Hindawi
Security and Communication Networks
Volume 2018, Article ID 4943509, 9 pages
https://doi.org/10.1155/2018/4943509

Research Article

TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest

Erxue Min,[1] **Jun Long**,[1] **Qiang Liu**,[1] **Jianjing Cui**,[1] **and Wei Chen**[2]

[1]*College of Computer, National University of Defense Technology, Changsha 410073, China*
[2]*School of Computer Science, University of Birmingham, Birmingham, British B15 2TT, UK*

Correspondence should be addressed to Erxue Min; mex338@qq.com

Received 3 May 2018; Accepted 24 June 2018; Published 5 July 2018

Academic Editor: Zhaoqing Pan

As we head towards the IoT (Internet of Things) era, protecting network infrastructures and information security has become increasingly crucial. In recent years, Anomaly-Based Network Intrusion Detection Systems (ANIDSs) have gained extensive attention for their capability of detecting novel attacks. However, most ANIDSs focus on packet header information and omit the valuable information in payloads, despite the fact that payload-based attacks have become ubiquitous. In this paper, we propose a novel intrusion detection system named TR-IDS, which takes advantage of both statistical features and payload features. Word embedding and text-convolutional neural network (Text-CNN) are applied to extract effective information from payloads. After that, the sophisticated random forest algorithm is performed on the combination of statistical features and payload features. Extensive experimental evaluations demonstrate the effectiveness of the proposed methods.

1. Introduction

Due to the advancements in Internet, cyberspace security has gained increasing attention [1, 2], which has encouraged many researchers to design effective defense systems called Network Intrusion Detection Systems (NIDSs). Currently, existing intrusion detection techniques fall into two main categories: misuse-based detection (also known as signature-based detection or knowledge-based detection) and anomaly-based detection (also known as behavior-based detection). Misuse-based detection systems extract the discriminative features and patterns from known attacks and hand-code them into the system. These rules are compared with the traffic to detect attacks. They are effective and efficient for detecting known type of attacks and have a very low False Alarm. Therefore, Misuse-based detection systems are currently the mainstream NIDSs and some sophisticated ones have been deposited in real scenarios, e.g., snort [3]. However, misuse detection systems require updating the rules and signatures frequently and they are incapable to identify any novel or unknown attacks. In recent years, anomaly-based network intrusion detection systems (ANIDSs) have attracted much attention for their capability of detecting zero-day attacks. They adopt statistical methods, machine learning algorithms, or data mining algorithms to model the pattern of normal network behavior and detect anomalies as deviations from normal behavior.

Various algorithms have been proposed to model network behavior and detect anomaly flows, including artificial neural networks [4], fuzzy association rules [5], Bayesian network [6], clustering [7], decision trees [8], ensemble learning [9], support vector machine [10], and so on [11, 12]. However, these methods mostly exploit the information in packet headers or the statistical information of entire flows and fail to detect the malicious content (e.g., SQL injection, cross-site scripting, and shellcode) in packet payloads. Classic processing methods for payloads can be divided into two categories. The first category requires prior knowledge of protocol formats, which cannot be applied to unknown protocols. The second category does not require expert domain knowledge; instead, they calculate some statistical features or conduct N-gram analysis, but they usually suffer from a high false positive rate. In recent years, deep learning algorithms [13, 14] have achieved remarkable results in many fields, e.g.,

Computer Vision (CV) [15], Natural Language Processing (NLP) [16], and Automatic Speech Recognition (ASR) [17]. They are proven to be capable to extract salient features from unstructured data. Considering the fact that the payloads of network traffic are sequence data similar to texts, we can apply modern deep learning techniques in NLP to the feature extraction of network payloads.

In this paper, we adopt word embedding [18] and text-convolutional neural network (Text-CNN) [19] to extract features from the payloads in network traffic. We combine the statistical features with payload features and then run random forest [20] for the final classification. The rest of this paper is organized as follows. In Section 2, we describe the related work. In Section 3, we describe the design and implementation of our methods. In Section 4, we show extensive experimental results to show the effectiveness of our methods. Finally, in Section 5, we conclude this paper.

