IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2020, accepted July 24, 2020, date of publication August 3, 2020, date of current version August 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013849

The Weighted Word2vec Paragraph Vectors for Anomaly Detection Over HTTP Traffic

JIELING LI, HAO ZHANG, AND ZHIQIANG WEI
College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China

Corresponding author: Hao Zhang (zhanghao@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672159, and in part by the Joint Straits Fund of Key Program of the National Natural Science Foundation of China under Grant U1705262.

ABSTRACT Anomaly detection over HTTP traffic has attracted much attention in recent years, which plays a vital role in many domains. This article proposes an efficient machine learning approach to detect anomalous HTTP traffic that addresses the problems of existing methods, such as data redundancy and high training complexity. This algorithm draws on natural language processing (NLP) technology, uses the Word2vec algorithm to deal with the semantic gap, and implements Term Frequency-Inverse Document Frequency (TF-IDF) weighted mapping of HTTP traffic to construct a low-dimensional paragraph vector representation to reduce training complexity. Then we employs boosting algorithm Light Gradient Boosting Machine (LightGBM) and Categorical Boosting (CatBoost) to build an efficient and accurate anomaly detection model. The proposed method is tested on some artificial data sets, such as HTTP DATASET CSIC 2010, UNSW-NB15, and Malicious-URLs. Experimental results reveal that both the boosting algorithms have high detection accuracy, high true positive rate, and low false positive rate. Compared with other anomaly detection methods, the proposed algorithms require relatively short running time and low CPU memory consumption.

INDEX TERMS Anomaly detection, Word2vec, TF-IDF, LightGBM, CatBoost.

I. INTRODUCTION

Industrial Internet continues to advance, and new-generation information technologies and applications such as 5G, blockchain, and cloud computing continue to deepen, are accelerating the process of digitalization and industrial upgrading in various industries. In the development of big data, Web security will remain a popular hot topic in network security. With the widespread use of network technologies such as JavaScript, etc., new types of attacks continue to emerge. New attack methods such as ''XML External Entities (XXE)'' and ''Insecure Deserialization'' have been added to the Open Web Application Security Project (OWASP) Top Ten Attack Types list [1]. Attack methods are becoming more and more diversified, which brings very serious challenges to Web security defense. Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia document, such as HTML. It was designed for communication between Webs. Anomaly detection for HTTP traffic has become the main research direction of security tools.

Anomaly detection for HTTP traffic is performed traditionally based on statistical analysis [2]–[4]. However, not all behaviors can be expressed by using statistical models. For some hidden attack methods, the establishment of corresponding behavioral models by pure statistical methods is impossible. This makes the research on new detection methods particularly important, i.e., research on anomaly detection based on machine learning methods and deep learning methods.

HTTP traffic data consists of hundreds or thousands of features. The rich network characterization methods can be found in the literature [5], [6]. Usually these features contain a large number of unrelated and redundant features, which increase the complexity of the anomaly detection model. Mimura and Tanaka [7] proposed a language-based agent log detection method. This approach divides proxy logs into words and uses the words as feature vectors. They utilized paragraph vector to convert the words into feature vectors automatically. However, they ignored the complete

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

HTTP request context, namely contextual semantics problem, so it may not contribute the detection rate. In addition, it is difficult to collect a sufficient amount of HTTP traffic to train the model, and only a small number of samples can be marked as malicious. In order to produce a distinctive feature representation, the corpus must be effectively summarized. In order to effectively summarize the corpus, the important words of classification must be extracted.

To address these issues, we employed the Word2vec algorithm for extracting word vectors to solve the problem of contextual association in this article. Then, we calculated the TF-IDF value of each word to generate a complete weighted paragraph vector to reduce the dimension. Finally, we adopted the LightGBM and CatBoost algorithms to quickly and accurately train the anomaly detection model. During the experiment, we used three benchmark datasets to evaluate the performance of our model, namely, HTTP DATASET CSIC 2010, UNSW-NB15, and Malicious-URLs. We compared the prediction results of our model with those of other models at the present stage. Experimental results indicated that our model performed well on the three datasets above.

The main contributions of this article are summarized as follows:

1) The Word2vec algorithm is used to solve the problem of contextual semantics. At the same time, the weight calculation of TF-IDF is introduced to improve the detection rate and accuracy of the model.
2) Customizing the size of the paragraph vector resulted in the reduction of the redundancy and complexity of the vector representation method, and the training efficiency of the anomaly detection model is improved.

The remainder of this article is structured as follows. Section II describes the related work. Section III briefly reviews the characteristics of the selected technology to make this article rational. Section IV presents the proposed algorithm. Section V presents the experimental results and corresponding analysis. Finally, the conclusions are presented in Section VI.

II. RELATED WORK

This section provides related work in the areas of feature engineering and detection algorithms used for choosing a classifier for anomaly traffic detection.

A. FEATURE ENGINEERING

Feature engineering methods include feature selection and feature extraction [8]. The HTTP traffic feature selection is based on certain rules to select some features from existing HTTP traffic features to represent the original HTTP traffic data. Many feature selection techniques have been developed for network traffic over the years, for example, Maximum Information Coefficient [9], General Feature Selection [10], [11], the combination of discretization and Filter [12], Information Gain Ratio [13], [14], Restricted Boltzmann Machine [21], [22](RBM) [15], Improved-RBM [16], Improved-Grey Wolf Optimization [17], and Hybrid Feature Selection [18], [19]. These approaches remove irrelevant and redundant features. The HTTP traffic feature extraction is used to extract a new set of features from the original HTTP traffic feature set through function mapping. The biggest benefit of this method is it obtain the smallest new feature set through transformation [20]. The commonly used network traffic feature extraction algorithms are as follows: the metric learning [21], the linear discriminant analysis [22], and the principal component analysis [23], [24]. The above mentioned feature selection and feature extraction algorithms can greatly improve the performance of the algorithm. However, the detection accuracy is not high enough, because it ignores the relevance of the full request context. The imbalance of traffic data and the popularity of compound attacks pose higher requirements on the reduction of HTTP traffic dimensions.

