IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 24, 2020, accepted October 19, 2020, date of publication October 26, 2020, date of current version November 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033494

Unknown Attack Detection Based on Zero-Shot Learning

ZHUN ZHANG[1], QIHE LIU[1], SHILIN QIU[1], SHIJIE ZHOU[1], (Member, IEEE), AND CHENG ZHANG[2]

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

Corresponding author: Qihe Liu (qiheliu@uestc.edu.cn)

This work was supported in part by the Sichuan Science and Technology Program under Grant 2018GZDZX0006, Grant 2019YFG0399, Grant 2017GZDZX0002, and Grant 20ZDYF0091; and in part by the National Natural Science Foundation of China under Grant 41774095.

ABSTRACT In recent years, due to the frequent occurrence of network intrusions, more and more researchers have begun to focus on network intrusion detection. However, it is still a challenge to detect unknown attacks. Currently, there are two main methods of unknown attack detection: clustering and honeypot. But they still have unsolved problems such as difficulty in collecting unknown attack samples and failure to detect on time. Zero-Shot learning is proposed to deal with the problem in this article, which can recognize unknown attacks by learning the mapping relations between feature space and semantic space (such as attribute space). When the semantic descriptions of all attacks (including known and unknown attacks) are provided, the classifier built by Zero-Shot learning can extract common semantic information among all attacks and construct connections between known and unknown attacks. The classifier then utilizes the connections to classify unknown attacks although there are no samples for unknown attacks. In this article, we first propose to use Zero-Shot learning to overcome the challenge of unknown attack detection and illustrate the feasibility of this method. Secondly, we then propose a novel method of Zero-Shot learning based on sparse autoencoder for unknown attack detection. This method maps the feature of known attacks to the semantic space, and restores the semantic space to the feature space by constrains of reconstruction error, and establishes the feature to semantic mapping, which is used to detect unknown attacks. Verification tests have been carried out by using the public dataset NSL_KDD. From the experiments conducted in this work, the results show that the average accuracy reaches 88.3%, which performs better than other methods.

INDEX TERMS Zero-shot learning, network intrusion, unknown attack detection, sparse semantic autoencoder.

I. INTRODUCTION

In the real network environment, attackers continuously change their behavior and search for unknown vulnerabilities to bypass the traditional security mechanism. Therefore, unknown attacks grow at a rate of more than 125% every year, and it is expected that there will be a new unknown attack every day until 2021 [1]. It is difficult to identify unknown network attacks since it would take an average of 312 days to obtain the full information of an attack [2]. Current security systems, such as Security Information and Event Management (SIEM), are ineffective because 85% of network intrusions detected attacks within weeks after they occurred [3]. The technology of intrusion detection has gradually become the focus of security researchers [4]–[8]. Researchers proposed various intrusion detection systems based on machine learning [9]–[13],among which supervised or semi-supervised learning methods have become the mainstream [14]–[19]. However, the traditional machine learning methods cannot identify the emerging unknown attacks on time, so unknown attack detection is still a difficult issue.

In recent years, researchers have made great achievements in network intrusion detection and unknown attack detection. For the network intrusion detection, Pervez and Farid [20] proposed a method of intrusion classification based on the combination of feature selection and the Support Vector Machine (SVM) classifier, which maintained the classification accuracy of the SVM classifier and improved

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

the classification ability of the intrusion detection system by reducing the input features of training data. Ganapathy *et al.* [21], [22] studied a variety of intelligent feature selection and classification techniques in detail and proposed an Intelligent Agent-based Enhanced Multiclass Support Vector Machine algorithm, which can detect the intruders in a distributed database environment. Although this model has a high detection rate, it focuses on intrusion detection rather than unknown attack detection. Kim *et al.* [23] preprocessed the data by utilizing data transformation and normalization, and they then inputted these data into the deep neural network (DNN) model. As the DNN algorithm was applied to the preprocessed data refinement, the model was established, and the whole KDD Cup 99 dataset [24] was used for verification. The experimental results show that the DNN model is good in intrusion detection. Kim *et al.* [25] constructed an Intrusion Detection System (IDS) detection model using the same method. The difference is that they introduced the long and short term memory (LSTM) architecture into the recursive neural network model, and achieved good results. However, even if all the hyperparameters in the neural network did not change, the performance of the IDS would change due to different data sets. Ashfaq *et al.* [14] proposed a novel fuzziness based semi-supervised learning approach by utilizing unlabeled samples assisted with a supervised learning algorithm. A single hidden layer feed-forward neural network (SLFN) is trained to output a fuzzy membership vector, and the sample categorization on unlabeled samples is performed using the fuzzy quantity. After merging each class into the training set, the classifier finally performed well on the NSL_KDD dataset [26].

Most of the above methods are based on supervised or semi-supervised, which can accomplish the task of anomaly detection well. However, they are not suitable for unknown attack detection due to the difficulties in collecting samples and delays in the detection process. For the detection of unknown attacks, Saied *et al.* [27] introduced the artificial neural network (ANN), which uses the known DDOS attacks as the training set to detect unknown DDOS attacks. Compared with the traditional method, the accuracy of this method is up to 98%. However, this method can only be used to identify a single attack type. Wang *et al.* [28] proposed a seed extension (SE) cluster technology for early attack detection. The unsupervised algorithm cluster has advantages over the traditional k-means algorithm. Lobato *et al.* [29] proposed an adaptive threat detection architecture to solve the problem that the previous work could not detect unknown attacks in real-time. This architecture transmitted the attack data collected from the honeypot in the network to the training model in real-time, that is, added the attack data to the training set in real-time. However, this method relies too much on extracting information from the honeypot. Rivero *et al.* [30] proposed a method of network intrusion detection based on Grassmann.In the learning phase, the decision tree is used to extract rules to generate attributes. In the inference phase, Grassmann manifold is used to calculate the distance between unknown attacks and known attacks. This method mentions the Zero-Shot Learning method, but it does not establish the relationship between known attacks and unknown attacks in the learning stage, so it is not the real Zero-Shot Learning. Although in machine learning, there are some techniques that can be used to classify unknown data, in the field of attack detection, people not only need to find out the unknown data but also need to distinguish what attack it comes from.

