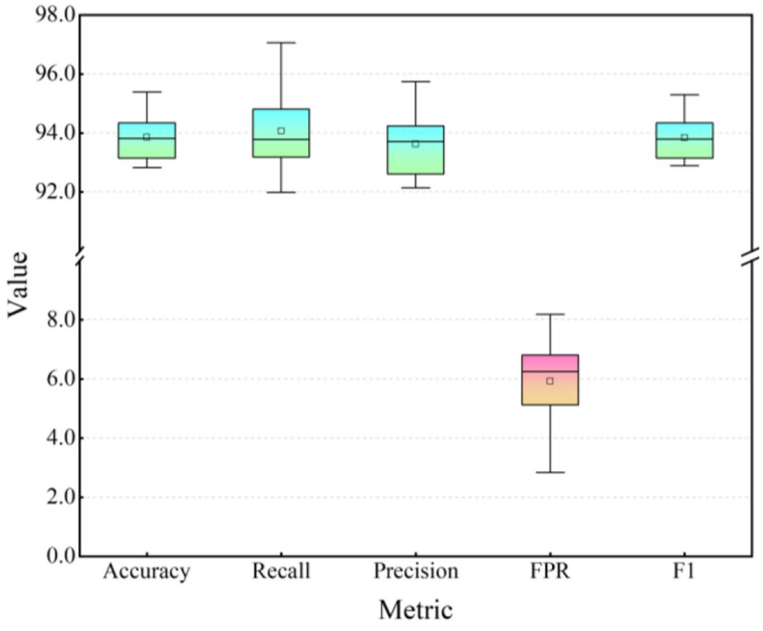

Fig. 12 TFD detection metrics on the All-United attack traffic dataset

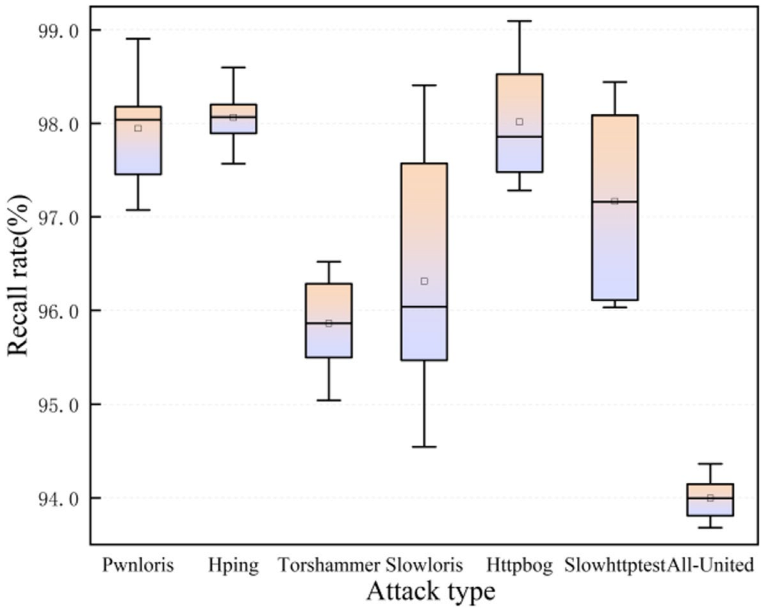

Fig. 13 Anomaly recall rate for each LDoS attack traffic detection

than 95%, and the detection rate of multiple attacks also reaches 94%. In addition, the recall here is calculated in terms of flow segments, and an attack will generate multiple flow segments, so the probability of this attack being detected will be close to 100%, and basically, there will be no problem of undetected attacks, so our proposed method is effective on the dataset we designed.