Some methods are employed to extract important words from HTTP logs using some NLP techniques. Liu *et al.* [25] proposed an integrated Web intrusion detection system that combines feature analysis and support vector machine (SVM) optimization. They extracted the characteristics of key parameters based on the summed detection points, which summarize the data features of the Web attacks for solving the limitations of large network data traffic and high dimensionality in data feature extraction. Mimura and Tanaka [7] utilized paragraph vector to convert the words into feature vectors automatically. In this approach, paragraph vector provides feature vectors to discriminate between benign traffic and malicious traffic. After that, Mimura [26] made improvements to the previous method, he calculated the word importance score (Term Frequency, TF) from malicious and benign words, and extracted the top N important words, then used Doc2vec to train it into a feature vector. Therefore, before performing anomaly detection on a large amount of high-dimensional HTTP traffic data, it is necessary to optimize and reduce the characteristics of the HTTP traffic data.

B. DETECTION ALGORITHMS

In recent years, researchers have designed many anomaly detection models to protect the network against attacks perpetrated by malicious users against Web applications. Zhao *et al.* [27] introduced a semi-supervised discriminant auto-encoder, they understood the nature of the attack based on the generalized transformation features derived directly from the unknown Web environment and data. Gong *et al.* [28] proposed model uncertainty to evaluate predictions made by DeepLearning-based Web attack models. There are also many algorithm models that use deep learning to detect Web attacks: Long Short-Term (LSTM) [29], [30], Specially Designed Convolution Neural Network (SDCNN) [31], Character-Level Convolution Neural Networks (CLCNN) [32], Channel Boosted and Residual learning-based CNN (CBR-CNN) structure [33]. It can be continuously optimized during training and testing to extract

more accurate feature values, but training complex network nodes or layers is slow.

Boosting is mainly used to boost a weak classifier or weak learner with the aim of achieving a higher accuracy classifier [18]. Adaptive boosting (Adaboost) is the most popular boosting algorithm, it focus on the misclassified samples during training, weak classifiers with high accuracy have great weight. Yuan *et al.* [34] proposed a semi-supervised intrusion detection system (SS-IDS) by combining tri-training with three different Adaboost algorithms. Kamarudin *et al.* [18] introduced the Logitboost algorithm to solve the limitations of Adaboost in handling with noise and outliers. Although the proposed algorithm can achieve a detection rate close to 100% relative to Web attacks, the overall calculation speed is still slow. The introduction of the LightGBM [35] and CatBoost [36], [37] algorithm was designed as an alternative solution to address the limitations in dealing with parallelization and memory consumption.

III. KNOWLEDGE BACKGROUND

A. NLP TECHNIQUE

Word vector technology is used to convert words in natural language into dense vectors. Words with similar semantics have similar vector representations. Word2vec is a more classic language model. It is a method of word clustering, and it achieves the purpose of word semantic inference and sentence sentiment analysis. It trains a neural network to obtain the weight matrix of the network, which can quickly and efficiently train word vectors. The model includes continuous bag of words (CBOW) and Skip-Gram training models. As show in Fig.1 (a), CBOW predicts the probability of the headword by context, whereas in Fig. 1 (b) Skip-Gram predicts the probability of the context by headword. CBOW's training speed is several times faster than that of Skip-Gram. The accuracy for common words is slightly higher for CBOW; Skip-Gram works well with a small amount of training data, well represents uncommon words or phrases [38], [39].

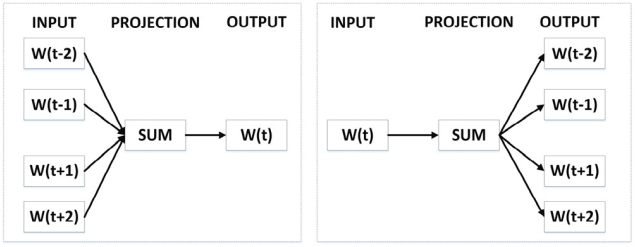

FIGURE 1. CBOW and Skip-Gram models.

TF-IDF is a commonly used weighting technique. It is a statistical method that is used to evaluate the importance of a word to a document set or a document in a corpus. If a word or phrase appears frequently in one article and rarely appears in other articles, that word or phrase has a good ability to distinguish categories.

B. LIGHTGBM AND CATBOOST

Traditional Gradient Boosting Decision Tree (GBDT) is faced with the adjustment of accuracy and efficiency with the geometric growth of data volume in recent years. To solve this problem, LightGBM [35] algorithm was proposed. It promotes the accuracy of prediction, greatly accelerates the prediction speed, and reduces the memory utilization [40]. LightGBM is a gradient boosting framework that uses tree-based learning algorithms. As shown in Fig. 2, compared with a traditional method like level-wise, the leaf-wise can reduce more losses when growing the same leaf. It transforms continuous eigenvalues into discrete histograms to reduce the feature dimension and improve the optimization of training speed [41]. In addition, LightBGM supports parallel tree enhancement operations, providing faster training speeds even on large data sets and can better adapt to the trend of sharp increase in HTTP traffic.

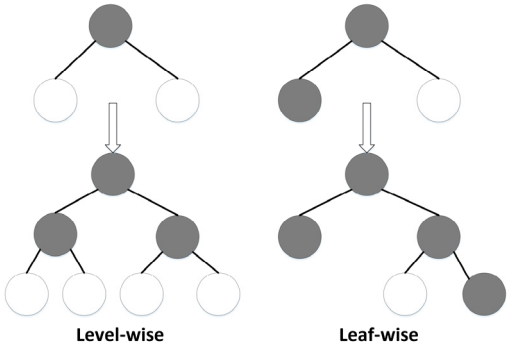

FIGURE 2. The generation strategy of tree in LightGBM [40].

CatBoost, like all standard gradient boosting algorithms, builds a new tree to fit the gradient of the current model. However, all classical lifting algorithms have overfitting problems caused by biased pointwise gradient estimation. CatBoost uses oblivious trees as basic predictors [36]. This tree is balanced and less likely to overfit. In order to use all the samples for training, CatBoost provides a solution, that is, first randomly sort all the samples, and then for a certain value of the categorical features, each feature of the sample is converted to a numeric type. The average value is based on the category labels ranked before the sample, and the priority and priority weighting factors are added at the same time [37]. This approach can reduce the noise caused by low-frequency sub-features in the category feature.

IV. OUR ALGORITHM

In the field of traffic anomaly detection, the pre-processing and feature extraction methods are generally used to solve the high-dimensional complexity of traffic data, and then the model and fine-tuning model are established through machine learning. In this research, our anomaly detection approach consists of three parts, namely, data preprocessing, feature extraction, and boosting classification algorithm.

Fig. 3 presents the proposed anomaly detection algorithm in detecting Web attacks supported by HTTP protocol.