2. Related Work

2.1. Payload-Based Intrusion Detection. In these days, payload-based attacks have become more prevalent, while older attacks such as network Probe, DoS, DDoS, and network worm attacks have become less popular. Many attacks place the exploit codes inside the payload of network packets; thus, header-based approaches cannot detect them. In this case, many payload-based detection techniques have been proposed. The first class of these methods is creating protocol parsers or decoders for different kinds of application. Snort [3] includes a number of protocol parsers for protocol anomaly detection. For example, the http_inspect preprocessor parses and normalizes HTTP fields, making them available to detect oversized header fields, non-RFC characters, or Unicode encoding. ALAD [21] builds models of allowed keywords in text-based application protocols such as FTP, HTTP, and SMTP. The anomaly score is increased when a rare keyword is used for a particular service. These parser-based methods have a high detection rate for known protocols. However, these methods require manually specified by experts and cannot deal with unknown protocols. The second class applies NLP techniques, e.g., N-gram analysis [22] to network traffic payloads. PAYL [23] uses 1-grams and unsupervised learning to build a byte frequency distribution model of payloads. McPAD [24] creates 2ν-grams and applies a sliding window to cover all sets of 2 bytes, ν positions apart in each network traffic payload. They require no expert domain knowledge and can detect zero-day worms, because payloads with exploit codes generally have an unusual byte frequency distribution. The drawbacks of them are unsatisfactory detection rate and relatively high computational overhead compared with parser-based methods.

2.2. Deep Learning for Intrusion Detection. Many deep learning techniques have been used for developing ANIDS. Ma et al. [25] evaluated deep neural network on the KDDCUP99 dataset, and Niyaz et al. [26] applied deep belief networks to intrusion detection on the NSL-KDD dataset. However, they only tested deep learning techniques on manually designed features, while their powerful ability to learn features from raw data has not been exploited. Recently, several attempts to learn effective features from raw packets have emerged. Yu et al. [27, 28] and Mahmood et al. [29] used autoencoder to detect anomaly traffic. Wang et al. [30] applied CNN to learn the spatial features of network traffic and used the image classification method to classify malware traffic, despite the fact that network payloads are more similar to documents. Torres et al. [31] transformed network traffic features into character sequence and used RNN to learn the temporal features, while Wang et al. [32] combined CNN and LSTM together to learn both spatial and temporal features. These methods are of great insights yet have evident weaknesses. Firstly, some time-based traffic features such as flow duration, packet frequency, and average packet length cannot be learned automatically by both CNN and LSTM. Besides, they ignore the semantic relation between each byte, which is a critical factor in NLP. In this paper, we remedy both problems by taking advantage of both expert domain knowledge and deep neural networks. The statistical features are manually designed and the payload features are extracted by deep learning techniques in NLP. To the best of our knowledge, no studies have made use of the advantages of both.

3. TR-IDS

TR-IDS aims at automatically extracting features from payloads of raw network packets to improve the accuracy of IDS. Since random forest has superior performance on structured data while convolutional network is suitable to handle unstructured data [33], we combine the advantages of both. It performs classification on bidirectional network flows (Biflow), which contains more temporal information than packet level datasets. The implementation schemes are illustrated in Figure 1, and the different stages of TR-IDS are described as follows:

(i) **Statistical features extraction**: we extract some critical statistical features from each network flow. These features include fields in packet headers and statistical attributes of the entire flow.

(ii) **Payload features extraction**: we map each byte in payloads into a word vector using word embedding and then extract salient features of payloads using text-convolutional neural network.

(iii) **Classification through random forest**: the statistical features and payload features are concatenated together, and then, the random forest algorithm is applied to classify the generated new dataset.

3.1. Statistical Features Extraction. In this section, we manually extract some discriminative features from the bidirectional network flows, where the first packet in each flow determines the forward (source to destination) and backward (destination to source) direction. We extract 44 statistical features from each flow, and most of them are calculated separately in both forward and backward direction. To be

