Computers, Materials & Continua CMC, vol.61, no.2, pp.657-675, 2019

Novel DDoS Feature Representation Model Combining Deep Belief Network and Canonical Correlation Analysis

Chen Zhang[1], Jieren Cheng[1, 2, 3, *], Xiangyan Tang[1], Victor S. Sheng[4], Zhe Dong[1] and Junqi Li[1]

Abstract: Distributed denial of service (DDoS) attacks launch more and more frequently and are more destructive. Feature representation as an important part of DDoS defense technology directly affects the efficiency of defense. Most DDoS feature extraction methods cannot fully utilize the information of the original data, resulting in the extracted features losing useful features. In this paper, a DDoS feature representation method based on deep belief network (DBN) is proposed. We quantify the original data by the size of the network flows, the distribution of IP addresses and ports, and the diversity of packet sizes of different protocols and train the DBN in an unsupervised manner by these quantified values. Two feedforward neural networks (FFNN) are initialized by the trained deep belief network, and one of the feedforward neural networks continues to be trained in a supervised manner. The canonical correlation analysis (CCA) method is used to fuse the features extracted by two feedforward neural networks per layer. Experiments show that compared with other methods, the proposed method can extract better features.

Keywords: Deep belief network, DDoS feature representation, canonical correlation analysis.

1 Introduction

Nowadays, computer security is widely concerned, especially in the field of computer network, computer system and computer chip, etc. [Cai, Wang, Zheng et al. (2013); Liu, Cai, Xu et al. (2015); Xu, Wei, Zhang et al. (2018); Zhang, Tan, Liang et al. (2018); Lin, Yan, Huang et al. (2018); Lin, Li, Huang et al. (2018)]. In the field of network security, Distributed denial of service (DDoS) attack has become one of the serious threats to network security and has brought huge losses to society in recent years. The purpose of DDoS attack is to consume target host's resource and make the host lose the ability of providing service to normal users [Alharbi, Aljuhani and Liu (2017)]. In order to achieve this goal, attacker control botnet and then command the botnet to send meaningless packets to target host [Zhang, Zhang and Yu (2017)]. For reducing the damage of DDoS, many defense mechanisms have been put forward and the attack detection has become an

[1] College of Information Science & Technology, Hainan University, Haikou, 570228, China.

[2] State Key Laboratory of Marine Resource Utilization in South China Sea, Haikou, 570228, China.

[3] Key Laboratory of Internet Information Retrieval of Hainan Province, Hainan University, Haikou, China.

[4] Department of Computer Science, University of Central Arkansas, Conway, AR 72035, USA.

[*] Corresponding Author: Jieren Cheng. Email: cjr22@163.com.

important part of these mechanisms [Cheng, Xu, Tang et al. (2018)]. DDoS attack detection is able to recognize normal and attack flow in the network flow so that let the server take the defense measures as soon as possible. Jiao et al. [Jiao, Ye, Zhao et al. (2017)] DDoS attack detection is mainly divided into two parts, the first part is feature extraction and second part is detection model. Feature extraction is used to get the difference of normal and attack flow by analyzing behavior between them and detection model takes this difference as input so as to determine the type of data. Liao et al. [Liao, Li, Kang et al. (2015)]. The difficulties of feature extraction are as follows: (1) The source IP address can be forged, which makes it difficult for attackers to be found; (2) There are many types of DDoS attack and each type has different characteristics; (3) The DDoS attack has burstiness, so it not only needs an keen method but also a method with comprehensive information [Kolias, Kambourakis, Stavrou et al. (2017)]. With the development of the cloud computing, Internet of things and big data, the types of attack have become more various, the scale of network data has also been greatly increased and the rule of DDoS attack has been more deeply hidden [Somani, Gaur, Sanghi et al. (2016)]. It makes traditional detection model unable to detect DDoS attacks effectively. These difficulties of feature extraction will be even more difficult to solve. Meanwhile the ineffectiveness of traditional detection models will also bring greater challenges.

In recent years, the method of extracting features from multiple protocols has emerged and it is proved that multiple protocols can effectively improve the accuracy of DDoS attack detection under the current network environment. So, the use of multiple protocol is the way to improve the accuracy of DDoS attack detection. And deep learning has been widely applied to various fields successfully. Deep learning is a technique which explores pattern hidden in data, and focuses on implicit information in data rather than reasons for data occurrence. If the data scale is enough, deep learning can learn a good pattern and obtain great classification or regression results about unknown data by using the pattern. Therefore, deep learning can be applied to DDoS attack detection model in the occasion of large scale data.

Feature extraction is an important part of DDoS defense. Most methods extract features by using statistics and filtering rules. However, these methods often rely on the experience of researchers and are easy to introduce artificial errors. In addition, artificially selecting features tend to lose the potential characteristics of the original data and lose the relationship between data attributes. Because deep learning not only has the ability to mine the potential features of data, but also identify the differences between different categories of data, in recent years, deep learning has been applied to various fields. The field of network security also attaches great importance to the application of deep learning [Niyaz, Javaid, Sun et al. (2016); Li, Wu, Yuan et al. (2018); Yuan, Li and Li (2017)]. In the field of network security, some researchers have applied deep belief network (DBN) to intrusion detection [Alom, Bontupalli and Taha (2015)]. These findings shows that it is feasible to apply deep learning to attack detection. In order to explore deep learning applied to DDoS feature extraction, this paper proposes a DDoS attack feature representation method based on deep belief network. Using the growth of network flows, the distribution of packet addresses and ports, and the diversity of packet sizes of different protocols, we quantify the packets in each sampling time. These

quantized values are used to train deep belief networks in an unsupervised manner. In order to enrich the expression of features, we initialize two feedforward neural networks (FFNN) with a trained DBN and one of the neural networks is trained in a supervised manner. These two feedforward neural networks can extract features from input data. In order to effectively fuse the features of the two feedforward neural networks, we use the canonical correlation analysis (CCA) method to fuse the two neural network mapped features layer by layer.

2 Related work

In recent years, researchers have made a lot of contributions and efforts to reduce the damage of DDoS attacks. According to different research directions, these contributions can be divided into two sorts: the attack detection method of DDoS and the defense mechanism of DDoS.