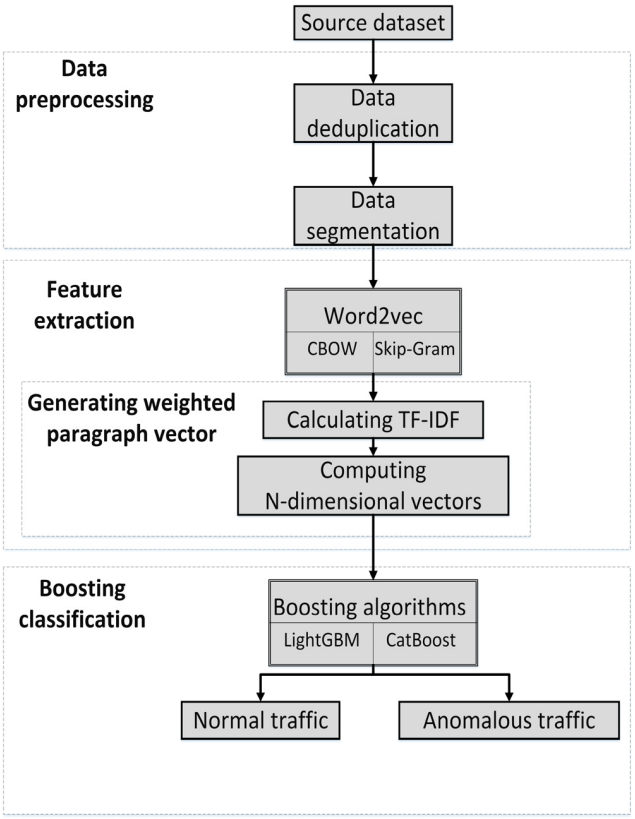

FIGURE 3. The proposed anomaly detection algorithm.

A. DATA PREPROCESSING

Datasets for network anomaly detection are very important for training and testing systems. For the next word vector training, we need to perform word segmentation on the data set. Fig.4 shows the data preprocessing process for three public datasets. This article focuses on the contextual dependencies of the processed data to better describe the purpose of the full request.

For the original HTTP DATASET CSIC 2010 datasets, the GET, POST, and PUT request data are extracted for detection. After requesting data extraction, string segmentation is performed on the data. The segmentation is performed according to the characteristics of the HTTP request. The next involves URL decoding and replacing special symbols with English space characters. For anomaly detection, information such as User-Agent, Pargma, and Cookie in the HTTP request header has little effect in distinguishing abnormal from normal request traffic. Therefore, this article only extracts fields, such as Method, Host, Path, and Parameters.

The Malicious-URLs dataset is similar to HTTP DATASET CSIC 2010 dataset. We extract the normal URL and the malicious URL separately and re-save them into new files. Next, we perform URL decoding and replace special

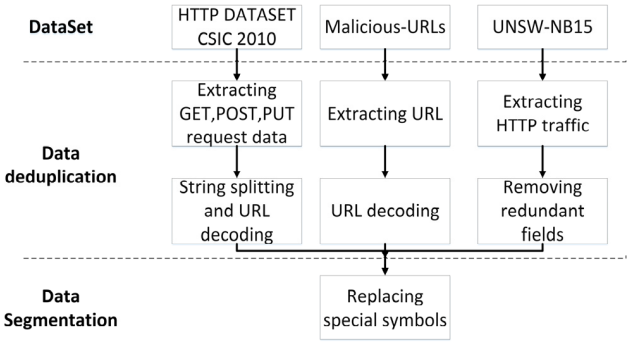

FIGURE 4. Data preprocessing steps diagram for three public datasets.

symbols with English space characters. Through experiments on the above two data sets, we found that the "&" symbol has an important effect on the parameter field. Thus, the "&" symbol is not replaced.

On the other hand, in UNSW NB-15, 8,287 instances in the training data and 18,724 instances in the testing data are based on HTTP traffic. We filter features with the same field value, including proto, service, is_ftp_login, ct_ftp_cmd, and is_sm_ips_ports. In addition, the source TCP window advertisement value "swin," destination TCP window advertisement value "dwin," source TCP base sequence number "stcpb," and destination TCP base sequence number "dtcpb" are not significant for recognizing Web attacks supported by HTTP protocol. Therefore, we also filter the above mentioned features.

B. FEATURE EXTRACTION

In this article, we combine normal and abnormal traffic requests into a corpus. Then, we use the Word2vec algorithm to train each word in the corpus and convert each word into a 300-dimensional vector. The HTTP DATASET CSIC 2010 dataset includes high-frequency words. So, we use the CBOW model for training. However, the Malicious-URLs dataset and UNSW NB-15 dataset includes low-frequency words. Thus, we use the Skip-Gram model for training. Out-of-vocabulary (OOV) problems occur due to too little corpus. We initially adopted FastText [42] to deal with OOV, but the detection effect was not good. Then, we continue to train the word using the word2vec model and use custom random vectors for low frequency words. Although random vectors will be a loss of some training effect here, the overall detection effect is still good.

Word2vec mean model is a text file representation method to characterize the content of text files. The Word2vec mean model averages the word vectors corresponding to each word in a text file to characterize the content of the text file. Equation (1) shows Word2vec mean model. Suppose the word vector is Model(word), then

$$V(doc) = \frac{\sum_{i=1}^{n} Model(word_i)}{n} \tag{1}$$

Among them, V(doc) represents the vector used to characterize the text file doc, and n is the number of words contained in doc.

TF-IDF is a classic statistical method for calculating word weights, which consists of two parts of data: TF and IDF. As show in formulas (2) and (3).

$$TF = \frac{t(word)}{s} \tag{2}$$

Among them, t(word) represents the number of occurrences of the word in the file, and s represents the sum of the number of occurrences of all words in the file.

$$IDF = \log \frac{len(M)}{1 + d[word]} \tag{3}$$

Among them, M represents the word frequency of the entire corpus, and d[word] represents the number of documents containing feature terms. To avoid the double counting of the number of documents containing feature items, we store this value in the dictionary after calculating it once.