The current studies of the unknown attack detection have made several achievements, but these methods still have many problems, such as the detected attack class is single, the attacks samples are difficult to obtain, existing models cannot detect attacks in real-time, and there is a lot of resource consumption. At present, the detection of unknown attacks is mainly based on the method of clustering or honeypot. Among them, the method based on clustering needs a sufficient number of attack examples, but the occurrence of unknown attacks is uncertain, and it is often difficult to collect a sufficient number of attack examples in a short time, so it is difficult to detect the unknown attack effectively on time. As for honeypot based methods, when collecting attack samples, need to consume a lot of resources to build a honeypot system that can deceive attackers. Meanwhile, honeypot systems still have the risk of being used by cunning intruders to attack other systems in turn. Compared to the above methods, the Zero-Shot Learning(ZSL) method, which detects the unknown attack only by its semantic description, is faster than the clustering-based method and consumes significantly fewer resources than honeypot-based method.

In this article, firstly, we propose to use ZSL to overcome the issue of unknown attack detection and illustrate the feasibility of this method. In this part, we describe two preconditions for applying ZSL method in the unknown attack detection, which is obtaining the semantic description of the attack and establishing the relationship between known attack and unknown attack. Then the feasibility of this method is explained. Secondly, due to the poor performance of existing ZSL methods for unknown attack detection, inspired by the semantic autoencoder model from Breiman [31], we propose a novel method of ZSL based on autoencoder for unknown attack detection. This method maps the feature of known attacks to the semantic space, and restores the semantic space to the feature space by constrains of reconstruction error, and establishes the feature to semantic mapping, which is used to detect unknown attacks. The method maps the features of known attacks to the semantic space, associates them with corresponding semantics, and establishes the feature-to-semantic mapping. On this basis, to extract semantic information more effectively, we apply the constraint of reconstruction error to restore the semantic space to the feature space, which strengthens the feature to semantic mapping. This mapping is used to detect unknown attacks. Verification tests have been carried out by using the public dataset NSL_KDD. The results show that our new ZSL method is suitable for unknown attack detection and is superior to other methods.

The main contributions of this article are as follows:

(1) We first propose to use ZSL to overcome the challenge of unknown attack detection and illustrate the feasibility of this method theoretically.

(2) We propose a novel ZSL method for unknown attack detection, which maps the semantic information to the semantic space for semantic correlation and by the constraints of error reconstruction restores the semantic space to the original semantic information. Experimental results show that this method can extract semantic information more effectively.

The remainder of this article is organized as follows. The related works are in section II. In section III, we describe the methodology of ZSL for unknown attacks detection. In section IV, we propose our novel method of ZSL for unknown attacks detection. Section V is the conclusion. Section VI is about future work.

II. RELATED WORK

ZSL is an emerging method, which has been widely used in computer vision, machine translation, and other fields. It is mainly used to recognize new classes that appear after the end of the training phase. The ZSL method can be divided into two phases: the learning phase and the inference phase. Before the learning, the semantic information of the classes (both seen classes and unseen classes) needs to be obtained first. Then, all the semantic information is summarized to form the semantic description library. The learning phase aims to establish mapping relations between original features and semantic features by learning the projection from feature space to semantic space (such as attribute space). In the inference phase, the new attack class generates its semantic vectors through the mapping relations acquired in the learning phase. Then the semantic vectors of this new class are compared with semantic descriptions in the semantic description library to find the closest one, then the new class is the closest one. For example, if the seen class has ''horse'' and ''black and white stripes,'' and the semantic description of the unseen class ''zebra'' is ''a horse with black and white stripes is a zebra.'' The model can identify zebra without the sample of zebra. Therefore, the semantic description is the bridge between the seen classes and the unseen classes.

Currently, the algorithm of ZSL is mainly divided into two schemes. The first scheme is only to learn the projection function from the sample feature space to the sample semantic space (attribute space) [32]. Lampert *et al.* [33] proposed a Direct Attribute Prediction (DAP) model. DAP model consists of three layers: an input layer, high-dimensional feature space, and an output layer. Firstly, DAP trains a p-dimensional classifier between the first and second layers to determine whether the input samples conform to the corresponding high-dimensional eigenvalues of each dimension in the p-dimensional space. Finally, the corpus knowledge base between the second and third layers is used to predict the test sample, that is, to find the closest vector to the semantic vector of the test sample. Akata *et al.* [32] presented a Label-Embedding for Attribute-Based Classification (ALE) model. This model aims to establish a mapping relationship from the original features to the high-dimensional semantic vector. Two functions are defined in ALE: prediction function $f(x; w)$ and the matching degree function $F(x, y; w)$, where x is the sample feature, y is the attribute vector, and w is the model parameter. The main problem of ALE is the mapping domain drift.

The second scheme is the idea of establishing an encoder-decoder [34]. In this scheme, the encoder was similar to the first scheme, but an additional constraint that the projected semantic space must be able to reconstruct the original feature space was added to the decoder. With this constraint, projection functions learned from known classes can be better generalized to new unknown classes. Kodirov *et al.* proposed a semantic autoencoder (SAE) model [31], which added a decoder based on ALE to restore the original features of the sample using the generated semantic vector. In other words, this model not only generated the mapping relationship between the original features and the semantic vector of the sample but also retained the information of the original sample. SAE has only one encoder and one decoder. The input sample X is transformed into semantic space S through matrix w, and sample X is restored through matrix w^T. The objective function of SAE is shown in (1).

$$\min_{w}(||X - W^T S||_F^2 + \lambda||WX - S||_F^2) \tag{1}$$

λ is the weight of the encoder. We can obtain the matrix W that minimizes the objective function through mathematical calculation. Firstly, we need the transformation formula of the matrix trace, and we can get the matrix W by setting the derivative of W to zero, as shown in (2) (3).

$$\min_{w}(||X^T - S^T W||_F^2 + \lambda||WX - S||_F^2) \tag{2}$$

$$-S(X^T - S^T W) + \lambda(WX - S)X^T = 0,$$

$$SS^T W + \lambda WXX^T = SX^T + \lambda SX^T. \tag{3}$$

SAE solved the problem of domain drift to a certain extent. However, because the given constraint is too strong, the mapping matrix w can be calculated without training. In other words, if the initial feature space and the semantic space are given, we can calculate the matrix W directly through the given formula.