(1) Attack detection can be used to distinguish between normal flows and attack flows. In this field, Rukavitsyn et al. [Rukavitsyn, Borisenko and Shorov (2017)] proposed a Self-learning method for DDoS detection model in cloud computing; Hsieh et al. [Hsieh and Chan (2016)] proposed DDoS detection method based on Neural Networks; Zhu et al. [Zhu, Tang, Shen et al. (2018)] proposed a privacy-preserving cross-domain attack detection scheme for SDNs; Idhammad et al. [Idhammad, Afdel and Belouch (2018)] proposed an online sequential semi-supervised ML approach for DDoS detection based on network Entropy estimation, Co-clustering, Information Gain Ratio and Exra-Trees algorithm; Wang et al. [Wang, Du, Sun et al. (2016)] put forward the hybrid attack detection and forensics model in M2M networks; Saied et al. [Saied, Overill and Radzik (2016)] proposed an Artificial Neural Network (ANN) algorithm to detect DDoS attacks based on specific characteristic features (patterns); Arivudainambi et al. [Arivudainambi, Kumar and Sibi (2018)] proposed an effective and accurate DDoS detection method using Lion optimization algorithm; Seo et al. [Seo and Lee (2016)] proposed a method for the effective detection of malware infection systems triggering IP-spoofed DDoS attacks on an edge network; Nezhad et al. [Nezhad, Nazari and Gharavol (2016)] proposed a DDoS attacks detection algorithm using ARIMA Time Series Model and Chaotic System in computer networks.

(2) The purpose of the defense mechanism is to reduce the damage of the attack, or even completely eliminate the attack. In this field, Rajarajan et al. [Rajarajan and Ganesan (2017)] proposed an agent based honeymesh for protecting the network resources like servers from intrusion related attacks; Zhang et al. [Zhang, Wang, Perrig et al. (2016)] proposed a flooding attack defense mechanism named Tumbler; Roberto et al. proposed a novel abstraction of the recursive DNS traffic to detect a flooding attack, which is a kind of Distributed Denial of Service [Alonso, Monroy and Trejo (2016)]; Malialis et al. [Malialis and Kudenko (2015)] proposed a novel design to the original Multiagent Router Throttling approach that it provides a decentralized coordinated response to the DDoS problem; Shiaeles et al. [Shiaeles and Papadaki (2014)] proposed an improved IP spoof detection method for web DDoS attacks; Li et al. [Li, Kao, Zhang et al. (2015)] proposed a network behavior-based botnet detection mechanism using PSO and K-means; Kumarasamy [Kumarasamy (2009)] proposed an effective defense mechanism for

Distributed Denial-of-Service (DDoS) attacks using router-based techniques.

All in all, feature representation plays an important role in DDoS defense. However, the current DDoS attack feature representation method often relies on manual experience, and it is easy to lose important information in the original data.

Since the deep learning method can better mine the information in the original data, we propose a DDoS feature representation method based on deep belief network. We quantify the raw data using network data traffic, the distribution of IP addresses, and changes in packet size between different protocols and convert these quantized values into equal-length binary values. The deep belief network is trained by equal-length binary, and then the two deep feedforward neural networks are initialized with the trained deep belief network. One of the feedforward neural networks no longer performs any training, and the other continues to train in a supervised manner. These two feedforward neural networks constitute the feature extractor. To fuse the features extracted by two feature extractors, the CCA method is used to fuse features extracted from each layer of two neural networks. The experimental results show that our proposed method can better represent DDoS attack characteristics than other statistical methods.

3 Network data preprocessing

Packets collected on the network cannot be directly analyzed, so we need to preprocess these packets. In the process, we fully considered the distribution of DDoS attack, the diversity of packet size and time sequence relationship between sampling points. In addition, to reduce the time spent on subsequent data analysis and to make full use of each attribute of the network packet, the value of each sample point is calculated from multiple attributes of the network packet. The details of the network data preprocessing are as follows.

The network data attributes are described as: $T = (time, sip, dip, protocol, sport, dport, size)$. The *time*, the *sip*, the *dip*, the *protocol*, the *sport*, the *dport* and the *size* represent the arrival time of packet, the source IP address of packet, the destination IP address of packet, the protocol of packet, the source port of packet, the destination port of packet and the size of packet respectively. In each sampling period *t*, we do the following processing on the network data:

The number of packets *s* in each period of sampling time *t* is counted and *s* is calculated by the following formula:

$$spackt_t = \lceil \log_2 s \rceil \tag{1}$$

The symbol $\lceil . \rceil$ indicates rounding up. By this way, we can speed up the data processing process by storing the number of packets per sample time with less binary code.

In each period of sampling time *t*, the number of source IP addresses *ssip* are divided by the number of destination IP addresses *sdip* and the number of source port *ssport* are divided by the number of destination ports *sdport*. The details of the formulation are as follows:

$$\begin{cases} divip_t = \left\lceil \dfrac{ssip}{sdip} \right\rceil \\ divport_t = \left\lceil \dfrac{ssport}{sdport} \right\rceil \end{cases} \tag{2}$$

The $divip_t$ can show the difference between normal flow and attack flow in the distribution of source IP addresses and destination IP addresses. And $divport_t$ can show the difference between normal flow and attack flow in the distribution of source ports and destination ports.

In each period of sampling time t, the number of types of packet size for each protocol are calculated by the following formula:

$$tps_t = \left\lceil \frac{\sum_{i=1}^{n} tps_i^2}{\sum_{i=1}^{n} tps_i} \right\rceil \tag{3}$$

where, tps_i represents the number of types of packet size for each protocol and n represents the number of types of protocol.

The tps_t can adaptively adjust the impact of packet size of each protocol and comprehensively calculate the number of the number of types of packet size of the current sampling point according to the influence of each protocol. In this way, compared to other protocols, the protocol that plays the main role of the attack can be amplified, making the attack data more recognizable.

Since $spackt_t$, $divip_t$, $divport_t$ and tps_t can express the difference between normal flow and attack flow from multiple aspects, we use these four quantized values as initial features for subsequent feature expression.

In order to use the time sequence relationship of sample point fully, we calculate the current quantized values by combining the quantized values before the current sampling time. The details of the formulation are as follows:

$$sp_t(x) = \left\lceil \frac{w_{t-n}x_{t-n} + \cdots w_{t-1}x_{t-1} + w_t x_t}{w_{t-n} + \cdots w_{t-1} + w_t} \right\rceil \tag{4}$$

where, x_t represents the quantized value in current sampling time, w_t represents the weight value in current sampling time and n represents the number of quantized values participating in the current sample point calculation before the current sampling time.

To facilitate neural network processing, we convert the value of each quantized values to binary. And since the input of the neural network are of equal length, we convert each attribute of the binary data into a uniform length according to the maximum length of the attribute.

4 Model of deep belief network

Deep belief network consists of multiple Boltzmann machines and multiple Boltzmann machines are stacked in a layered structure. Unlike other back propagation methods used

in other neural network training, the deep belief network adopts a layer-by-layer training approach. Specifically, first, the bottom-layer Boltzmann machine is fully trained, and then the next-layer Boltzmann machine is trained with the trained Boltzmann machine. Follow the above training method, until all the Boltzmann machine in the network is fully trained. Because the deep belief network uses layer-by-layer training, it has a faster training speed. In addition, the deep belief network is trained in an unsupervised manner, thus having lower requirements on data sets. Deep belief network mainly includes the following basic structure:

$\mathrm{v} = (v_1, v_2, \cdots, v_{nv})^T$ represents the data vector of the visible layer, this vector is the input data. $\mathrm{h} = (h_1, h_1, \cdots, h_{nh})^T$ represents the data vector of the hidden layer, this vector is a feature extracted by the Boltzmann machine.