Multiply formulas (2) and (3) to get the weight of text feature words. Equation (4) shows the calculation method of the original TF-IDF.

$$TF - IDF = TF \times IDF \tag{4}$$

In text classification research, the text in the text library is usually marked into several different categories, and the TF-IDF algorithm only considers the total frequency of feature words in the entire text library, ignoring the distribution in the category, resulting in certain Words contributing to category judgment are missing. Therefore, this article proposes to introduce the TF-IDF algorithm of quasi-frequency variance. The quasi-frequency variance measures the distribution of words in different categories. The calculation formula of the quasi-frequency variance measurement is shown in formula (5).

$$\tau = \frac{\sqrt{\sum_{i=1}^{C} (\frac{d[word]}{C} - dc_i[word])^2}}{C} \tag{5}$$

Among them, τ represents the quasi-frequency variance of the word; C represents the number of text categories; d[word] is the number of documents in the entire text library containing feature terms; dc_i[word] is the number of documents containing the feature terms in the category C_i. The larger the τ, the greater the fluctuation of the word in the category, the more uneven the distribution, and the greater the judgment effect on the category, so the calculation of the TF-IDF algorithm based on the variance of the class frequency is shown as below.

$$TF - IDF_\tau = TF \times IDF \times \tau \tag{6}$$

In this article, in order to get the weight of each word, we calculate TF-IDF manually and use equation (6) to calculate the TF-IDF value. Algorithm 1 presents the detailed process of calculating TF-IDF. The while loop (step 1-10) takes $O(n^2 * m)$ time in the worst case. Where n and m denote the total number of document and the length of each request, respectively. The other for loops will iterate n*m times to read each word, which will take O(n*m) time in the worst case. Therefore, The overall computational complexity of the algorithm turns out to be $O(n^{2*}m)$, whereas data dictionary takes up O(n) space. Calculating TF-IDF is the highest time complexity in this scheme, but accurately calculating the weight of each word can greatly increase the detection rate.

Algorithm 1 Calculating TF-IDF

Input: Data set D, word
Output: TF-IDF value
Step1. Calculate t, s.
Step2. Calculate TF.
Step3. Calculate M, MC. M is the word frequency of the entire corpus, and MC is the word frequency of the category C_i.
Step4. Calculate the number of documents containing feature terms and save it to dictionary d and dc_i.

```
for text in D do
  for word in text do
    if word not in d then
      d[word] = sum(1 for c in M if word in c)
    end if
    if word not in dc_j then
      dc_i[word] = sum(1 for c in MC_i if word in c)
    end if
  end for
end for
```

Step 5. Calculate IDF value.
Step 6. Calculate τ value.

```
if dc_i[word] == NULL then
τ = 1
end if
return TF × IDF × τ
```

The core innovation of this article is the way by which word vectors are effectively represent as paragraph vectors. We multiply the previously trained word vector by the TF-IDF value, add them, and divide them by the number of words to get the paragraph vector. The detailed process are shown in Algorithm 2. The general meaning is shown in Fig 5. For instance, the result of the weighted average of {0.89,...,0.29,...,0.59} is 0.09. It is an efficient vector mapping algorithm (with complexity of O(N*n)). Where n represents the length of each request, and N can be set freely, which is equivalent to a constant. Therefore, The overall computational complexity of the algorithm turns out to be O(n) and the temporary variable takes up O(n) space.

C. BOOSTING CLASSIFICATION ALGORITHMHI

In this section, the proposed ensemble classification method based on a boosting algorithm is described. We use the boosting algorithm LightGBM and CatBoost to boost

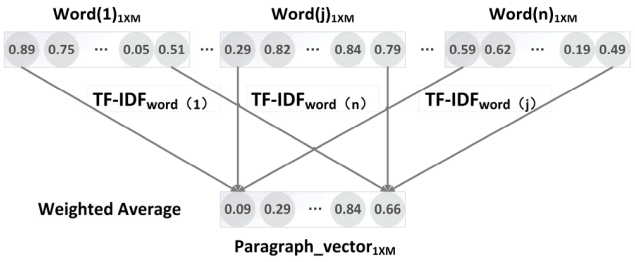

FIGURE 5. Paragraph vector generation diagram.

Algorithm 2 Generating Weighted Paragraph Vector

Input: Data set D, word

Output: paragraph vector

Step1. Initialize N(N is the dimension of the paragraph vector), Load data set D.

Step2. Extract terms Word$_j$ from each sample obtained Doc(i)∈D (Doc is a paragraph of variable size composed of PATH, PARAMETERS, etc.),

$0 < i \leq |D|, 0 < j \leq |Doc(i)|$.

1. **if** Model(Word$_j$) == NULL **then**
2. Model(Word$_j$) = np.random.uniform(0, 1, 300)
3. **end if**

Step3. Calculate the TF-IDF$_j$ value of each word as described in Algorithm 1.

Step4. Compute N-dimensional vectors.

$$paragraph\_vector = \frac{\sum_{n}^{N} \sum_{j}^{|Doc(i)|} (TF - IDF(\text{Word}_j) \bullet Model(Word_j))[n]}{|Doc(i)|}$$

4. **for** n in range N **do**
5. temp = 0
6. **for** j in |Doc(i)| **do**
7. temp+ = (TF-IDF(Word$_j$)∗ Model(Word$_j$))[n]
8. **end for**
9. temp = temp/|Doc(i)|
10. **end for**

// The size of paragraph_vector is N (N can be set according to the actual situation).

classification. Through preliminary experiments and literature research, these two algorithms are more suitable for improving training efficiency and detection accuracy than the widely used Adaboost algorithm. We consider a training data set with N samples and divided into two classes. In this study the two classes are abnormal and normal. The two classes are defined as $y \in \{0, 1\}$, i.e., samples in class $y = 0$ are instances of normal traffic, whereas those in $y = 1$ are the samples of abnormal traffic. We let the set of training data be $\{(x_1, y_1),\ldots, (x_i,y_i),\ldots, (x_n, y_n)\}$, where x_i is the feature vector, and y_n is the target class. The pseudo code of Algorithm 3 shows the implementation of Boosting algorithms. The conditional operators take O(1) time. Next, the for loop iteratively judges the prediction result until n times which takes O (n) time in the worst case. Therefore, The overall computational complexity of the algorithm turns out to be O(n), whereas the overall space complexity is only O(1). Gradient Boosting is an efficient implementation that can accelerate and reduce memory consumption.