III. THE PROPOSED METHODOLOGY

For unknown attack detection, semantic information is the key to link known attacks and unknown attacks. The application of the ZSL method needs to satisfy two preconditions.

The first precondition is that we need to obtain the semantic descriptions of both known attacks and unknown attacks. Among them, semantic descriptions of known attacks can be easily obtained in a variety of ways, such as Wikipedia, Google, and various secure websites. For unknown attacks, it is difficult to timely access to its data and related information in detail. However, its important features, symptoms, and other related description information will soon appear

in many network security forums, and we can get the feature descriptions of unknown attacks by focusing on these sites. For example, we can capture attacks and their associated descriptions in real-time on well-known websites like Exploits Database by Offensive Security [35]. In this article, for the NSL_KDD dataset, we integrate the semantic description from the attack by building a semantic description library.

The second precondition is the need to construct the relationship between known attacks and unknown attacks. In other words, we need to extract commonalities between known and unknown attacks. For example, the unknown class ''zebra'' has the two feature descriptions of ''black and white stripes'' and ''horse''. And these two feature descriptions need to be learned in the known class, among which the more descriptions learned, the higher the recognition accuracy. For ease of understanding, in this part, unknown attacks are roughly divided into two classes. One is evolved attacks, which evolved from one or more original attacks. The other class is new attacks, which means these attacks have unique feature descriptions that cannot be found in any known attacks. For the evolved attacks, most of the feature descriptions of such attacks can be learned from known attacks. For the new attacks, although it is impossible to find related feature descriptions from any known attacks, it can be quickly distinguished from other attacks due to its unique features. In this article, the proposed method maps the semantic information of known attacks and unknown attacks to the semantic space for semantic correlation.

Taking the attacks of DOS as an example, Table 1 shows six attacks of DOS and their feature descriptions. As for DOS attacks, they eventually lead the computer to denial of service. Among them, Pod, ICMP flood, and Smurf appear the related feature descriptions of the ping packets, while Land and SYN flood all have the related descriptions of the TCP SYN packets. The only difference between Fraggle and Smurf is that they use different packets (Fraggle is UDP packets, Smurf is ICMP packets). There are commonalities among attacks, especially among attacks of the same kind. So the model can learn feature descriptions from other attacks.

TABLE 1. Six attacks of DOS and their feature descriptions.

Attacks	Descriptions	Feature descriptions
Pod	Pod is a form of attack that sends malicious ping packets to a target computer. It is illegal for the IP protocol that sending more than 65,536 bytes of a ping packet. If the ping packet is sent in multiple segments, the target computer must constantly repackage the packet, which may cause a buffer overflow and cause the system to crash.	• malicious ping packets; • ICMP; • beyond the byte; • buffer overflow; • denial-of-service;
ICMP flood	The attackers use the ICMP echo request message to flood the target device. They consume the resources of the target host by sending a large number of ping packets in a short time. It will be paralyzed or unable to provide other services when the host resources are exhausted.	• a large number of ping packets; • ICMP echo request message; • resources exhausted; • denial-of-service;
Smurf	Modify the reply address of the ICMP echo request (ping) packet to the broadcast address of the affected network. All hosts in the network will respond to ICMP echo requests after receiving the broadcast, causing network congestion.	• ping packets; • The ICMP reply address is set to the broadcast address of the victim network; • denial-of-service;
Fraggle	Modify the reply address of the UDP reply request packet to the broadcast address of the affected network. All hosts in the network will respond to the UDP reply request after receiving the broadcast, causing network congestion.	• The UDP reply address is set to the broadcast address of the victim network; • denial-of-service;
Land	Modify the source and destination addresses of the TCP SYN packet (usually used to open a new connection) to its IP address. When the target host opens an empty connection, it constantly replies to itself and consumes system resources until it crashes.	• TCP SYN packet; • both the source and destinationt addresses being its own IP addresses; • denial-of-service;
SYN flood	The attacker sends a large number of TCP SYN packets to the server without confirming them, which makes each of them cannot complete three handshakes. These TCP connections consume CPU and memory due to the pending state, making the server unable to service normal users.	• a large number of TCP SYN packets; • connection unsuccessful; • resources exhausted; • denial-of-service;

IV. PROPOSED METHOD

The method of unknown attacks detection based on ZSL has two parts. The first part is the establishment of the attack semantic description library, which aims to store the semantic description of the known attack class and the unknown attack class. The second part is model building, which maps the semantic information of known attacks and unknown attacks to the semantic space for semantic association and restores the semantic space to the original semantic information for error reconstruction.

A. SEMANTIC DESCRIPTIONS LIBRARY OF ATTACKS

Building a machine-readable semantic description library is one of the keys to ZSL. The construction of semantic description library can be divided into two ways, (1) engineered semantic spaces(each dimension of the semantic space is designed by humans), and (2) learned semantic spaces(each dimension does not have an explicit semantic meaning) [36]. In this article, we choose the attribute space of engineering semantic space. Attribute spaces are kinds of semantic spaces that are constructed by a set of attributes, which are one of the most widely used semantic spaces in ZSL [33], [37]–[39].

There are two steps to build an attack semantic description library based on attribute space. (1) Collect the semantic description information of all attacks (including known attacks and unknown attacks). (2) Transformed the semantic

description information of all attacks into numerical vectors through natural language processing, which can be recognized by computers. For the first step, the unknown attacks mainly collect from security websites, and known attacks can be obtained by Wikipedia. The second step is the one we focus on. Although some methods such as one-hot encoding can directly calculate the vectors of different words, they also have two defects. One is the dimension disaster which means too many words lead to growing dimensions. Second, they do not consider the semantic relationship between words. Word to vector(Word2VEC) [40] takes each text as a training sample to fully considers the semantic information between words, and then maps each word to a shorter vector, which solves the above two defects well. So we choose Word2VEC to convert semantic information into vectors.