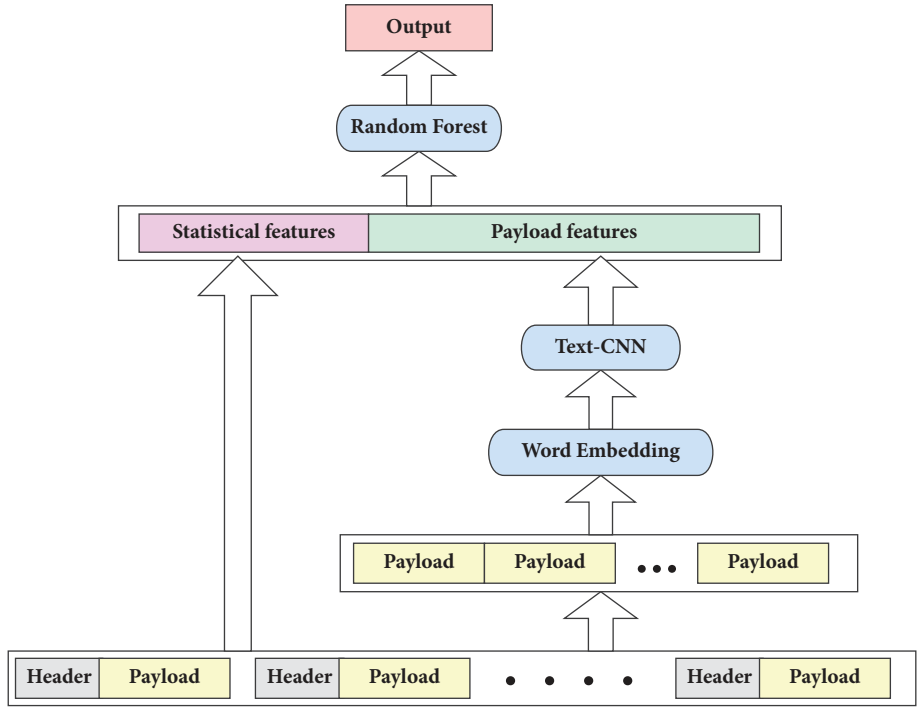

Figure 1: The general architecture of TR-IDS.

more specific, we first extract some basic features such as protocol, source port, and destination port, while ip addresses are not included because they vary in different networks and thus cannot generalize the characteristic of attacks. Then, some statistical attributes such as packet number, bytes number, and tcp flag number are calculated. After that, some time-based statistical measures are also extracted, such as the speed of transmission and time interval between two packets. These features are vital signatures for detecting attacks such as Probe, DoS, DDoS, Scan, U2R, and U2L, which have distinctive traffic patterns. We list all these features in Table 1

3.2. Payload Features Extraction. In this section, we introduce our deep-learning-based method of extracting features from network payloads. Word embedding technique is used to transfer one-hot representation of each byte to continuous vector representation. Then, text-convolutional network is utilized to extract the most salient features from each payload.

Byte-Level Word Embedding. The effective representation of each byte in payloads is a critical step. Yu et al. [27] took the decimal value of each byte as a feature. This method is not suitable as it introduces order relation to each byte. Wang et al. [32] adopted one-hot encoding to each byte and consider each sample as a picture; then a conventional CNN is applied to extract features. However, this method neglects the similarity in semantics and syntax of different bytes, and the worse is that it significantly increases the computation complexity. To remedy this problem, we utilize word embedding to map each byte into a low dimensional vector, preserving the semantic information and consuming much less computational cost. By now, the most well-known method of word embedding is word2vec [34], which is convenient to implement and has superior performance. Two popular kinds of implementation of word2vec are CBoW and Skip-Gram [35]. Since Skip-Gram generally has a better performance [35], in this paper, we apply Skip-Gram to our byte-embedding task.

The task of Skip-Gram is, given one word, predicting the surrounding words. The trained model does not perform any new task; instead, we just need the projection matrix, which contains the vector representation of each word. We define two parameter matrices, $W \in \mathbb{R}^{d\times|V|}$ and $W' \in \mathbb{R}^{|V|\times d}$, where d is the embedding dimension which can be set as an arbitrary size. Note that V is the vocabulary set and $|V|$ is the size of V. Each word in V is represented as a $|V| \times 1$ one-hot vector. The architecture of Skip-Gram is illustrated in Figure 2, and Skip-Gram works in the following 4 steps.

Step 1. Generate the one-hot input vector $x_i \in \mathbb{R}^{|V|}$ of the center word.

Step 2. Get the embedded vector of the center word $v_i = Wx_i \in \mathbb{R}^d$.

Step 3. For each surrounding word, generate a score vector $z = W'v_i$ and then turn it into probabilities, $\hat{y} = softmax(z)$. Thus, we obtain $2m$ softmax outputs, $\hat{y}_{i-m}, \ldots, \hat{y}_{i-1}, \hat{y}_{i+1}, \ldots, \hat{y}_{i+m}$, where m denotes the window size.

Step 4. Match the generated probability vectors with the true probabilities, which are the one-hot vectors of the actual output, $y_{c-m}, \ldots, y_{c-1}, y_{c+1}, \ldots, y_{c+m}$. The divergence

Table 1: Statistical features of the network flow.