Algorithm 3 Boosting Classification

Input: Training data named x, training label named y, boosting algorithms named type

Output: The score result, include accuracy_score and confusion_matrix

1. tra_x, test_x, tra_y, test_y = train_test_split(x, y)
2. **if** type == 'LightGBM'**then**
3. tra_x1, val_x1, tra_y1, val_y1 = train_test_split(tra_x, tra_y)
4. train = lightgbm.Dataset(tra_x1, tra_y1)
5. eval = lightgbm.Dataset(val_x1, val_y1)
6. model = lightgbm.train(train, valid_sets = [eval,train])
7. **end if**
8. **if** type == 'CatBoost' **then**
9. model = CatBoostClassifierfit(tra_x, tra_y)
10. **end if**
11. y_pred = model.predict(test_x)
12. **for** i in range(0, len(y_pred)) **do**
13. **if** y_pred[i] >= 0.5 **then**
14. y_pred[i] = 1
15. **else**
16. y_pred[i] = 0
17. **end if**
18. **end for**
19. return score(test_y, y_pred)

In a nutshell, the computational complexity of the design model depends on the sum of the computational complexities of the above algorithms. Therefore, the overall time and space complexity is computed as follows.

$$T(C) = O(n^{2*}m) + O(n) + O(n) => O(n^{2*}m)$$
$$S(C) = O(n) + O(n) + O(1) => O(n)$$

V. EXPERIMENTAL RESULTS AND ANALYSIS

This section evaluates the performance of the proposed methods by performing various experiments on three public standard intrusion detection datasets. The experiments were conducted on a 2.4 GHz Intel Core i5 with 8GB RAM running on Windows 10 operating system.

A. DATA DESCRIPTION

Data constitute the basis of computer network security research. The accurate choice and reasonable use of data are the prerequisites for conducting network security research. The datasets like DARPA or KDD'99 are outdated

and do not cover many of the current attacks [43]. From the few public datasets, we choose HTTP DATASET CSIC 2010 [44], UNSW-NB15 ()[45], and Malicious-URLs [46] as our experiment datasets. Three different benchmark datasets have been used to perform extensive performance evaluation. These datasets vary in size and dimensionality.

HTTP DATASET CSIC 2010 dataset was produced by the Spanish Research National Council. It contains already marked requests for Web services. The data set is automatically generated. It contains 72,000 normal requests and more than 25,000 abnormal requests, where 36,000 are normal data for training, and 36,000 are normal data for testing. As shown in Table 1. The dataset contains many main types of attacks, such as SQL injection, buffer overflow, information collection, file disclosure, CRLF injection, XSS, server-side inclusion, parameter tampering, and so on.

TABLE 1. Distribution of HTTP DATASET CSIC 2010 dataset.

Data Set	Normal Data		Anomaly Data	
CSIC 2010	Train 36,000	Test 36,000	Train 0	Test 25,065

The UNSW-NB 15 was simulated by using the IXIA PerfectStorm tool in the Cyber Range Lab at the Australian Centre for Cyber Security, which extracts a total of 49 features to reflect the nature of network traffic, including 5 stream features, 13 basic features, 8 content features, 9 time features, 5 general features, 7 connection features. UNSW-NB15 is more complex than KDD99 and is considered as a new benchmark data set for evaluating NIDSs [47]. The full dataset contains captured raw traffic of 100GB with nine synthetic types of attacks, namely, Backdoors, DoS, Analysis, Fuzzers, Generic, Worms, Reconnaissance, Shellcode and Exploits [18]. In these sets of traffic, 9,361 instances in the normal data and 17,650 instances in the anomaly data are based on HTTP traffic, as presented in Table 2.

The Malicious-URLs dataset is a project on Github that uses machine learning to detect malicious URL. It is a very effective method to detect abnormal traffic by identifying whether the domain name of the Host is a malicious domain name. If the host domain name communicating with a host or server comes from a malicious domain name, we can use this information to determine that the request is a malicious request. Either normal URL or malicious URL are present for tagging. The total traffic captured is 420,464. These data are divided into two sets, namely, normal and abnormal, which consist of 344,821 and 75,643 instances of traffic, respectively. As shown in Table 3, this dataset has a larger amount of training data than the two previous public datasets.

B. EVALUATION METHOD

True Positive (TP): real normal traffic and the model classification result is also normal traffic. False Negative (FN): real normal traffic but the model classification result is attack traffic. False Positive (FP): real attack traffic but the model classification result is normal traffic. True Negative (TN): real attack traffic and model classification results are also attack traffic.

TABLE 2. Distribution of HTTP traffic for UNSW-NB 15 dataset.

Data Set	Normal Data		Anomaly Data	
UNSW -NB 15	Train 4,013	Test 5,348	Train 4,274	Test 13,376

TABLE 3. Distribution of Malicious-URLs dataset.

Data Set	Normal Data	Anomaly Data
Malicious -URLs	344,821	75,643

To better evaluate the performance of the proposed method in this article, evaluation is performed by using various indicators such as Accuracy (ACC), true positive rate (TPR), and false positive rate (FPR), which are defined as follows:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{7}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{8}$$

$$\mathrm{FPR} = \frac{FP}{FP + TN} \tag{9}$$

C. EFFECTS OF OUR ALGORITHM STRATEGIES

In order to ensure the validity of the experiment, we try to maintain the consistency of parameters that are not involved in the comparison. We have set the test sample ratio to 10%. When we experiment with one parameter, we choose the best value for other parameters.

1) WORD SEGMENTATION CONTRAST

When data preprocessing is used for word segmentation, all special characters are replaced with spaces, but some special characters have special meanings, such as "&". In order to quickly perform comparison experiments, we took the same amount of data as the HTTP DATASET CSIC 2010 dataset, that is, only a part of its dataset, because the Malicious-URLs dataset is relatively large. Table 4 shows that the "&" symbol has an important effect on the parameter field, and keeping the "&" symbol can achieve higher accuracy, higher TPR, and lower FAR than deleting it. The injection attack generally carry more parameters, and the "&" symbol can reflect this feature.

2) WORS2VEC'S ITERATION COUNT

The core innovation of this article is to propose a weighted Word2vec paragraph vector. Thus, we first compare the parameters of the Word2vec word vector training model. Wors2vec's iteration count defaults to 5. We gradually

TABLE 4. Word segmentation contrast experiment.

Algorithm	Special symbol	HTTP DATASET CSIC 2010			Malicious-URLs		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	Delete "&"	98.69%	99.28%	3.03%	94.37%	96.29%	8.36%
	Keep "&"	**99.40%**	**99.73%**	**1.57%**	**95.11%**	**97.13%**	**7.77%**
CatBoost	Delete "&"	98.71%	99.33%	3.11%	94.18%	95.71%	8.01%
	Keep "&"	**99.49%**	**99.82%**	**1.45%**	**94.34%**	**95.96%**	**7.97%**

TABLE 5. Performance for different iteration count.