In this article, we explain the steps of generating the semantic description library by combining Word2VEC. The first is to train the Word2VEC model, which mainly relies on the WikiCorpus. The second is to convert semantic text into vector form through Word2VEC model. The details are shown below:

Step1: According to the WikiCorpus, initialize vocabulary $V = (v_1, v_2, v_3, \ldots, v_n)$ as N-dimensional vectors (randomly generated).

Step2: Count the number of occurrences of each word in vocabulary v_i as the weight of this word, and build a Huffman tree based on these weights. In Huffman trees, the leaf node represents the frequency of words.

Step3: Select the word $v_i \in V$ in the vocabulary as the center word in turn, and set window size. Train all words in the window centered on v_i, and add up these vectors to get a new vector V_{new}.

Step4: Each leaf node of the Huffman tree represents a word. Take v_i as an example, V_{new} can finally reach the Huffman leaf node corresponding to v_i by selecting the left or right subtree at the root node of the Huffman tree. Each non-leaf node has a Softmax regression layer, and each time in the process of V_{new} selection, a probability will be calculated with a non-leaf node through Softmax, and the formula is (4).

$$Softmax(v_i) = v_i' = \frac{e^{v_i}}{\sum_{j=1}^{n} e^{v_j}} \tag{4}$$

where v_i represents the calculation result of a dimension value of V_{new} and non-leaf nodes. n represents the dimension.

Step5: Calculate the Softmax between v_i and the root node to get the probability p_1 of choosing the right subtree, and so on, to get the probability values p_2 and P3 in the remaining selection process. We can get the probability $P = p_1 \times p_2 \times p_3$ of V_i under the Word2VEC model, and the residual is $1 - P$. Then gradient descent can be used to update the parameters of each non-leaf node in the Huffman tree until the Huffman encoding calculated by each V_i according to Softmax is close to the real encoding. Finally, the loss function converges to obtain the vector representation of each word, that is, to obtain the Word2VEC model.

Step6: Standardize the attack description information collected.

Step7: Through the trained Word2VEC model, the words in each attack description are transformed into vectors representation, and then the word vectors of each word in each attack description are superposition and averaged to form an attack semantic description library.

B. SEMANTIC MAPPING MODEL

The ideas of the traditional ZSL usually only learn the projection function from the feature space to the semantic space. However, this projection function can only predict the semantic classification of the same class. The SAE model improved on this. It established a constraint on the original projection function, and the semantic space generated by the encoder can be restored to the feature space of the sample by the decoder. This improvement solves the problem of mapping domain drift to some extent. The encoder and decoder of the SAE model are based on linear transformation. It can be seen from the formula. If the initial feature space and the semantic space are given, we can calculate the matrix w directly from the given formula.

In the SAE model, W^T is used to restore the input sample X, which imposes a compulsory constraint on encoding. We hope to reduce this constraint by combining sparse autoencoder with linear coding. The hidden layer from the middle of the encoder represents the compressed representation of the input sample. When the dimension of the middle layer is larger than the dimension of the input sample, or there are too many neurons in the hidden layer, it is difficult for the ordinary autoencoder to get the compressed representation of the input sample. However, the sparse autoencoder adds sparsity limitation to the neurons, so that the autoencoder can still learn some rules in the sample features when there are many neurons. The key of the sparse autoencoder is how to deal with the sparsity of neurons. Different activation functions have different sparsity methods. Taking the sigmoid activation function as an example, if the output of sample features obtained by linear calculation of neurons and nonlinear transformation of the sigmoid function is close to 0, the neuron is not activated, otherwise, the neuron is activated. Fig. 1. shows the autoencoder structure adopted in this article, in which the first encoding layer and the last decoding layer adopt the nonlinear sparse encoder, and the second encoding layer and semantic decoding layer adopt the linear encoder (without activation function).

We will introduce the semantic autoencoder through the following steps:

Step1: Convert the input sample into word vectors.

Step2: Input the word vectors into the autoencoder. The first layer of the autoencoder is sparse coding. The input sample is x, and the encoding function is shown in (5).

$$y = f(x) = S_f(Wx + b) \tag{5}$$

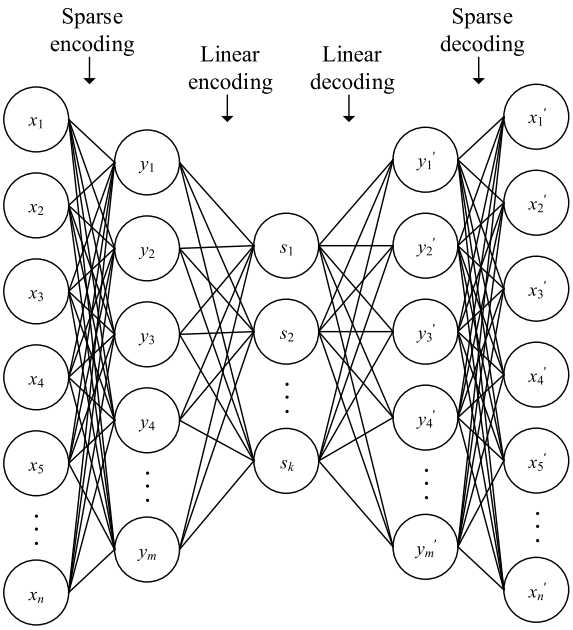


FIGURE 1. The model of sparse semantic autoencoder.

S_f is the activation function, whose function is to perform the nonlinear transformation. Here, W is the weight, and b is the bias term.

Step3: The sparse autoencoder will generate an average activation degree in the encoding process. If the activation function S_f in Step2 adopts the sigmoid activation function, as shown in (6), the output of the neuron will be limited to (0,1).

$$S_f(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

When the output of the neuron approaches 1, the neuron is considered to be activated. When the neuron approaches 0, the neuron is considered to be suppressed. When the input is x, assuming that $a_i(x)$ represents the activation degree of neuron i, the average activation degree can be calculated by the formula (7).

$$\overline{\rho}_i = \frac{1}{m} \sum a_i(x_i) \tag{7}$$

where m represents the sample dimension.