Feature	Description
protocol	Protocol of the flow
src_port	Source port
dst_port	Destination port
f(b)_urg_num	Number URG flags in the forward(backward) direction (0 for UDP)
f(b)_ack_num	Number ACK flags in the forward(backward) direction (0 for UDP)
f(b)_psh_num	Number PSH flags in the forward(backward) direction (0 for UDP)
f(b)_rst_num	Number RST flags in the forward(backward) direction (0 for UDP)
f(b)_syn_num	Number SYN flags in the forward(backward) direction (0 for UDP)
f(b)_fin_num	Number FIN flags in the forward(backward) direction (0 for UDP)
pkts_num	Total packets in the flow
bytes_num	Total bytes in the flow
f(b)_pkts_num	Total packets in the forward(backward) direction
f(b)_bytes_num	Total bytes in the forward(backward) direction
f(b)_len_min	Minimum length of packet in the forward(backward) direction
f(b)_len_max	Maximum length of packet in the forward(backward) direction
f(b)_len_mean	Mean length of packet in the forward(backward) direction
f(b)_len_std	Standard deviation length of packet in the forward(backward) direction
duration	Duration of the flow
pkts_psec	Number of packets per second
bytes_psec	Number of packets per second
f(b)_pkts_psec	Number of forward(backward) packets per second
f(b)_bytes_psec	Number of forward(backward) bytes per second
f(b)_intv_min	Minimum time interval between two packets sent in the forward(backward) direction
f(b)_intv_max	Maximum time interval between two packets sent in the forward(backward) direction
f(b)_intv_mean	Mean time interval between two packets sent in the forward(backward) direction
f(b)_intv_std	Standard deviation time interval between two packets sent in the forward(backward) direction

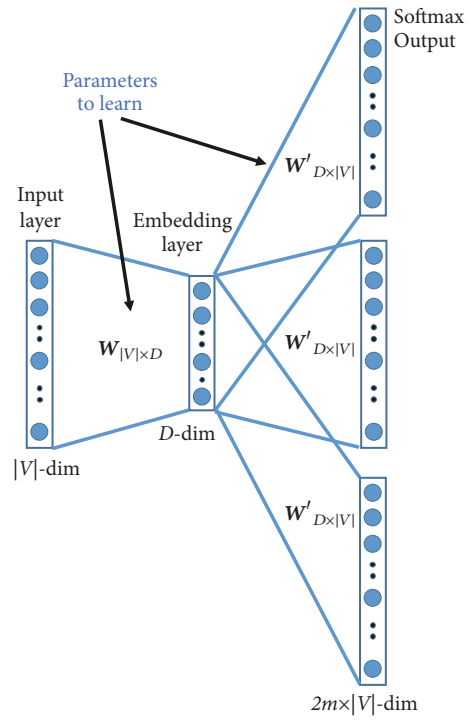

Figure 2: This figure illustrates how Skip-Gram works.

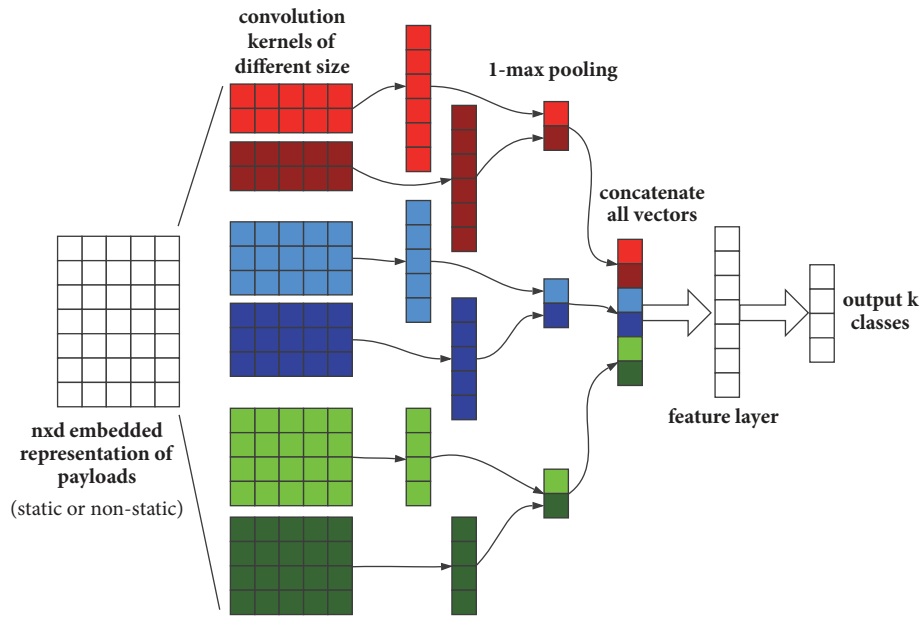

FIGURE 3: This figure illustrates how Text-CNN works.

between generated probabilities and true probabilities is the loss function for optimizing the parameters.