Algorithm	Iter	HTTP DATASET CSIC 2010			UNSW-NB 15		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	5	98.56%	99.48%	4.12%	95.15%	89.98%	2.09%
	25	98.96%	99.57%	2.79%	99.25%	98.29%	0.22%
	50	99.13%	99.65%	2.38%	**99.40%**	**98.61%**	**0.17%**
	100	99.28%	99.72%	1.98%	99.37%	98.50%	**0.17%**
	500	99.36%	99.72%	1.70%	98.88%	97.65%	0.45%
	1000	**99.40%**	**99.73%**	**1.57%**	97.33%	93.82%	0.79%
CatBoost	5	98.94%	99.62%	3.03%	94.89%	90.09%	2.55%
	25	99.23%	99.65%	1.98%	98.70%	97.33%	0.56%
	50	99.35%	99.80%	1.98%	99.14%	**98.29%**	0.39%
	100	99.37%	99.70%	1.61%	**99.18%**	98.18%	**0.28%**
	500	99.43%	99.76%	1.53%	98.11%	95.84%	0.68%
	1000	**99.49%**	**99.82%**	**1.45%**	96.18%	91.05%	1.07%

TABLE 6. Performance for different implementation.

Algorithm	Implementation	HTTP DATASET CSIC 2010			UNSW-NB 15		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	CBOW	**99.40%**	99.73%	**1.57%**	94.30%	88.92%	2.83%
	Skip-Gram	99.34%	**99.75%**	1.86%	**99.40%**	**98.61%**	**0.17%**
CatBoost	CBOW	**99.49%**	**99.82%**	**1.45%**	94.74%	89.98%	2.72%
	Skip-Gram	99.44%	99.76%	1.49%	**99.14%**	**98.29%**	**0.39%**

increase the number of iterations. As shown in Table 5, in the HTTP DATASET CSIC 2010 dataset, with increasing iteration count from "5" to "1,000," TPR and ACC have increased gradually, whereas FPR has decreased progressively. The LightGBM algorithm is generally less accurate than the CatBoost algorithm. For the UNSW-NB 15 dataset, our approach achieves the best performance when the iteration count is 50 and 100. Considering the training time, we choose 50 as the best iteration count. Furthermore, the LightGBM algorithm performs better than the CatBoost algorithm on this data set, contrary to the performance on the previous data set.

3) CBOW AND SKIP-GRAM

CBOW and Skip-Gram are implemented in word2vec for vector representation of text. The second section of this article describes their differences. As presented in Table 6, CBOW is more suitable for the HTTP DATASET CSIC 2010 dataset, because it has high-frequency words, and the accuracy for common words is slightly higher for CBOW. By contrast, Skip-Gram is better for the UNSW-NB 15 dataset due to its low-frequency words, and because Skip-Gram represents uncommon words or phrases well.

4) DIMENSION OF THE PARAGRAPH VECTOR

The dimension of the paragraph vector is the most important parameter of the algorithm in this article. Therefore, we must measure the effect of the dimension of the paragraph vector on the method performance and determine the best dimension of the paragraph vector via multiple evaluation experiments. In our experiments, the dimension of the paragraph vector range from 50 to 300, because the dimension of the word vector training model is 300. As shown in Table 7, those metrics achieve the best results when the dimension of the paragraph vector is 150 to 300. In most cases, when the dimension of the paragraph vector increases to 300, the performance decreases instead. So, the dimension of the paragraph vector corresponding to it is steadily chosen to be 250.

5) DIFFERENT WORD VECTOR TRAINING MODELS

Table 8 shows the experimental results in the same dimension (300) using different word vector training models.

TABLE 7. Effect of different dimension of the paragraph vector (N is the dimension of the paragraph vector).

Algorithm	N	HTTP DATASET CSIC 2010			UNSW-NB 15		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	50	99.09%	99.54%	2.22%	98.40%	97.23%	0.96%
	100	99.20%	99.61%	1.98%	99.11%	98.40%	0.51%
	150	99.27%	99.72%	2.02%	99.07%	98.40%	0.56%
	200	99.32%	99.65%	1.65%	99.22%	98.18%	0.22%
	250	**99.40%**	**99.73%**	**1.57%**	**99.40%**	**98.61%**	**0.17%**
	300	99.37%	99.70%	1.61%	99.33%	**98.61%**	0.28%
CatBoost	50	99.26%	99.58%	1.65%	98.26%	97.12%	1.13%
	100	99.34%	99.68%	1.65%	98.88%	97.65%	0.45%
	150	**99.51%**	**99.83%**	1.41%	99.03%	97.97%	0.39%
	200	99.44%	99.69%	**1.29%**	98.85%	97.76%	0.56%
	250	99.49%	99.82%	1.45%	99.14%	98.29%	0.39%
	300	99.47%	99.79%	1.45%	**99.33%**	**98.72%**	**0.34%**

TABLE 8. Effectiveness of different word vector training models.

Algorithm	Model	HTTP DATASET CSIC 2010			UNSW-NB 15		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	Word2vec	**99.36%**	**99.72%**	**1.70%**	**99.40%**	**98.61%**	**0.17%**
	GloVe	98.81%	99.61%	3.52%	94.41%	90.20%	3.34%
	FastText	98.96%	99.62%	2.95%	86.93%	67.94%	2.94%
CatBoost	Word2vec	**99.43%**	**99.76%**	**1.53%**	**99.14%**	**98.29%**	**0.39%**
	GloVe	99.19%	99.75%	2.44%	94.41%	90.41%	3.46%
	FastText	99.24%	99.72%	2.14%	86.89%	70.07%	4.14%

The hyperparameters, such as the number of iterations and sliding window size, are kept as same as possible. For more rapid comparison of the differences, we set the number of iterations to 500 instead of 1000 for the HTTP DATASET CSIC 2010. Furthermore, for the UNSW-NB 15 dataset, we choose the best iteration count of 50. As shown in Table 8, the Word2vec method achieves better results than the other two word vector training methods. For the HTTP DATASET CSIC 2010 dataset, the three word vector models performed well. For the UNSW-NB 15 dataset, the accuracy and TPR are less than 90%, and the FAR is high when FastText is used. Therefore, solving the OOV problem in this article is not suitable. In addition to its higher accuracy than the other two word vector models, Word2vec has low CPU memory consumption and is easy to use, whereas GloVe [48] or FastText consumes high memory and has a long training time.

6) TF-IDF PERFORMANCE EVALUATION

The TF-IDF performance evaluation section shows the classification results using the average of Word2vec, TF-IDF, TF-IDFτ. As shown in Table 9, using the TF-IDF algorithm compared with the Word2vec algorithm, the accuracy rate increased, and using the TF-IDFτ algorithm, the accuracy rate increased more. Similarly, the false positive rate has dropped even more. This shows that the Word2vec mean model weighted by the TF-IDFτ algorithm can effectively improve the detection ability.