Step4: We set the sparsity hyperparameter ρ to constrain i, that is, i approaches ρ. This is achieved by adding a penalty factor into the loss function, which is calculated through KL divergence, as shown in (8).

$$KL(\rho||\overline{\rho}_i) = \rho log\frac{\rho}{\overline{\rho}_i} + (1-\rho)log\frac{1-\rho}{1-\overline{\rho}_i} \tag{8}$$

Step5: The second layer of linear encoder mainly encodes the output y of the first layer again, but the linear encoder has no activation function. As shown in (9).

$$h = g(x) = W_2 x + b_2 \tag{9}$$

Step6: The decoding process is similar to the inverse operation of the encoding process, which is mainly to constrain the encoding process. After decoding, the encoder needs to conduct backpropagation to optimize the parameters of each layer. The loss function we set is shown in (10).

$$J(W, b) = \lambda|V(x) - s| + |V'V(x) - x| \tag{10}$$

In the above equation, k is the encoding weight we set, $V(x)$ is the encoding function, and $V'(x)$ is the decoding function.

Step7: To constrain the output of each neuron, a penalty factor was added to the loss function $J(W, b)$ of the sparse self-encoder, namely the KL divergence obtained by Step4, as shown in (11).

$$J_{sparse}(W, b) = J(W, b) + \beta \sum_{i=1}^{s2} KL(\rho||\overline{\rho}_i) \tag{11}$$

where, $s2$ is the number of neurons in the hidden layer, and β is the weight of the punishment factor.

Step8: The autoencoder updates the weights, bias terms, and the average activation degree of neurons through multiple forward propagation and backpropagation. So the activation function in Step7 can reach a smaller value. Finally, a coding model mapped by attack feature to semantic space vector is obtained.

V. EXPERIMENTS EVALUATIONS AND DISCUSSION

In this section, we introduce the dataset first. And then, the implementation details will be described. The last part is the conclusion. In addition, in this article, the code is written based on Python 3, and the semantic autoencoder is implemented by the deep learning framework of Keras.

A. NSL-KDD DATASET

KDD CUP 99 (KDD99) dataset [24], [26] which collected by Defense Advanced Research Projects Agency(DARPA) in 1998 to evaluate intrusion detection performance is widely used in intrusion detection researches. The reason why this dataset is still favored by researchers 20 years later is that this dataset is almost from the real network environment. Over a period of about nine weeks, DARPA collected a large number of packets captured by TcpDump (including network connections and audit records) by simulating normal network traffic and various attacks. On this basis, researchers used technologies of data mining to perform a series of preprocessing and feature extraction on the original packet, which was used in the KDD CUP competition in 1999 and formed the famous KDD99 dataset.The KDD Cup 99 intrusion detection database contains approximately 5 million samples. The dataset contains four different types of attacks: DOS, R2L, U2R, and PROBE. Each instance is represented by a TCP/IP connection consisting of 41 features.

The NSL_KDD dataset [20], [41] improves the KDD 99: (1) The training set of NSL_KDD dataset does not contain redundant records. (2) There is no duplicate record in the test set of NSL_KDD dataset. (3) The number of selected records

from each class is inversely proportional to the percentage of records in the original KDD dataset. (4) The number of records set in the training and testing is reasonable.Therefore, after the improvement, the variation range of classification rates of different machine learning methods increases, which makes the evaluation results of different research work more consistent and comparable.

The NSL_KDD dataset contains normal traffic and four different attack types (DOS, R2L, U2R, PROBE), which are further subdivided into 39 attacks. Some attacks have fewer instances, such as spy and UDP storm, each of them has only two instances. Some other classes have too many instances, such as normal (77,053 instances) and Neptune (45,871 instances).Each record in the NSL_KDD dataset consists of forty one features and one label. Details of the dataset are available in [41].

In previous work, most researchers choose one or two attacks in each type as the testing set. However, there are two problems in this way. First, it is easier to classify the four classes because of the large gap between them, which results in higher classification accuracy. Secondly, the test samples of some classes are too small to reflect the accuracy of classification adequately. To effectively avoid these problems, we deleted attacks with few samples and selected 12 attacks as the training set, 10 attacks as the testing set, in which six attacks from DOS, one from PROBE, one from R2L, and two attacks from U2R. In this article, we focus on unknown attack detection, so the attacks of the testing set not appear in the training set. The training set and the testing set are shown in Table 2.

TABLE 2. The training set and the testing set.

Class	Training set	Amount	Testing set	Amount
Normal	-	-	-	-
DOS	apache2	737	back	1315
	neptune	45871	mailbomb	293
			pod	242
			teardrop	904
			processtable	685
			smurf	3311
Probe	mscan	996	ipsweep	3740
	nmap	1566		
	portsweep	3088		
	saint	319		
	satan	4368		
U2R	buffer_overflow	50	httptunnel	133
	perl	5		
R2L	guess_passwd	1284	snmpgetattack	178
	warezclient	890	snmpguess	331
	warezmaster	964		

B. PERFORMANCE METRICS

The effectiveness is evaluated based on four possible metrics namely true negative (TN), true positive(TP), false positive (FP), and false negative (FN). If the positive class in the test dataset is classified to be a positive class, that is TP. TN is obtained if a negative class is predicted to be a negative class. FN is obtained if a positive class is classified to be a negative class, and if a negative class is predicted to be a positive class, it is FP. The combination of precision and recall is selected as the evaluation criteria in this article.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{14}$$

C. THE EXPERIMENTAL STEPS OF UNKNOWN ATTACK DETECTION

Section IV introduces the establishment of the semantic description library and the model of the sparse semantic autoencoder. The recognition of unknown attacks is based on the similarity of attacks and the content introduced in section IV. For example, there are two similar attacks A and B, and both belong to DOS. Among them, A is a known attack and B is an unknown attack. In the training phase, only A participated in the construction of the sparse semantic autoencoder. In the inference phase, inputs B to the sparse semantic autoencoder model. The model will generate the semantic vector of B in the similar encoding mode to A. Because they are both from DOS, they have many similarities in semantic descriptions. For example, the common feature of the DOS attacks is making the target computer or network cannot provide normal service or resource access. So the semantic vectors of A and B generated by the model are also close. After the semantic vector of B is generated by the model, we can find the semantic vector closest to it in the attack semantic description library to recognize the unknown attack. The specific steps are as follows:

Step1: Word2vec is used to process the features of unknown attack to get the corresponding word vector $V_m = (V_{m_1}, V_{m_2}, V_{m_3}, \ldots, V_{m_n})$.