$W = (w_{i,j}) \in R^{nv \times nh}$ represents the connection weight of each node in the adjacent two-layer Boltzmann machine.

$\mathrm{o} = (o_1, o_2, \cdots, o_{nv})^T$ represents the output vector, which can be used as the feature of the original data mapped by the deep belief network.

Deep belief network has good feature representation ability, and it uses layer-by-layer training method, so the trained parameters have local characteristics and can be used to initialize other neural networks, so that other neural networks also have good feature representation ability after fine-tuning. Therefore, we choose deep belief network as the feature representation model. The framework of the deep belief network is shown in Fig. 1.

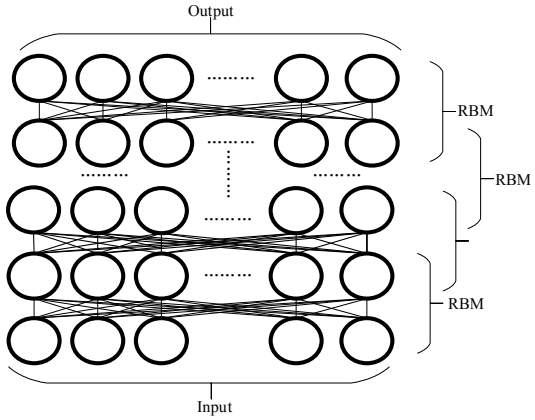

Figure 1: The framework of the deep belief network

5 A novel feature representation framework

5.1 Feed forward neural network trained using deep belief network

Currently, there are two ways to extract data features using the deep belief network. (1) Firstly, the deep belief network is trained in an unsupervised manner, then a feedforward neural network is initialized with the trained deep belief network and all hidden layers of the feedforward neural network are used as feature extractor. (2) Because the deep belief network adopts a layer-by-layer greedy approach, it lacks awareness of global

information. So, some methods use a deep belief network trained in an unsupervised manner to initialize the feedforward neural network, and then use the same data set to train the feedforward neural network in a supervised manner, and use the hidden layer of the feedforward neural network as feature extractor.

The first way can be seen as the feature of the data in an unsupervised form. This form of feature representation is more ambiguous and abstract because there is no instructional training method. The second method reflects the feature of data representation under the form of supervised learning. The representation of this feature is clearer and more precise due to the use of labels for instructional learning. Since the expression of data features is complex, some data needs to be more abstract and vaguer expressions so that the data can be regarded as a class, and some data needs to be clearer and more precise so that the data can be regarded as different classes. Therefore, in order to effectively represent the data, we use these two feature extraction methods as modules for preliminary extraction of data features. We record the feedforward neural network created by method (1) in an unsupervised manner as Unsupervised-FFNN and the feedforward neural network trained by method (2) in supervised learning as Supervised-FFNN. The model for preliminary representation of feature shown in Fig. 2:

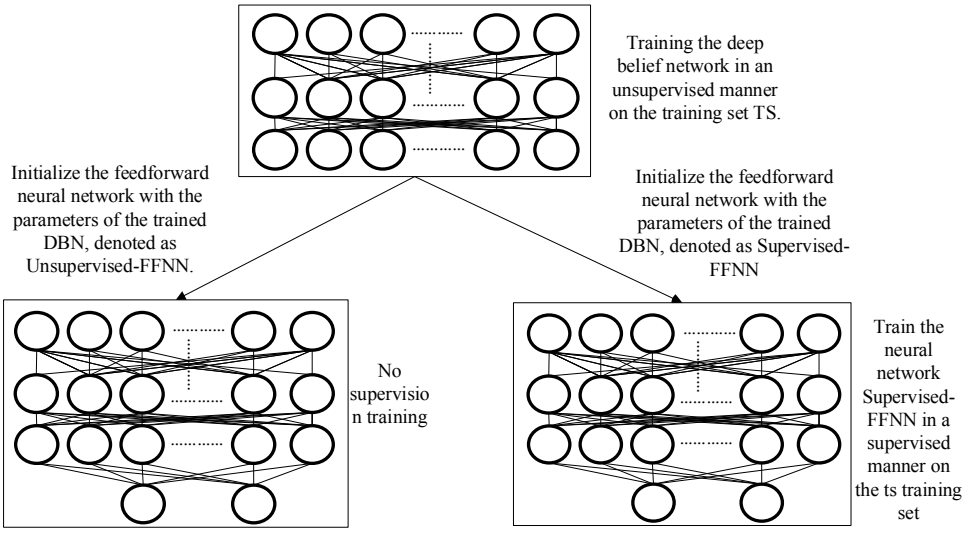

Figure 2: The model for preliminary representation of feature

5.2 Multi-layer feature extraction method using CCA

The features matrix of the data to be extracted are input to the two neural networks to obtain the neural network feature expression matrix corresponding to the feature. The canonical correlation analysis can measure the similarity of two matrices and similar matrices can often be fused. The features extracted by the two neural networks, namely Unsupervised-FFNN and Supervised-FFNN can be seen as the way to observe the data at different angles. Through the CCA method, we can effectively combine the two features

of the two angle observations based on correlation, thus enriching the expression of features. In addition, the upper layers of the feature extracted by the neural network are often abstract, and the expression of the lower features is often detailed. Therefore, the use of multi-level expressions can reflect the characteristics of the features from abstraction to refinement, thus further enriching the expression of features. In order to achieve multi-angle and multi-level expression of features, we use the CCA method to fuse the two feature matrices mapped of the two neural network layer by layer. Each feature after fusion is used as an attribute of the feature. Therefore, the number of attributes owned by each feature is equal to the number of layers of the neural network.

We introduce the calculation method of any layer multi-angle multi-layer feature expression. The details of the calculation method are as follows:

The matrix formed by mapping the samples of the feature to be extracted through each layer of the Unsupervised-FFNN is recorded as $f_{layer}\left[f_1 f_2 \cdots f_l\right]$ and the matrix formed by mapping the samples of the feature to be extracted through each layer of the Supervised-FFNN is recorded as $f_{layer}{}'\left[f_1{}' f_2{}' \cdots f_l{}'\right]$. Where, l indicates the number of layers of the hidden layer of the neural network. f_i represents the feature matrix of all the samples of the *i-th* layer after being mapped by the Unsupervised-FFNN and f_i' represents the feature matrix of all the samples of the *i-th* layer after being mapped by the Supervised-FFNN.