D. EVALUATION RESULTS AND COMPARISONS

The above mentioned analysis shows that even though many differences exist between the three datasets, both boosting algorithm achieve higher accuracy, higher TPR, and lower FAR when choosing the best parameter. Moreover, in the field of anomaly detection, the detection rate and accuracy are close to 100%, which can well protect the entire Web environment and ensure the safe operation of the network environment. At the same time, because the research in this article is based on the normal and abnormal binary classification problem, the deformation or new-type attack detection problem can be solved.

In the real world, traffic data are very large. To test the robustness of our proposed approach, we use Malicious-URLs dataset to evaluate the detection effect of the algorithm model. Table 10 shows the best experimental results of our algorithm on the three datasets. For the Malicious-URLs dataset with the largest amount of data, the proposed method can achieve best detection effect. In addition, our training is fast. Therefore, our algorithm has good stability in handling the new testing dataset and keeps the true and false positives in a reasonable range.

The research on Web attack detection has always been the research focus of many scholars at home and abroad, and has achieved fruitful results. The relative comparisons of the proposed scheme against the existing schemes has been tabulated in Table 11-13.

TABLE 9. TF-IDF performance evaluation.

Algorithm	Model	HTTP DATASET CSIC 2010			UNSW-NB 15		
		ACC	TPR	FPR	ACC	TPR	FPR
LigthtGBM	Word2vec	98.68%	99.48%	3.68%	98.14%	97.23%	1.36%
	Word2vec +TF-IDF	99.07%	**99.86%**	3.23%	**99.40%**	**98.72%**	0.22%
	Word2vec +TF-IDFτ	**99.40%**	99.73%	**1.57%**	**99.40%**	98.61%	**0.17%**
CatBoost	Word2vec	98.56%	99.53%	4.25%	97.70%	96.80%	1.81%
	Word2vec +TF-IDF	98.93%	99.69%	3.27%	99.03%	98.18%	0.51%
	Word2vec +TF-IDFτ	**99.49%**	**99.82%**	**1.45%**	**99.14%**	**98.29%**	**0.39%**

TABLE 10. Best performance of our algorithm for three public dataset.

Data Set	ACC	TPR	FPR	Paragraph Vectors Generate	Boosting Algorithm Training
Malicious -URL	99.62%	99.82%	1.32%	358min	580sec
CSIC 2010	99.49%	99.82%	1.45%	20min	312sec
UNSW-NB 15	99.40%	98.61%	0.17%	27min	25sec

TABLE 11. Comparison with other published methods for CSIC 2010.

Test_number	Method	ACC	TPR	FPR
10%	CLCNN	98.80%	N/A	N/A
	J48	N/A	95.97%	3.54%
	Proposed Method	**99.49%**	**99.82%**	**1.45%**
25%	SDCNN	96.49%	93.35%	**1.37%**
	Proposed Method	**99.16%**	**99.72%**	2.43%

Table 11 shows that compared with the methods of CLCNN [32], J48 [43], and SDCNN [31], the proposed method achieves the highest accuracy and highest TPR on HTTP DATASET CSIC 2010 dataset. Similar findings were observed on the other two benchmark datasets. The reported result comparisons are not necessarily very accurate due to the fact that comprehensive resemblance is not an easy task, as different researchers have used different proportions of traffic types, sampling methods, computational time, and pre-processing methods. However, we try to keep the known parameters the same.

Table 12 shows the results of comparison with the advanced method of Artificial Bee Colony and Artificial Fish Swarm (ABC-AFS) [49] and Logitboost [18] for the UNSW-NB 15 dataset. Compared with the existing scheme, the accuracy is improved by up to 9%. The reported result comparisons are for reference only, because this article only extracted the HTTP traffic in this dataset.

TABLE 12. Comparison with other published methods for UNSW-NB 15.

Method	ACC	TPR	FPR
ABC-AFS	98.90%	98.60%	**0.13%**
Logitboost	90.33%	89.80%	8.22%
Proposed Method	**99.40%**	**98.61%**	0.17%

TABLE 13. Comparison with original method for Malical-URLs.

Method	ACC	TPR	FPR
LR	98.62%	96.64%	**0.37%**
Proposed Method	**99.62%**	**99.82%**	1.32%

Furthermore, the Malical-URLs dataset on Github uses Logistic Regression (LR) to detect malicious URLs. As shown in Table 13, although the false alarm rate is not lower than the original method, the ACC and TPR have improved.

Hence, it can be concluded from the obtained results that the proposed scheme is effective in anomaly detection in HTTP traffic, against the current state-of-the-art schemes.

VI. CONCLUSION

In this article, we proposed an efficient machine learning approach for detecting anomalous HTTP traffic and the NLP technique was employed for building the effective feature vectorizations, which can be used to reduce training complexity of the boosting algorithms. The main advantages of this method are as follows: it uses the Word2vec algorithm to solve the semantic gap of each entry in HTTP traffic, and employs the TF-IDF algorithm weighting technology that highlights the distinguishing ability of keywords in the entire request. Moreover, the average idea is used to convert the complete request into paragraph units to achieve detection. The size of the vector is set to avoid the redundant expression of the vector by other algorithms while controlling the complexity of the entire model training. We have

evaluated the proposed method using three public dataset namely HTTP DATASET CSIC 2010, UNSW-NB15, and Malicious-URLs. The experiments have shown that the proposed method achieves satisfactory results. Compared with the current state-of-the-art schemes, the ACC and TPR is improved by 1% to 9%, and the FPR is kept within a reasonable range.

When detection is performed on the same type of data, the larger training set is, the higher expected detection accuracy and the lower training speed will be. Therefore, under the condition of ensuring a certain detection accuracy, speeding up the training of the detector is an important future direction for the research on anomaly traffic detection. In addition, the network information is updated very quickly, and it needs to retrain the model quickly for a long time, so we will study the incremental model of Word2vec in the future.

REFERENCES

[1] OWASP Foundation. *OWASP Top 10-2017*. Accessed: Nov. 29, 2017. [Online]. Available: http://www.owasp.org.cn/owasp-project/OWASP Top102017v1.1.pdf

[2] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[3] S. E. Smaha, "Haystack: An intrusion detection system," in *Proc. 4th Aerosp. Comput. Secur. Appl.*, Dec. 1988, pp. 37–38.

[4] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 810–820, Jul. 2002.

[5] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[6] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.