To verify the adaptability of our proposed method to heterogeneous network traffic data, it is tested respectively on five publicly available datasets, including NSL-KDD, DARPA2000, ISCX2016, CICDDoS2019, and UTSA2021, where:

NSL-KDD dataset is the most commonly used dataset in the field of network traffic anomaly detection research, including ping-of-death, syn flood, smurf and other resource-consuming attacks, NSL-KDD is a dataset generated based on the improvement of KDD-CUP-99, which removes the redundant data in the KDD-CUP-99 dataset and makes an appropriate selection of the ratio of normal and abnormal data, with a more reasonable distribution of the number of test and training data.

The DARPA2000 dataset [28] is a standard dataset in the field of network intrusion detection and is one of three separate datasets in the DARPA dataset. Unlike DARPA 1998 and DARPA 1999, the DARPA2000 dataset focuses on attack traffic for Windows NT and adds internal attack and internal eavesdropping data.

The ISCX2016 low-speed denial-of-service attack dataset [29] is a dataset generated in a simulation environment, where the developers obtained eight different application-layer DoS attack traffic by building web servers such as Apache Linux, PHP5, and Drupal v7, and mixed them with the normal traffic from the original ISCX-IDS dataset to form an LDoS attack traffic dataset.

The CICDDoS2019 dataset [25] is a real-world DDoS attack-like dataset. The latest DDoS attack procedures, including reflective attacks, are used to simulate the generation of attack traffic. There are 50,063,112 samples in the dataset, among which there are 50,006,249 DDoS attack samples and 56,863 normal samples, with very few normal samples, so the dataset is used with the choice of loading normal traffic from external sources or selecting only some attack samples.

The UTSA2021 dataset [30] is a set of normal and attack traffic at different rates generated using the DNS network testbed, mainly including multiple rates of TCP SYN flood attacks, HTTP slow read and slow acquisition attacks, and in this study, a subset of Syn50 with an attack peak of 50 r/s is used to participate in the validation.

Since we design the detection method for LDoS attacks at the transport and application layers, each data set needs to be processed to extract DoS traffic and normal traffic to form a new data set before conducting detection experiments, and then feature extraction is performed on the new data set to form the required data structure for our detection. For larger datasets, such as CICDDoS2019, only about 1% of partial attack samples are selected to improve computing efficiency.

Since the network configuration environment of each dataset has a large variability, the model TFD is trained using some normal samples from each dataset and the threshold value for anomaly determination is calculated before conducting the detection, and then the detection set containing both normal and anomalous samples is tested with the recall rate, accuracy rate and F1 value as the detection index, and the specific results are shown in Fig. 14, which shows that we Concerned about the attack recall rate, basically maintain above 90%, which reached more than 98% on NSL-KDD, DARPA2000, and 96% on CICDDoS2019, UTSA2021, while the recall rate of 91.7% on ISCX2016 is relatively low. To analyze the reason for this is that the attack traffic in the three datasets NSL-KDD, DARPA2000, and CICDDoS2019 are DoS or DDoS attacks, which have more obvious statistical characteristics than LDoS in terms of packet interval and packet size, and thus can be easily identified. In the UTSA2021 dataset, we chose the attack peak of 50 r/s subset Syn50, which is more challenging than detecting attack samples from a dataset with higher attack rates, and then the 96% recall is still a good result. The reason for the relatively low recall of ISCX2016 is probably due to the fact that the attack samples are generated in a different network environment than the normal samples, such that the TFD model trained using the normal samples was more rough and difficult to detect small changes in the statistical features.

The accuracy of the model on the NSL-KDD, DARPA2000, ISCX2016, and CICDDoS2019 datasets did not achieve as excellent results as the recall, and the analysis may be due to the fact that we chose relatively few feature vectors and simple model results, which required much less training parameters compared to large deep network models, using sufficient samples for training. The overfitting problem occurs. This makes the model more “demanding” in determining normal traffic, so that some normal samples are mistaken for attack samples.