The f_i is recorded as $f_i = \begin{bmatrix} f_{i,11} & f_{i,12} \cdots f_{i,1m} \\ f_{i,21} & f_{i,22} \cdots f_{i,2m} \\ \vdots & \vdots \ \vdots \ \vdots \\ f_{i,n1} & f_{i,n2} \cdots f_{i,nm} \end{bmatrix}$ and the fi' is recorded as $f_i{}' = \begin{bmatrix} f_{i,11}{}' & f_{i,12}{}' \cdots f_{i,1m}{}' \\ f_{i,21}{}' & f_{i,22}{}' \cdots f_{i,2m}{}' \\ \vdots & \vdots \ \vdots \ \vdots \\ f_{i,n1}{}' & f_{i,n2}{}' \cdots f_{i,nn}{}' \end{bmatrix}$.

where, i represents the *i-th* layer of the neural network, n indicates the number of feature samples to be extracted and m represents the dimension of the sample.

In CCA method, the *i-th* layer of the correlation of f_i and $f_i{}'$ can be described as:

$$\max_{U,V} \rho_{U,V} = corr(U,V) = \frac{\mathrm{cov}(U,V)}{\sigma_U \sigma_V} \tag{5}$$

where $\begin{cases} U_i = \begin{bmatrix} \alpha_{i,11} f_{i,11} + \alpha_{i,12} f_{i,12} + \cdots + \alpha_{i,1m} f_{i,1m} \\ \alpha_{i,21} f_{i,21} + \alpha_{i,22} f_{i,22} + \cdots + \alpha_{i,2m} f_{i,2m} \\ \cdots\cdots \\ \alpha_{i,n1} f_{i,n1} + \alpha_{i,n2} f_{i,n2} + \cdots + \alpha_{i,nm} f_{i,nm} \end{bmatrix} \\ V_i = \begin{bmatrix} \alpha_{i,11}{}' f_{i,11}{}' + \alpha_{i,12}{}' f_{i,12}{}' + \cdots + \alpha_{i,1m}{}' f_{i,1m}{}' \\ \alpha_{i,21}{}' f_{i,21}{}' + \alpha_{i,22}{}' f_{i,22}{}' + \cdots + \alpha_{i,2m}{}' f_{i,2m}{}' \\ \cdots\cdots \\ \alpha_{i,n1}{}' f_{i,n1}{}' + \alpha_{i,n2}{}' f_{i,n2}{}' + \cdots + \alpha_{i,nm}{}' f_{i,nm}{}' \end{bmatrix} \end{cases}$

The canonical coefficients are obtained after CCA is finished. The canonical coefficients matrix of A and B is recorded as

$$A_i = \begin{bmatrix} \alpha_{i,11}\ \alpha_{i,12} \cdots\ \alpha_{i,c} \\ \alpha_{i,21}\ \alpha_{i,22} \cdots\ \alpha_{i,2c} \\ \vdots\quad \vdots\ \vdots\ \vdots \\ \alpha_{i,m1}\ \alpha_{i,m2} \cdots\ \alpha_{i,mc} \end{bmatrix} \quad B_i = \begin{bmatrix} \alpha_{i,11}'\ \alpha_{i,12}' \cdots\ \alpha_{i,1c}' \\ \alpha_{i,21}'\ \alpha_{i,22}' \cdots\ \alpha_{i,2c}' \\ \vdots\quad \vdots\ \vdots\ \vdots \\ \alpha_{i,m1}'\ \alpha_{i,m2}' \cdots\ \alpha_{i,mc}' \end{bmatrix}$$

where, c represents the rank of the feature matrices f and f'. The canonical variables matrix of U and V is recorded as

$$SU_i = \begin{bmatrix} f_{i,11} - \overline{x_{i,1}}\ f_{i,12} - \overline{x_{i,1}}\ f_{i,1m} - \overline{x_{i,1}} \\ f_{i,21} - \overline{x_{i,2}}\ f_{i,22} - \overline{x}_{i,2}\ f_{i,2m} - \overline{x}_{i,2} \\ \cdots\cdots \\ f_{i,n1} - \overline{x_{i,n}}\ f_{i,n2} - \overline{x_{i,n}}\ f_{i,nm} - \overline{x_{i,n}} \end{bmatrix} A_i \quad SV_i = \begin{bmatrix} f_{i,11}' - \overline{x_{i,1}'}\ f_{i,12}' - \overline{x_{i,1}'}\ f_{i,1m}' - \overline{x_{i,1}'} \\ f_{i,21}' - \overline{x_{i,2}'}\ f_{i,22}' - \overline{x_{i,2}'}\ f_{i,2m}' - \overline{x_{i,2}'} \\ \cdots\cdots \\ f_{i,n1}' - \overline{x_{i,n}'}\ f_{i,n2}' - \overline{x_{i,n}'}\ f_{i,nm}' - \overline{x_{i,n}'} \end{bmatrix} B_i$$

where

$$\overline{x_{i,j}} = \frac{1}{n+1}\sum_{i=1}^{n} f_{i,ji}\ ,\ \overline{x_{i,j}'} = \frac{1}{n+1}\sum_{i=1}^{n} f_{i,ji}'$$

The correlation vector is recorded as $R_i = \begin{bmatrix} r_{i,1}\ r_{i,2} \cdots\ r_{i,c} \end{bmatrix}$.

The correlation obtained by the CCA method reflects the distance of two different neural network extracted features in one dimensional space. The distance between the features representing the same object is small and the distance between the features representing the different objects is large. Therefore, in order to effectively fuse the features of the same object, we only fuse features with a correlation greater than or equal to "0.8". For facilitating the calculation, we reconstruct the R_i matrix into the following form:

$$SR_i = \begin{bmatrix} r_{i,1}\ r_{i,1} \cdots\ r_{i,1} \\ r_{i,2}\ r_{i,2} \cdots\ r_{i,2} \\ \cdots\cdots \\ r_{i,c}\ r_{i,c} \cdots\ r_{i,c} \end{bmatrix}. \text{ Where } \begin{cases} r_{i,h} = r_{i,h}\ if\ r_{i,h} >= 0.8 \\ r_{i,h} = 0 \quad if\ r_{i,h} < 0.8 \end{cases} \tag{6}$$

The canonical variables indicates that two feature matrices f_i and f_i' are mapped to values in one-dimensional space by different canonical coefficients and each row of a canonical variables represents a different representation of a sample point that is mapped by a different canonical coefficients. And because the correlation coefficient reflects the similarity of the two features, the features with large correlation should be considered more closely. In order to fuse the two features and reflect the different effects of different canonical variables on the value of fusion, first we add up the canonical variables in matrices SU_i and SV_i. Then we multiply the added values by the reconstruction correlation coefficient SR_i. Finally, we linearly add up the multiplied values by row. The formula is as follows:

$$ff_i = SR_i(SV_i + SU_i)I_{n\times 1} \tag{7}$$

Where, $I_{n\times 1} = \begin{bmatrix} 1\ 1 \cdots 1 \end{bmatrix}^T$.

The method of multi-angle multi-layer feature calculation in the remaining layers is the same as above. The feature multi-angle multi-level extraction model shown in Fig. 3.