Step2: Input the vector V_m to the sparse semantic autoencoder, and generate its semantic vector $P_{new} = (p_1, p_2, p_3, \ldots, p_n)$.

Step3: According to section IV, we established the attack semantic description library $S = (P_1, P_2, P_3, P_4, \ldots, P_n)$.

Step4: Cosine similarity is frequently used to calculate the similarity between texts. The cosine value of two vectors whose angle is equal to 0° is equal to 1, which is that these two vectors tend to be similar. On the contrary, if the angle between two opposite vectors is equal to 180°, the cosine value is equal to −1. And, if the angle between two perpendicular vectors is equal to 90°, and the cosine is equal to 0. The calculation formula is shown in (15).

$$cos\theta = \frac{P_{new} \cdot P_i}{||P_{new}|| \times ||P_i||} = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} \tag{15}$$

TABLE 3. Some of the semantic descriptions.

Attacks	Protocol_type	Service	Src_bytes	Dst_bytes	Wrong_fragment	Count	Srv_count	Dst_host_diff_srv_rate
pod	icmp	ecr_i	1480	0	1	4	6	0.16
smurf	icmp	ecr_i	1032	0	0	375	375	0.04
nmap	icmp	eco_i	8	0	0	1	23	0
ipsweep	icmp	eco_i	8	0	0	1	21	0
saint	icmp	eco_i	20	0	0	1	55	0
satan	icmp	eco_i	20	0	0	1	1	0.06
snmpgetattack	udp	private	105	143	0	2	1	0.01
snmpguess	udp	private	46	46	0	3	3	0.01

x_i is the value of the i^{th} dimension in p_{new}. y_i is the value of the i^{th} dimension in the P_i.

Step5: Each vector in the semantic description library S is taken out separately to calculate the cosine value together with P_{new}. And we can get the vector P whose cosine value is closest to 1 from S. P is the vector most similar to P_{new}, and the attack type of P_{new} is determined according to the label of vector P.

D. EXPERIMENTAL DETAILS

The symbols and their representations in the verification experiment are shown in Table 4.

TABLE 4. Symbols representation.

Symbols	Representation
Tr	Train set
TrS	Semantics of train set
Te	Test set
TeS	Semantics of test set
$TePr$	Predicted value of the test set
TrE	Vector of train set
$TrSE$	Vectors of semantics of train set
TeE	Vector of test set
$TeSE$	Vectors of semantics of test set
$TePr\_hit$	Predictive classification of test set

Through the Random Forest [31], [42] algorithm to filter the importance of attributes, we select 1, 2, 3, 4, 5, 22, 23, 24, 25, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39 as important features to establish the attribute table for the table head. According to the attack characteristics, 20 columns of attributes of 22 attacks (12 training sets and 10 test sets) are added manually, and the partially filled contents are shown in Table 3. Due to the existence of abbreviated words (such as eco_i, etc.) in the data set, a large amount of additional training is required. Therefore, for simplicity, the whole NSL_KDD dataset is used as a *Corpus* to train Word2VEC model. The complete algorithm steps are shown in Algorithm 1.

By way of explanation, our experiment is divided into two parts, for the first part: (1) According to Table 2, the dataset is divided into training set *Tr*, *TrS*, test set *Te*, *TeS*. (2) Use *Corpus* to train Word2VEC model *ebd_model*. (3) The trained Word2VEC model *ebd_model* is used to transform *Tr*, *TrS*,*Te*,*TeS* into word vector *TrE*, *TeE*, *TrSE*,*TeSE*.

For the second part: (1) According to section IV, establish the encoding and decoding model *ae_model*. (2) The corresponding *TrE* and *TrSE* are input into the model. The relations is established in the inner semantic space and constrained by reconstruction error. (3) Input *TeE* to get the predicted value *TePr*. (4) According to Formula (15), the cosine similarity between *TePr* and *TeSE* is calculated, and the predicted classification *TePr_hit* is obtained.

Algorithm 1 Unknown Attack Detection

Initialize: *TrE*=[],*TrSE*=[],*TeE*=[],*TeSE*=[],*TePr_hit*=[]
Input: *Corpus*, *Tr*, *TrS*, *Te*, *TeS*

1: Train the Word2VEC model according to the A of section IV:*ebd_model* = *Word2Vec*(*Corpus*, *size* = 1, *window* = 40, *min_count* = 0)
2: **for** *i* in *Tr* **do**
3: *TrE*.append(*ebd_model*[i]);
4: **end for**
5: **for** *i* in *TrS* **do**
6: *TrS*.append(*ebd_model*[i]);
7: **end for**
8: **for** *i* in *Te* **do**
9: *TeE*.append(*ebd_model*[i]);
10: **end for**
11: **for** *i* in *TeS* **do**
12: *TeSE*.append(*ebd_model*[i]);
13: **end for**
14: According to the B of Section IV,constructe model: *ae_model* = *Model*(*inputs* = *input_img*, *outputs* = [*decoded*, *encoder_output*])
15: The compiling of ae_model is *ae_model.fit*(*TrE*, [*TrE*, *TrSE*], *class_weight* = [10, 1], *epochs* = 100, *batch_size* = 64)
16: *TePr*=ae_model.predict(*TeE*)
17: According to steps 4 and 5 in the previous section, *TePr* was classified and *TePr_hit* was obtained.

Output: *TePr_hit*

TABLE 5. The experimental results of unknown attack detection (Method 1: SAE, Method 2: Encoder, Method 3: The method we proposed).