The UTSA2021 dataset performs the most spectacularly, even surpassing the All-United dataset we designed. The reason for this is probably due to the network traffic collection environment. The normal and attack samples of the UTSA2021 dataset are generated and collected in the same network environment, so they have good isomorphism and can be trained to produce more “pure” classifiers. In contrast, the normal samples of the All-United dataset are collected from the real network

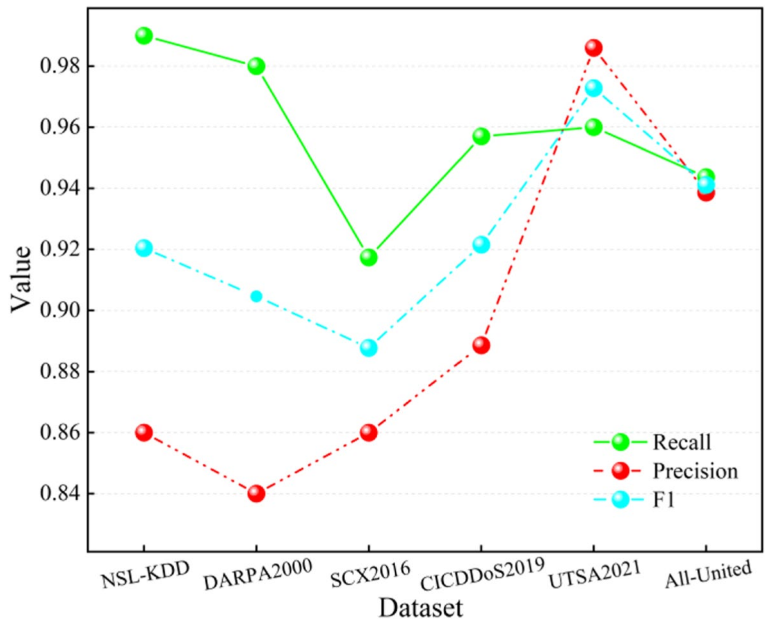

Fig. 14 TFD model for performance detection of DOS attacks in each dataset

environment, which are inevitably disturbed by external conditions and produce "impurities" during the collection process. The attack samples are collected from experimental platforms with similar topology and are less subject to external interference. The model trained with the "impurity" data is a "rough" model, which may ignore the small differences between the feature data and cause misjudgment of the sample type.

The TFD model shows strong adaptability on several datasets, achieving a recall of 91.7% even on ISCX2016. Since we conduct detection experiments with stream fragments, an attack stream is composed of multiple stream fragments, which makes the detection probability of the attack stream will be much higher than the detection probability of the stream fragments, therefore, the TFD detection model we designed can well meet the original design intention of targeting the discovery of attack samples.

Conclusions

For the traditional LDoS attack detection method, it needs to extract more features, consumes more resources, cannot meet the demand of real-time online detection, and the problem that the experimental environment is too different from the real network environment. A method of attack traffic detection (TFD) based on the time-frequency domain features of network traffic is proposed, and only the arrival interval and size of the first 16 packets of the network traffic segment are selected as feature data. The TFD is modeled using normal traffic feature data. The TFD learns the spatial distribution of normal traffic features and can accurately reconstruct the normal traffic, while reconstructing the attack will generate a large reconstruction error, and the model determines the attack based on this error.

The experimental results show that the proposed method can quickly process the traffic feature data obtained from the real network environment and accurately detect the contained multiple attack traffic features with a recall rate of more than 94%, which has the capability requirement of real-time online detection. In addition, although the experimental environment is based on the Web service of a cloud service network, the proposed method can be used in principle for the detection of other LDoS attacks against HTTP servers or HTTPS servers with strong generalizability because the feature data only rely on the statistical information of packet interval and packet size in the network traffic, without the need to manually design the features, let alone analyze the specific contents of the traffic.

Advantages:

(1) The number of detection features is less because only the two features of partial packet size and arrival time interval in the traffic slice need to be considered, so compared with the traditional flow level detection method which needs to consider the complete traffic features, the number of features used is less.