[7] M. Mimura and H. Tanaka, "Leaving all proxy server logs to paragraph vector," *J. Inf. Process.*, vol. 26, pp. 804–812, Dec. 2018.

[8] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, and H. Wan, "Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction," *Expert Syst. Appl.*, vol. 150, Jul. 2020, Art. no. 113277.

[9] M.-H. Chen, P.-C. Chang, and J.-L. Wu, "A population-based incremental learning approach with artificial immune system for network intrusion detection," *Eng. Appl. Artif. Intell.*, vol. 51, pp. 171–181, May 2016.

[10] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, S. Petrovic, and K. Franke, "Applying feature selection to payload-based Web application firewalls," in *Proc. 3rd Int. Workshop Secur. Commun. Netw. (IWSCN)*, May 2011, pp. 75–81.

[11] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović, and K. Franke, "Application of the generic feature selection measure in detection of Web attacks," in *Computational Intelligence in Security for Information Systems*. Cham, Switzerland: Springer, 2011, pp. 25–32.

[12] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, "A combination of discretization and filter methods for improving classification performance in KDD cup 99 dataset," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 359–366.

[13] P. Singh and A. Tiwari, "An efficient approach for intrusion detection in reduced features of KDD99 using ID3 and classification with KNNGA," in *Proc. 2nd Int. Conf. Adv. Comput. Commun. Eng.*, May 2015, pp. 445–452.

[14] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," *Appl. Intell.*, vol. 48, no. 10, pp. 3193–3208, 2018.

[15] S. Garg, K. Kaur, S. Batra, G. S. Aujla, G. Morgan, N. Kumar, A. Y. Zomaya, and R. Ranjan, "En-ABC: An ensemble artificial bee colony based anomaly detection scheme for cloud environment," *J. Parallel Distrib. Comput.*, vol. 135, pp. 219–233, Jan. 2020.

[16] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 566–578, Mar. 2019.

[17] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 924–935, Sep. 2019.

[18] M. H. Kamarudin, C. Maple, T. Watson, and N. S. Safa, "A LogitBoost-based algorithm for detecting known and unknown Web attacks," *IEEE Access*, vol. 5, pp. 26190–26200, 2017.

[19] H. Li, W. Guo, G. Wu, and Y. Li, "A RF-PSO based hybrid feature selection model in intrusion detection system," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 795–802.

[20] 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. 22, pp. 949–961, Sep. 2017.

[21] J. Gupta and J. Singh, "Detecting anomaly based network intrusion using feature extraction and classification techniques," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 1453–1456, 2017.

[22] A. A. Aburomman and M. Bin Ibne Reaz, "Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection," in *Proc. IEEE Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, Oct. 2016, pp. 636–640.

[23] K. Keerthi Vasan and B. Surendiran, "Dimensionality reduction using principal component analysis for network intrusion detection," *Perspect. Sci.*, vol. 8, pp. 510–512, Sep. 2016.

[24] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, Mar. 2019.

[25] C. Liu, J. Yang, and J. Wu, "Web intrusion detection system combined with feature analysis and SVM optimization," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, p. 33, Dec. 2020.

[26] M. Mimura, "Adjusting lexical features of actual proxy logs for intrusion detection," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102408.

[27] F. Zhao, H. Zhang, J. Peng, X. Zhuang, and S.-G. Na, "A semi-self-taught network intrusion detection system," *Neural Comput. Appl.*, pp. 1–11, Apr. 2020.

[28] X. Gong, J. Lu, Y. Zhou, H. Qiu, and R. He, "Model uncertainty based annotation error fixing for Web attack detection," *J. Signal Process. Syst.*, pp. 1–13, Feb. 2020.

[29] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, *Long Short Term Memory Networks for Anomaly Detection in Time Series*. vol. 89. Louvain-la-Neuve, Ottignies-Louvain-la-Neuve, Belgium: Presses universitaires de Louvain, 2015.

[30] A. Bochem, H. Zhang, and D. Hogrefe, "Streamlined anomaly detection in Web requests using recurrent neural networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 1016–1017.

[31] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin, "A deep learning method to detect Web attacks using a specially designed CNN," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 828–836.

[32] M. Ito and H. Iyatomi, "Web application firewall using character-level convolutional neural network," in *Proc. IEEE 14th Int. Colloq. Signal Process. Appl. (CSPA)*, Mar. 2018, pp. 103–106.

[33] N. Chouhan, A. Khan, and H.-U.-R. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105612.

[34] 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, Jun. 2019, Art. no. 155014771984605.

[35] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.

[36] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6638–6648.

[37] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," 2018, *arXiv:1810.11363*. [Online]. Available: https://arxiv.org/abs/1810.11363

[38] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Workshop ICLR*, Jan. 2013, pp. 1–12.

[40] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019.
[41] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A multi-layer hardware trojan protection framework for IoT chips," *IEEE Access*, vol. 7, pp. 23628–23639, 2019.
[42] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: Compressing text classification models," 2016, *arXiv:1612.03651*. [Online]. Available: https://arxiv.org/abs/1612.03651
[43] R. Kozik, M. Choraś, R. Renk, and W. Hołubowicz, "A proposal of algorithm for Web applications cyber attack detection," in *Proc. IFIP Int. Conf. Comput. Inf. Syst. Ind. Manage.* Cham, Switzerland: Springer, 2015, pp. 680–687.
[44] (2010). *HTTP DATASET CSIC*. [Online]. Available: http://www.isi.csic.es/dataset
[45] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[46] Faizan Ahmad. *Using-Machine-Learning-to-Detect-Maclicious-URLs*. Accessed: Feb. 18, 2017. [Online]. Available: https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs
[47] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., A Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[48] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[49] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Comput. Netw.*, vol. 136, pp. 37–50, May 2018.

JIELING LI was born in Fujian, China, in 1995. He is currently pursuing the master's degree in computer with Fuzhou University. His research interests include machine learning and cyberspace security.

HAO ZHANG was born in Anhui, China, in 1981. He received the B.S. and M.S. degrees in computer science from the University of Electronic Science and Technology of China, in 2002 and 2006, respectively, and the Ph.D. degree in applied mathematics from Fuzhou University, China, in 2015. He is currently an Associate Professor with the College of Mathematics and Computer Science, Fuzhou University. His research interests include artificial intelligence, machine learning, and cyberspace security.

ZHIQIANG WEI was born in Fujian, China, in 1994. He received the M.S. degree from Fuzhou University, in 2020. His research interests include artificial intelligence, machine learning, and cyberspace security.