Type	Attacks	Amount	Method 1		Method 2		Method 3	
			Recall	Precision	Recall	Precision	Recall	Precision
DOS	back	1315	0.0%	/	98.5%	96.6%	**98.8%**	**98.2%**
	mailbomb	293	0.0%	/	86.3%	87.2%	**93.5%**	**87.3%**
	pod	242	0.0%	/	**94.2%**	23.8%	89.2%	**26.9%**
	teardrop	904	**84.0%**	43.2%	65.7%	98.2%	69.7%	**99.5%**
	processtable	685	0.0%	/	**99.4%**	96.9%	99.3%	**97.0%**
	smurf	3311	**90.1%**	69.4%	79.6%	**94.7%**	85.2%	92.5%
Probe	ipsweep	3740	**97.2%**	85.5%	95.3%	96.1%	96.2%	**98.8%**
U2R	httptunnel	133	0.0%	0.0%	0.0%	/	**21%**	**96.6%**
R2L	snmpgetattack	178	0.0%	0.0%	52.2%	45.6%	**60.1%**	**53.2%**
	snmpguess	331	25.1%	**46.1%**	**69.2%**	42.3%	54.1%	45.0%
Average accuracy			67.0%		86.0%		**88.3%**	

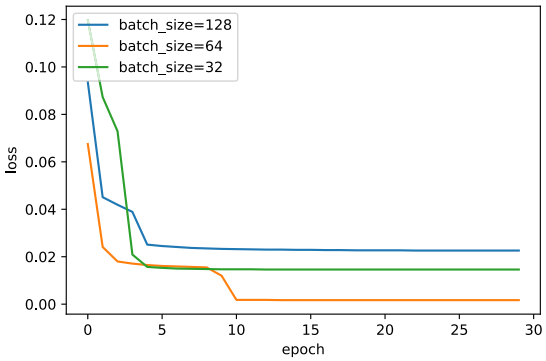


FIGURE 2. The curve of loss function of sparse semantic autoencoder.

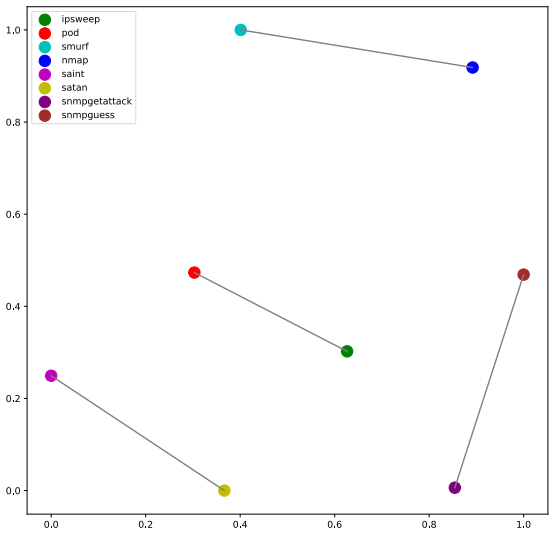


FIGURE 3. Attacks in the embeding space.

It can be seen from Fig. 2. We selected different batch sizes and the best batch size is 64. After 10 iterations, the overall loss function of the model basically converges. We selected 8 kinds of attacks. Some of their semantic descriptions are shown in Table 3. These attacks can be divided into two parts. The first part contains pod, sumrf, nmap, ipsweep, saint, and satan. The second part contains snmpgetattack and snmpguess. In each part, the attacks have similar attack segments. But between the two parts, the attack segments are different. In the first part, the attacks are similar in terms of protocol_type, service, and dst_bytes. But the attacks in the second part differs from the first part in these respects. At the same time, there is a certain similarity between them. Convert them into semantic vectors and project into the embedding space. The six attacks in the first part can be connected into three lines. They point in roughly the same direction. The two attacks in the second part can be connected into one line, which direction is different from the first three lines, as shown in Fig. 3. So attacks with similar segments have the same drift directions, which indicates that the semantic features we extracted are correct.

E. EVALUATIONS AND DISCUSSION

We conducted two groups of comparative experiments. The first group adopted the method of SAE, a classic ZSL method. The second group is the method of the encoder for the experiment. The experimental results are shown in Table 5 (Method 1:SAE, Method 2:encoder, Method 3: the method we proposed). Among them, although the SAE model has a very high recall rate for individual attacks, the recall rate of six attacks is 0, so the traditional ZSL method is not suitable for unknown attack detection on the whole. The encoder model has greater improvement in results than the SAE model. Due to the constraint of reconstruction errors, the average accuracy of the method we proposed is 2.3% higher than the encoder, which proves that our ZSL method has been applicable to unknown attack detection.

Because there are only two types of training samples of U2R and the number is small, the classification effect of the three models is not good. In the first group of experiments, the SAE model has a good performance in the identification of three attacks: teardrop, smurf, and ipsweep. However, for other unknown attacks, the recall rate and precision ratio are all 0, and the precision ratio is generally low, so it is not applicable to unknown attack detection. In the second

group of experiments, the model of encoder has been greatly improved compared with the SAE model, and has a higher recall rate in recognition of pod, processtable and snmpguess, but the precision ratio is generally lower than the model we proposed. Among them, Pod is similar to smurF, so the encoder without reconstruction error constraints prefer pod, which leads to a large number of smurf samples being identified as pod. According to the statistical results, although the recall rate of pod increased by 5%(12 samples), more than 200 smurf samples were identified as pod, leading to a decrease in the recall rate of smurf. The situation is the same for snmpguess and snmpattack. In general, the novel ZSL method we proposed is suitable for unknown attack detection.