(2) Learning the internal relationship of feature sequences in the time domain and frequency domain is beneficial to mining potential diversity information in traffic and providing more sufficient discrimination references for attack detection.
(3) Have the ability of early detection. It is not necessary to calculate the characteristic data of the complete stream, but only need to count the two characteristics of the first 16 packets to perform the operation. Compared with the traditional detection method, it has the ability of early detection.

Shortcomings:

(1) In this paper, the statistical characteristics of some data packets in the first 10 seconds of the traffic are discussed, the detection results of network traffic characteristics in different granularity are not discussed, and there is a lack of research on the detection of slice traffic segments in different time scales.
(2) It is required that the connection time of the stream is longer than 10 seconds, and the eigenvalue of the stream less than 10 seconds is filled with "0". This will lead to an increase in the operation cost compared with the original traffic. Secondly, this filling operation of attack traffic and normal traffic may lead to an increase in false alarms.

Abbreviations
DDoS: Distributed denial of service; LDoS: Low-rate denial of service; r/s: Requests per second; MF-DFA: Multifractal detrended fluctuation analysis; PCA: Principal pomponent analysis; SVM: Support vector machine; MLP: Multilayer perceptron; HTTP: Hyper text transfer protocol; FTP: File transfer protocol; TFD: Time domain & Frequency domain Based Detection, TFD; LSTM: Long Short Term Memory; FAR: False positive rate; SDN: Software Defined Network.

Authors' contributions
Yu Fu: Conceptualization; Investigation; writing –original draft. Xueyuan Duan: Supervision; Data Curation; formal analysis (lead); writing – review and editing. Kun Wang: Methodology; writing – review and editing. Bin Li: Software; Validation. The author(s) read and approved the final manuscript.

Funding
This work was supported by National Key Research and Development Program of China under Grant 2018YFB0804104.

Availability of data and materials
The datasets used or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author details
[1]Department of Information Security, Naval University of Engineering, Wuhan 430033, China. [2]College of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China. [3]Henan Key Laboratory of Analysis and Applications of Education Big Data, Xinyang 464000, China. [4]School of Mathematics and Information Engineering, Xinyang Vocational and Technical College, Xinyang 464000, China.