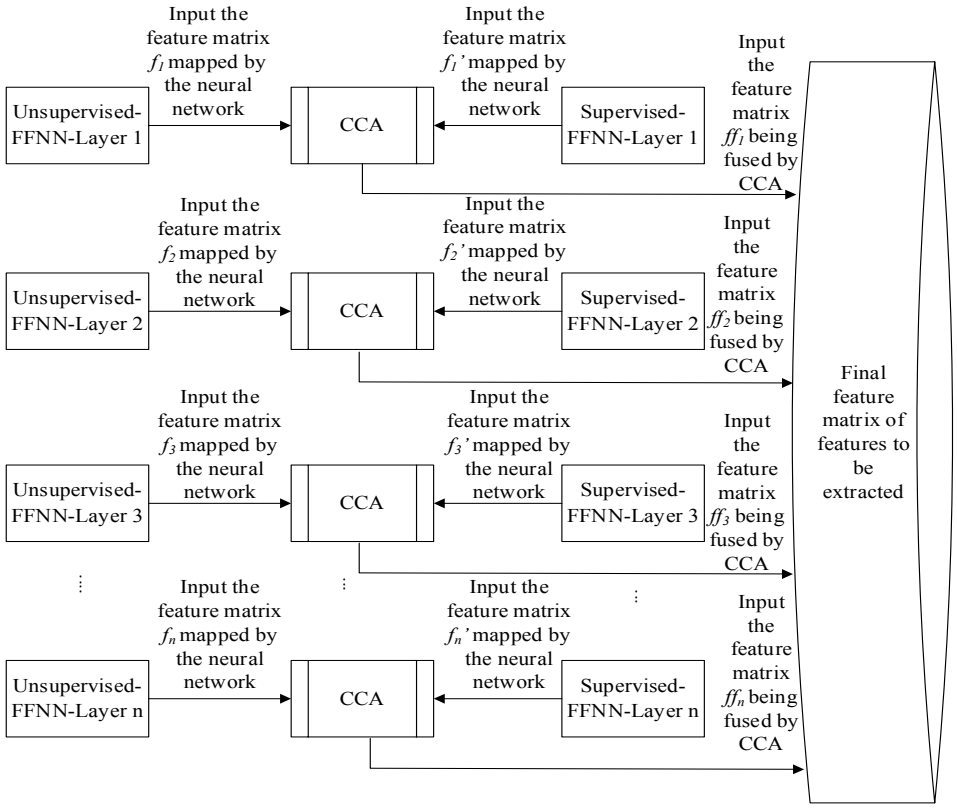

Figure 3: The model for multi-angle multi-level feature extraction

6 Experiment

6.1 The environment and evaluation of experiment

The data set used in the experiment is the CAIDA "DDoS Attack 2007". This data set collects nearly one hour of network packets (20:50:08 UTC-21:56:16 UTC) and the attack launched at 21:13. The experimental environment we use are MATLAB 2014a, Wireshark 2.2.1 and Java1.8. The evaluation criteria we use are the detection rate (DR), the false alarm rate (FR) and total error rate (ER). These evaluation criteria are calculated as follows:

$$\begin{cases} DR = \dfrac{TN}{TN + FN} \\ FR = \dfrac{FP}{TP + FP} \\ ER = \dfrac{FN + FP}{TP + FP + TN + FN} \end{cases} \tag{8}$$

where, TP represents that the number of normal test samples which is correctly identified, FP represents the number of normal test samples which is incorrectly identified, TN represents the number of attack test samples which is correctly identified and FN represents the number of attack test samples which is incorrectly identified.

6.2 Experiment and analysis

We set the sampling time to 0.1s and collect a total of 39,107 quantized values through data preprocessing. And attack flow is divided into early attack stage and attack peak stage. The number of DBN and FFNN hidden layers is set to half the number of input layer nodes, and the number of nodes per layer is the same as the number of input layer nodes. Seventy percents of early attack flows, attack peak flows and normal flows are used to train DBN in unsupervised way and the Supervised-FFNN is trained with the same data in a supervised manner. The remaining 30% of the data is used to extract features and features extracted from these data constitute feature sets. The n of the formula 4 is set to "10" and the weight w_t is set to "1", the weight w_{t-1} is set to "0.9", the weight w_{t-2} is set to "0.8", the weight w_{t-3} is set to "0.7", the weight w_{t-4} is set to "0.6", the weight w_{t-5} is set to "0.5", the weight w_{t-6} is set to "0.4", the weight w_{t-7} is set to "0.3", the weight w_{t-8} is set to "0.2", the weight w_{t-9} is set to "0.1".

In order to verify the performance of the proposed method representation features, we designed a feature comparison experiment, a classification experiment, and a classification experiment with feature fluctuation. The methods in the literature [Nezhad, Nazari and Gharavol (2016); Chen, Ma and Wu (2013)] are used as comparative experiment in each experiment.

In the feature comparison experiment, since the compared methods are all one-dimensional features, we use principal component analysis (PCA) to convert the features of our method into one dimension and then compare the features of the other two methods. The figures of the features of the three methods are shown in Figs. 4-6.

Figs. 4-6 show that the features extracted by proposed method are more compact and can represent the difference between normal flow and attack flow earlier than the other two methods. These compact features reduce the amount of data that is misjudged, and earlier representations of normal and attack flows can improve the accuracy of early attack identification. Next, we specifically analyze the reasons why our proposed method is better than the other two methods.

In terms of tightness, since the output value of each layer of the neural network is gradually changed, the values of each dimension of the extracted features are not greatly different so that the distribution of the extracted features processed by the PCA method is relatively tight by the PCA method. In addition, in the data preprocessing process, we use

time sequence relationship of the sample points, which also promotes the tightness of the resulting features. Early attacks have very similar characteristics to normal flows, which makes features require detailed expressions and multidimensional information. In order to reduce the probability of being misjudged, features also need to describe the same type of data with more ambiguous information. Therefore, this paper uses a supervised way to describe the features of the data in more detail, and uses an unsupervised way to better classify the same kind of data in a fuzzy way, and each data is expressed as a high-dimensional feature with multiple layers of information. However, the other two features use fluctuating network flows, so the data distribution is not compact enough and the two features use only one-dimensional features and thus are insufficient for the expression of attack flows and normal flows.