VI. CONCLUSION

The novel ZSL method we proposed can solve the issue of unknown attacks detection. Compared with the clustering-based and the honeypot-based methods, it does not need to collect detailed samples of unknown attacks in advance, only needs to collect feature descriptions of unknown attacks. The feature descriptions can be obtained in various network security forums. The ZSL-based method not only reduces the occupation of network resources but also reduces the consumption of costs, and has a better real-time performance. ZSL method can effectively improve the accuracy of unknown attack detection and the ability to recognize intrusion. Our sparse semantic autoencoder can fully learn the mapping relations between the initial feature and semantic feature when the semantic descriptions of the training set are normative and sufficient. When identifying a new attack, if the semantic description of the attack is unique, it can be easily identified based on the particularity of the attack. Moreover, the semantic description of the attack can be learned from other attacks. In other words, the new attack can learn its corresponding semantic vector through the semantic autoencoder. Experimental results show that the ZSL method not only performs well in the identification of unknown attacks across classes but also has a strong identification ability in the same class.

VII. FUTURE WORK

The unknown attack detection method based on ZSL is more dependent on the establishment of semantic descriptions. How to use different descriptions to distinguish similar attacks is the key to improve the detection accuracy. We will continue to study the similarities and differences between attacks in the next work. The experiment in this article is based on the NSL_KDD dataset. However, this data set is relatively old, and some attack types have been outdated. In the next work, we will start to obtain the semantic description of attacks from various network security BBS, and try to improve our model. In the future, we will pay more attention to the detection of new unknown attacks in the real network environment.

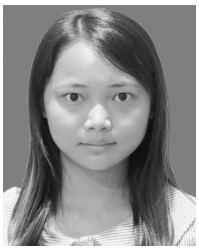
REFERENCES

[1] *Internet Security Threat Report*, I. Symante, Mountain View, CA, USA, 2014.
[2] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 833–844.
[3] P. Clay, "A modern threat response framework," *Netw. Secur.*, vol. 2015, no. 4, pp. 5–10, Apr. 2015.
[4] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 19, pp. 1–13, Jan. 2019.
[5] Q. Wang and P. Lu, "Research on application of artificial intelligence in computer network technology," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 5, May 2019, Art. no. 1959015.
[6] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "A comprehensive study on APT attacks and countermeasures for future networks and communications: Challenges and solutions," *J. Supercomput.*, vol. 75, no. 8, pp. 4543–4574, Aug. 2019.
[7] C. R. King, A. Zhang, T. M. Tessier, S. F. Gameiro, and J. S. Mymryk, "Hacking the cell: Network intrusion and exploitation by adenovirus E1A," *mBio*, vol. 9, no. 3, May 2018.
[8] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.
[9] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.
[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[11] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[12] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[13] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Comput. Secur.*, vol. 77, pp. 304–314, Aug. 2018.
[14] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.
[15] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.
[16] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, and J. Ma, "SU-IDS: A semi-supervised and unsupervised framework for network intrusion detection," in *Proc. Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 322–334.
[17] J. Camacho, G. Macia-Fernandez, N. M. Fuentes-Garcia, and E. Saccenti, "Semi-supervised multivariate statistical network monitoring for learning security threats," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2179–2189, Aug. 2019.
[18] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1949–1959, Apr. 2019.
[19] Y. Yuan, L. Huo, Y. Yuan, and Z. Wang, "Semi-supervised tri-Adaboost algorithm for network intrusion detection," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 6, 2019, Art. no. 1550147719846052.
[20] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *Proc. 8th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, Dec. 2014, pp. 1–6.
[21] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: A survey," *EURASIP J. Wireless Commun. Netw.*, vol. 2013, no. 1, p. 271, Dec. 2013.
[22] S. Ganapathy, P. Yogesh, and A. Kannan, "Intelligent agent-based intrusion detection system using enhanced multiclass SVM," *Comput. Intell. Neuroence*, vol. 2012, Jan. 2012, Art. no. 850259.

[23] J. Kim, N. Shin, S. Y. Jo, and S. Hyun Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 313–316.
[24] C. Elkan, "Results of the KDD'99 classifier learning," *ACM SIGKDD Explor. Newslett.*, vol. 1, no. 2, pp. 63–64, Jan. 2000.
[25] J. Kim, J. Kim, H. L. Thi Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.
[26] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[27] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, Jan. 2016.
[28] J. Wang, L. Yang, J. Wu, and J. H. Abawajy, "Clustering analysis for malicious network traffic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[29] A. G. P. Lobato, M. A. Lopez, I. J. Sanz, A. A. Cardenas, O. C. M. B. Duarte, and G. Pujolle, "An adaptive real-time architecture for zero-day threat detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[30] J. Rivero, B. Ribeiro, N. Chen, and F. S. Leite, "A Grassmannian approach to zero-shot learning for network intrusion detection," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 565–575.
[31] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[32] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 819–826.
[33] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 951–958.
[34] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3174–3183.
[35] *Exploits Database by Offensive Security*. Accessed: Sep. 20, 2020. [Online]. Available: https://www.exploit-db.com/
[36] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–37, 2019.
[37] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1410–1418.
[38] D. Parikh and K. Grauman, "Relative attributes," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 503–510.
[39] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–37, 2019.
[40] X. Rong, "Word2vec parameter learning explained," 2014, *arXiv:1411.2738*. [Online]. Available: https://arxiv.org/abs/1411.2738
[41] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.
[42] N. Shone, T. Nguyen Ngoc, V. Dinh Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

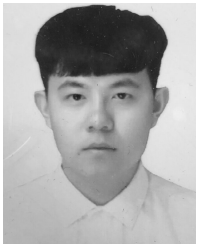
ZHUN ZHANG is currently pursuing the master's degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His main research interests include cybersecurity and artificial intelligence security.

QIHE LIU received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, in 2005. He is currently an Associate Professor with the University of Electronic Science and Technology of China. His current research interests include machine olfaction, data mining, and computer vision. He has authored or coauthored over 20 research papers published in international journals and conference proceedings.



SHILIN QIU is currently pursuing the Ph.D. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. Her research interest includes artificial intelligence adversarial technology.

SHIJIE ZHOU (Member, IEEE) received the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China (UESTC), in 2004. He is currently a Professor with the School of Information and Software Engineering, UESTC. His research interests include cybersecurity communications and computer networks, traffic simulation, and artificial intelligence.



CHENG ZHANG received the master's degree in computer science from the University of Electronic Science and Technology of China, in 2019. His research interests include artificial intelligence and cybersecurity.