Received: 29 June 2022 Accepted: 5 August 2022
Published online: 30 August 2022

References
1. Adi E, Baig Z, Lam CP et al (2015) Low-rate denial-of-service attacks against HTTP/2 services. In: Proceedings of 2015 IEEE International Conference on IT Convergence & Security (ICITCS), pp 133–139
2. Wu ZHJ, Li WJ, Liu L et al (2020) Low-rate DoS attacks, detection, defense, and challenges: a survey. IEEE Access 8:43920–43943
3. Kurose JF, Ross KW (2021) Computer networking. A top-down approach, 8th edn. Pearson, New York
4. Manimurugan S, Almutairi S (2022) A user-based video recom-mendation approach using CAC filtering, PCA with LDOS-CoMoDa. J Supercomput 78:9377–9391
5. Luo XP, Chang RK (2005) On a new class of pulsing denial-of-service attacks and the defense. In: Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA
6. Guirguis M, Bestavros A, Matta I (2004) Exploiting the transients of adaptation for RoQ attacks on internet resources. In: Proceedings of the 12th IEEE International Conference on Network Protocols, ICNP 2004, Berlin, Germany, pp 184–195
7. Doshi R, Apthorpe N, Feamster N (2018) Machine learning ddos detection for consumer internet of things devices. In: Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, pp 29–35
8. Badshah A, Jalal A, Farooq U, Rehman GU, Band SS, Iwendi C (2022) Service level agreement monitoring as a service: an independent monitoring service for service level agreements in clouds, Big Data. Ahead of print https://doi.org/10.1089/big.2021.0274
9. Massimo F, Massimiliano R (2015) Stealthy denial of service strategy in cloud computing. IEEE Trans Cloud Comput 3(1):80–94
10. Wu ZJ, Zhang LY, Yue M (2015) Low-rate DoS attacks detection based on network multifractal. IEEE T rans Dependable Secur Comput 13:559–567
11. Xie S, Xing C, Zhang G et al (2019) Research on table overflow ldos attack detection and defense method in software defined networks. In: International Conference on Big Data and Security. Springer, Singapore, pp 80–97
12. Liu L, Wang HY, Wu ZHJ et al (2020) The detection method of low-rate DoS attack based on multi-feature fusion. Digit Commun Netw 6(4):504–513
13. He Z, Zhang T, Lee RB (2017) Machine learning based DDoS attack detection from source side in cloud. In: Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, pp 114–120
14. Kuzmanovic A, Knightly EW (2003) Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe Germany, pp 75–86
15. Kuzmanovic A, Knightly EW (2006) Low-rate TCP-targeted denial of service attacks and counter strategies. IEEE ACM Trans Netw 14(4):683–696 Karlsruhe, Germany
16. Tang D, Yan Y, Dai R et al (2022) A novel LDoS attack detection method based on reconstruction anomaly. Clust Comput 25:1373–1392
17. Jin C, Wang H, Shin K (2003) Hop-count filtering: an effective defense against spoofed DoS traffic. In: Proc. ACM CCS
18. Wu ZJ, Zhang JA, Yue M (2017) Approach of detecting low-rate DoS attack based on combined features. J Commun 38(5):19–30
19. Liu D, Shuai D (2003) Multifractal characteristic quantities of network traffic models. In: Proceedings of the International Conference on Grid and Cooperative Computing, Shanghai, China, pp 413–417
20. Zhang C, Cai Z, Chen W et al (2012) Flow level detection and filtering of low-rate DDoS. Comput Netw 56(15):3417–3431

21. Wu Z, Wang M, Yan C et al (2017) Low-rate DoS attack flows filtering based on frequency spectral analysis. China Commun 14(6):98–112
22. Zhang DSH, Tang D, Tang L, et al (2019) PCA-SVM-based approach of detecting low-rate dos attack. In: Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China. pp 1163–1170
23. Yan, Y, Tang D, Zhan S, et al (2019) Low-rate dos attack detection based on improved logistic regression. In: Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China. pp 468–476
24. Pérez-Díaz JA, Valdovinos IA, Choo KKR, Zhu D (2020) A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access 8:155859–155872
25. Du ZH, Ma LP, Sun GZ (2019) Network traffic anomaly detection based on wavelet analysis. Comput Sci 46(8):178–182
26. Agrawal N, Tapaswi S (2018) Low rate cloud DDoS attack defense method based on power spectral density analysis. Inf Process Lett 138:44–50
27. Brynielsson J, Sharma R (2015) Detectability of low-rate HTTP server DoS attacks using spectral analysis. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France, pp 954–961
28. Wu XX, Tang D, Tang L, et al (2018) A low-rate dos attack detection method based on hilbert spectrum and correlation. Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China. pp. 1358–1363
29. Swami R, Dave M, Ranga V (2019) Defending DDoS against software defined networks using entropy. In: Proceedings of 2019 IEEE 4th International Conference on Internet of Things: Smart Innovation and Usages, pp 1–5
30. Sharafaldin I, Lashkari AH, Hakak S et al (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: Proceedings of the International Carnahan Conference on Security Technology (ICCST), Chennai, India, pp 1–8
31. Kwon D, Natarajan K, Suh SC et al (2018) An empirical study on network anomaly detection using convolutional neural networks. In: ICDCS, pp 1595–1598
32. Bodström T, Hämäläinen T (2018) State of the art literature review on network anomaly detection with deep learning. In: Internet of things, smart spaces, and next generation networks and systems, pp 64–76
33. Veit A, Wilber M, Belongie S (2016) Residual networks behave like ensembles of relatively shallow networks. In: Advances in neural information processing systems, pp 550–558

Publisher's Note