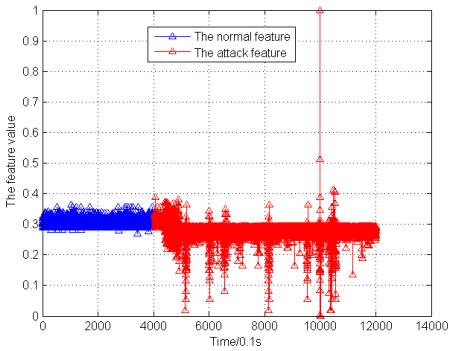

Figure 4: The feature of proposed method in 0.1 sampling time

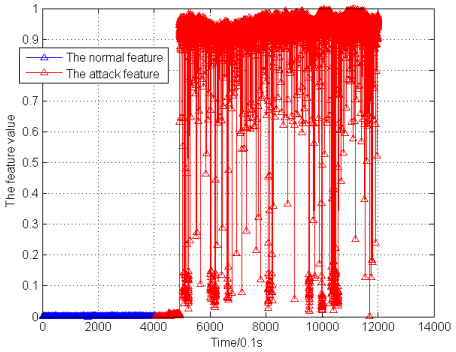

Figure 5: The feature of Nezhad et al.'s method in 0.1 sampling time

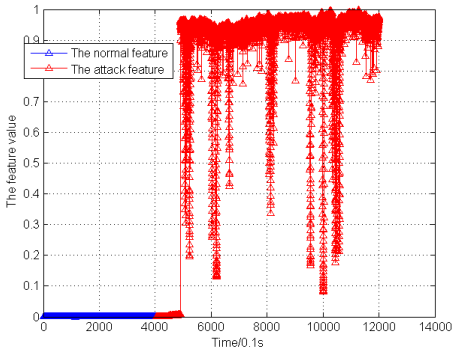

Figure 6: The feature of Chen et al.'s method in 0.1 sampling time

The features that two contrast methods extract and the features our method proposed are in the same period of time.

In classification experiment, we use SVM and feedforward neural networks to classify the features of the three methods. In the feature set, 70% of the early attack features, attack peak features and normal flow features constitute the training set, and the

remaining 30% are test sets. The results of classifying the features of the three methods using two classifiers are shown in Tab. 1, Tab. 2.

It can be seen from Tab. 1 and Tab. 2 that SVM and FFNN can more accurately identify the attack flow by using the features extracted in this paper. However, FFNN uses the features of this paper to identify normal streams with lower accuracy than the other two methods.

The SVM classifies the samples based on the maximum interval between the two types of samples, so the greater the distance between the two types of samples, the more helpful the accuracy of the identification. It can be seen from Figs. 4-6 that the method of this paper can reflect the difference between attack flow and normal flow earlier, so that the distance between the early attack flow and the normal flow becomes larger which is more conducive to SVM judgment, thus improving the early detection of attack flow. Since the FFNN also discriminates the network flow according to the difference between the normal flow and the attack flow, the feature of identifying the attack flow earlier is more favorable for the FFNN classification. However, in the training set, the number of samples of the attack flows is larger than the number of normal flows samples, the FFNN is trained to be more biased toward the attack flow, excessively dividing the boundary to the normal flow side, resulting in a small number of normal flows are identified as attacks.

Table1: The evaluation of three feature representation methods for SVM in 0.1 s sampling time

	Nezhad et al.'s method	Chen et al.'s method	The proposed method
DR	0.8056	0.8092	0.912
FR	0	0	0
ER	0.130997305	0.128571429	0.059

Table2: The evaluation of three feature representation methods for FFNN in 0.1 s sampling time

	Nezhad et al.'s method	Chen et al.'s method	The proposed method
DR	0.762	0.8052	0.9816
FR	0	0	0.0379
ER	0.1604	0.1313	0.0571

In classification experiment with feature fluctuations, to verify the robustness of the three features, we randomly amplify the normal flow features of the three methods. Random amplification of low magnification from 1 to 2 times to 1 to 11 times. High magnification multiple random amplification from 1 to 12 times to 1 to 21 times. The results of SVM and FFNN classification at each stage are shown in Figs. 9-16.

It can be seen from Figs. 7-9 that SVM uses this feature to identify the attack flow more accurately, but the accuracy of identifying normal flow is lower than the other two methods. Although the features extracted by Nezhad et al.'s method and Chen et al.'s method cannot show the difference between the attack flow and the normal flow earlier, the difference value between the attack flow and the normal flow exhibited by the two methods is often large. Therefore, random amplification of normal flow using low magnification has little effect on the distribution of features, so it does not affect the

classification results of SVM. In the same way, the method of this paper also conforms to this rule in most cases. However, the normal flow features extracted by the method in this paper are very close, and the values are relatively small and within a certain range. After random amplification of the normal flow, the value of a part of the normal flow can be as far away from other values of the same category, and this situation only appear in the attack flow. The value of this phenomenon might often be judged as the attack flow. Therefore, some of the normal flow features of the method in this paper are misjudged.

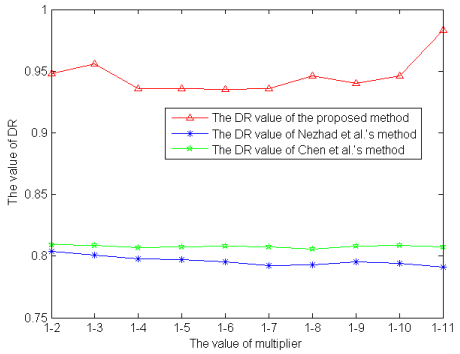

Figure 7: The DR values of three features classified by SVM in 0.1 s sampling time

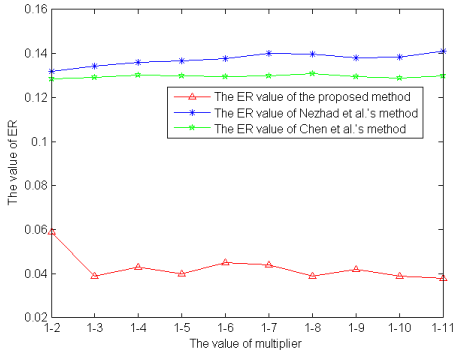

Figure 8: The ER values of three features classified by SVM in 0.1 s sampling time

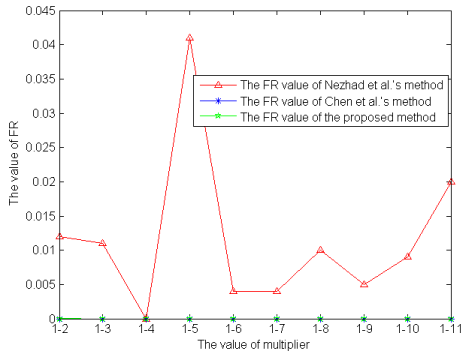

Figure 9: The FR values of three features classified by SVM in 0.1s sampling time

It can be seen from Figs. 10-15 that in the case of FFNN classification, the method proposed in this paper is not as good as the other two methods in the low magnification random amplification experiment, but in the high magnification random amplification experiment, the method is superior to other methods. It can be seen from Figs. 10-12 that FFNN is more susceptible to interference in the case of low-magnification random amplification of normal flow, but as the multiple increases, the performance of FFNN gradually improves. This is because the neural network is very sensitive to abnormal data, and some normal flows processed by random fluctuations exhibit very similar characteristics to the abnormal data, which easily break the pattern that the neural network has learned, resulting in the neural network not recognizing part of the attack

flow and the normal flow. Therefore, in the case of random amplification of normal flow low magnification, FFNN performs poorly. But, it can be seen from Figs. 13-15 that as the magnification increases, the volatility of the data increases. The neural network gradually learns the local volatility of the features of the proposed method, and the performance is getting better and better. However, under high-magnification random amplification, some of the normal flow features of Nezhad et al.'s method and Chen et al.'s method is higher than some of the attack flow features, thus more and more normal flows are recognized incorrectly. Therefore, this experiment can show that the features extracted by the method have better performance under severe data fluctuations.

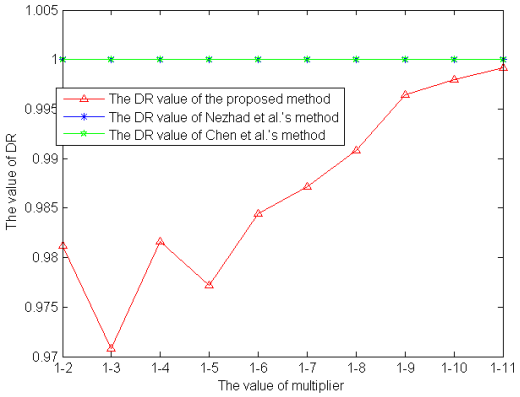

Figure 10: Under low magnification random amplification, the DR values of three features classified by FFNN in 0.1 s sampling time

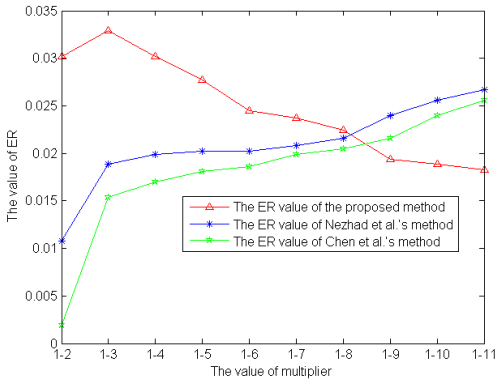

Figure 11: Under low magnification random amplification, the ER values of three features classified by FFNN in 0.1 s sampling time

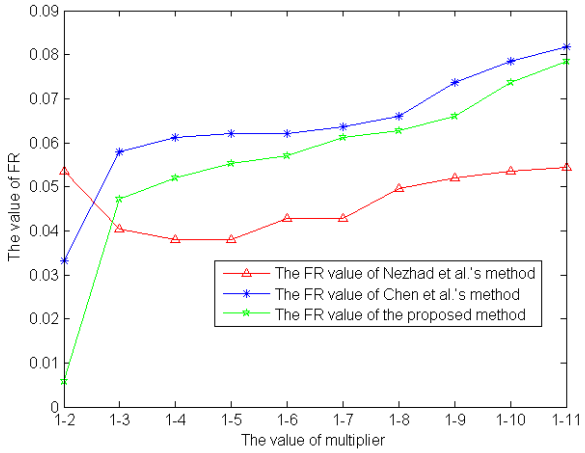

Figure 12: Under low magnification random amplification, the FR values of three features classified by FFNN in 0.1 s sampling time

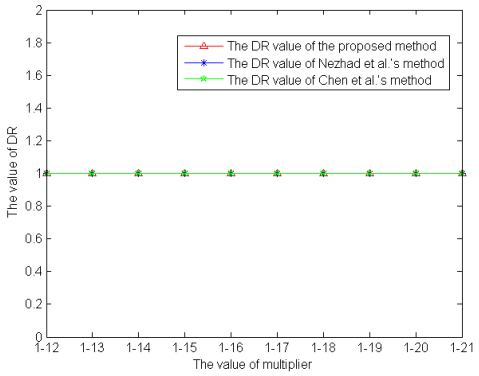

Figure 13: Under high magnification random amplification, the DR values of three features classified by FFNN in 0.1 s sampling time

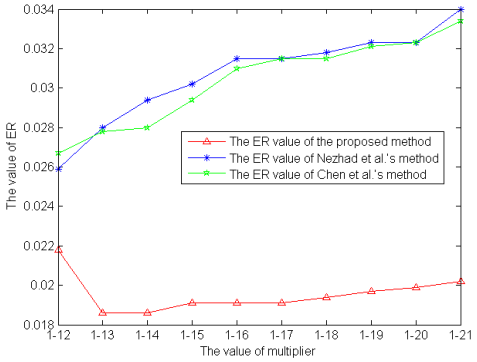

Figure 14: Under high magnification random amplification, the ER values of three features classified by FFNN in 0.1 s sampling time

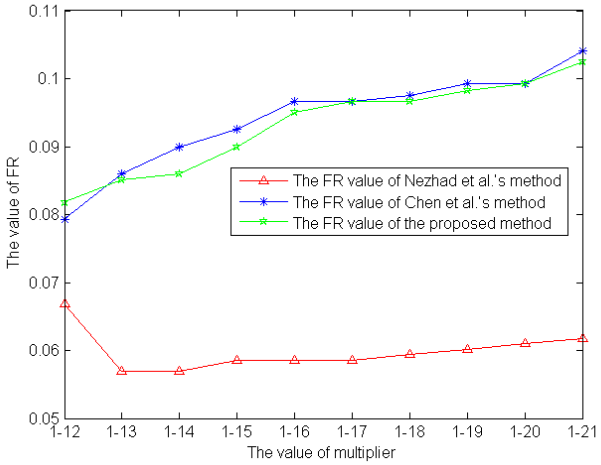

Figure 15: Under high magnification random amplification, the FR values of three features classified by FFNN in 0.1 s sampling time

7 Conclusions and future work

In this paper, we propose a new DDoS feature representation method. We use the most typical characteristics of the DDoS attack to quantify the data as much as possible to retain the information of original data. In order to extract the features of DDoS attacks from multiple angles, we train the deep belief network in an unsupervised manner and initialize two feedforward neural networks, one of which continues to be trained in a supervised manner. Since the features extracted by each layer of the neural network have different levels of abstraction, to further enrich the expression of features, we use the CCA method to fuse the features extracted by the two neural networks by layer.

Next, we will continue to explore ways to improve the performance of normal flow features of this paper under low magnification random amplification and represent better DDoS attack features using other deep learning models.

Acknowledgement: This work was supported by the National Natural Science Foundation of Hainan (2018CXTD333, 617048); National Natural Science Foundation of China (61762033, 61702539); The National Natural Science Foundation of Hunan (2018JJ3611); Social Development Project of Public Welfare Technology Application of Zhejiang Province (LGF18F020019); Hainan University Doctor Start Fund Project (kyqd1328); Hainan University Youth Fund Project (qnjj1444); State Key Laboratory of Marine Resource Utilization in South China Sea Funding.

References

Alharbi, T.; Aljuhani, A.; Liu, H. (2017): Holistic DDos mitigation using NFV. *IEEE 7th Annual Computing and Communication Workshop and Conference*, pp. 1-4.

Alom, M. Z.; Bontupalli, V.; Taha, T. M. (2015): Intrusion detection using deep belief networks *National Aerospace and Electronics Conference*, pp. 339-344.

Alonso, R.; Monroy, R.; Trejo, L. A. (2016): Mining IP to domain name interactions to detect DNS flood attacks on recursive DNS servers. *Sensors*, vol. 16, no. 8, pp. 1311.

Arivudainambi, D.; Kumar, K.; Chakkaravarthy, S. (2018): LION IDS: a meta-heuristics approach to detect DDoS attacks against software-defined networks. *Neural Computing and Applications*, pp. 1-11.

Cai, Z.; Wang, Z.; Zheng, K.; Cao, J. (2013): A distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering. *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 417-427.

Chen, Y.; Ma, X.; Wu, X. (2013): DDos detection algorithm based on preprocessing network traffic predicted method and chaos theory. *IEEE Communications Letters*, vol. 17, no. 5, pp. 1052-1054.

Cheng, R.; Xu, R.; Tang, X.; Sheng, V. S.; Cai, C. (2018): An abnormal network flow feature sequence prediction approach for DDos attacks detection in big data environment. *Computers, Materials & Continua*, vol. 55, no. 1, pp. 95.

Hsieh, C. J.; Chan, T. Y. (2016): Detection DDos attacks based on neural-network using apache spark. *International Conference on Applied System Innovation*, pp. 1-4.

Idhammad, M.; Afdel, K.; Belouch, M. (2018): Semi-supervised machine learning approach for DDos detection. *Applied Intelligence*, pp. 1-16.

Jiao, J.; Ye, B.; Zhao, Y.; Stones, R. J.; Wang, G. et al. (2017): Detecting TCP-based DDos attacks in Baidu cloud computing data centers. *IEEE 36th Symposium on Reliable Distributed Systems*, pp. 256-258.

Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. (2017): DDos in the IoT: mirai and other botnets. *Computer*, vol. 50, no. 7, pp. 80-84.

Kumarasamy, S. (2009): An effective defence mechanism for distributed denial-of-service (DDos) attacks using router-based techniques. *International Journal of Critical Infrastructures*, vol. 6, no. 1, pp. 73-80.

Li, C.; Wu, Y.; Yuan, X.; Sun, Z.; Wang, W. et al. (2018): Detection and defense of DDos attack-based on deep learning in openflow-based SDN. *International Journal of Communication Systems*, vol. 31, no. 5, e3497.

Li, S. H.; Kao, Y. C.; Zhang, Z. C.; Chuang, Y. P.; Yen, D. C. (2015): A network behavior-based botnet detection mechanism using pso and k-means. *ACM Transactions on Management Information Systems*, vol. 6, no. 1, pp. 3.

Liao, Q.; Li, H.; Kang, S.; Liu, C. (2015): Application layer DDos attack detection using cluster with label based on sparse vector decomposition and rhythm matching. *Security and Communication Networks*, vol. 8, no. 17, pp. 3111-3120.

Lin, Q.; Li, J.; Huang, Z.; Chen, W.; Shen, J. (2018): A short linearly homomorphic proxy signature scheme. *IEEE Access*, vol. 6, pp. 12966-12972.

Lin, Q.; Yan, H.; Huang, Z.; Chen, W.; Shen, J. et al. (2018): An id-based linearly homomorphic signature scheme and its application in blockchain. *IEEE Access*, vol. 6, pp. 20632-20640.

Liu, S.; Cai, Z.; Xu, H.; Xu, M. (2015): Towards security-aware virtual network embedding. *Computer Networks*, vol. 91, pp. 151-163.

Malialis, K.; Kudenko, D. (2015): Distributed response to network intrusions using multiagent reinforcement learning. *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 270-284.

Nezhad, S. M. T.; Nazari, M.; Gharavol, E. A. (2016): A novel dos and DDos attacks detection algorithm using arima time series model and chaotic system in computer networks. *IEEE Communications Letters*, vol. 20, no. 4, pp. 700-703.

Niyaz, Q.; Javaid, A.; Sun, W.; Alam, M. (2016): A deep learning approach for network intrusion detection system. *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 15, pp. 21-26.

Rajarajan, G.; Ganesan, L. (2017): A decoy framework to protect server from wireless network worms. *Wireless Personal Communications*, vol. 94, no. 4, pp. 1965-1978.

Rukavitsyn, A.; Borisenko, K.; Shorov, A. (2017): Self-learning method for DDos detection model in cloud computing. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*, pp. 544-547.

Saied, A.; Overill, R. E.; Radzik, T. (2016): Detection of known and unknown DDos attacks using artificial neural networks. *Neurocomputing*, vol. 172, pp. 385-393.

Seo, J. W.; Lee, S. J. (2016): A study on efficient detection of network-based IP spoofing DDos and malware-infected systems. *SpringerPlus*, vol. 5, no. 1, pp. 1878.

Shiaeles, S. N.; Papadaki, M. (2014): FHSD: an improved IP spoof detection method for web DDos attacks. *Computer Journal*, vol. 58, no. 4, pp. 892-903.

Somani, G.; Gaur, M. S.; Sanghi, D.; Conti, M. (2016): DDos attacks in cloud computing: collateral damage to non-targets. *Computer Networks*, vol. 109, pp. 157-171.

Wang, K.; Du, M.; Sun, Y.; Vinel, A.; Zhang, Y. (2016): Attack detection and distributed forensics in machine-to-machine networks. *IEEE Network*, vol. 30, no. 6, pp. 49-55.

Xu, J.; Wei, L.; Zhang, Y.; Wang, A.; Zhou, F. et al. (2018): Dynamic fully homomorphic encryption-based Merkle Tree for lightweight streaming authenticated data structures. *Journal of Network and Computer Applications*, vol. 107, pp. 113-124.

Yuan, X.; Li, C.; Li, X. (2017): Deepdefense: identifying DDos attack via deep learning. *IEEE International Conference on Smart Computing*, pp. 1-8.

Zhang, B.; Zhang, T.; Yu, Z. (2017): DDos detection and prevention based on artificial intelligence techniques. *3rd IEEE International Conference on Computer and Communications*, pp. 1276-1280.

Zhang, Y.; Wang, X.; Perrig, A.; Zheng, Z. (2016): Tumbler: adaptable link access in the bots-infested internet. *Computer Networks*, vol. 105, pp. 180-193.

Zhang, X.; Tan, Y.; Liang, C.; Li, Y.; Li, J. (2018): A covert channel over VoLTE via adjusting silence periods. *IEEE Access*, vol. 6.

Zhu, L.; Tang, X.; Shen, M.; Du, X.; Guizani, M. (2018): Privacy-preserving DDos attack detection using cross-domain traffic in software defined networks. *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628-643.