When it comes to the byte-level word embedding in our algorithms, each byte is considered as a word and represented as a one-hot vector. We first extract the payloads of all packets in each flow and concatenate them together as a flow payload. Each flow payload can be analogized as a sentence and they are composed of a text corpus, i.e., a training dataset. The embedding size can be set as a relatively small value (e.g., 10). After the training of Skip-Gram, we obtain the embedded representation of each byte.

Extract Payload Features through Text-CNN. We apply Text-CNN to extract features from the embedded payloads. Text-CNN is a slight variant of the CNN architecture and achieves excellent results on many benchmarks of sentence classification (or document classification) [19]. Text-CNN adopts the one-dimensional convolution operation to extract features from the embedded sentences. In Text-CNN, filters have a fixed width of embedding size, but have varying heights in the one layer, while in conventional CNNs, the sizes of filters in one layer are usually the same. The architecture of Text-CNN is illustrated in Figure 3.

Let $\mathbf{x}_i \in \mathbb{R}^d$ be a d-dimensional word vector corresponding to the embedded representation of ith word in a sentence (in our task, each byte corresponds to a word; thus, each payload is considered as a sentence). A sentence of length n (padded if the length is smaller than n) is denoted as

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_n \tag{1}$$

Note that $\oplus$ is the concatenation operator. When executing a convolution operation, a convolution filter $\mathbf{w} \in \mathbb{R}^{h \times d}$ is applied to a window of h words in the sentence to generate a new feature. To be specific, a feature c_i is calculated as follows:

$$c_i = f\left(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b\right) \tag{2}$$

where $\mathbf{x}_{i:i+h-1}$ is a window of words, b is a bias, and f is a nonlinear function. This filter is applied to each possible window $[\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}]$ to generate a new feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h-1}]$ and $\mathbf{c} \in \mathbb{R}^{n-h+1}$. Then, a max-pooling operation is applied to the feature map to obtain the maximum value $c_{\max} = \max(\mathbf{c})$, which is the most important feature of each feature map.

The process of extracting one feature by one filter is described above, and we have multiple filters with varying window size to extract multiple features. Note that, in the original Text-CNN, the features are concatenated and directly passed to a fully-connected *softmax* layer to output the probabilities of different classes. But in our implementation, we insert a feature layer between the concatenated layer and output layer. After the supervised training of the model, we extract features of each payload from this layer.

Classification through Random Forest. The Random Forest (RF) [20] is an ensemble algorithm consisting of a collection of tree-structured classifiers. Each tree is constructed by a different bootstrap sample from the original data using a decision tree algorithm, and each node of trees only selects a small subset of features for the split. The learning samples not selected with bootstrap are used for evaluation of the tree, called out-of-bag (OOB) evaluation, which is an unbiased estimator of generalization error. After the construction of the forest, once a new sample needs to be classified, it is fed into each tree in the forest and each tree casts a unit vote to certain class which indicates the decision of the tree. The

forest chooses the class with the most votes for the input sample.

RF has the following advantages:

(i) It has excellent performance in accuracy on structured data.

(ii) It is robust against noise and does not over-fit in most cases.

(iii) It is computational efficient and can run on large-scale datasets with high dimensions.

(iv) It can handle unbalanced datasets.

(v) It can output the importance weight of each feature.

These merits of RF encourage us to choose it as our final classification. In this step, we concatenate the statistical features and payload features to generate the final representation of the network flows. Then, this new dataset is fed into the RF algorithm for training and validation.

4. Performance Evaluation

4.1. Datasets and Preprocessing. We evaluate the performance of our method on ISCX2012 dataset [36]. It is an intrusion detection dataset generated by the Information Security Center of Excellence (ISCX) of the University of New Brunswick (UNB) in Canada in 2012. This dataset consists of 7 days of network activity, including normal traffic and four types of attack traffic, i.e., Infiltrating, HttpDoS, DDoS, and BruteForce SSH. Although KDDCUP99 dataset [37] is widely used to evaluate IDS techniques, it is really old-fashioned and cannot actually reflect the behavior of modern attacks. In contrast, ISCX2012 is much more updated and closer to reality. This dataset consists of seven raw pcap files and a list of label files. The label files record the basic information of each network flow, e.g., label, ip address, port, start time, and stop time. We have to split the network flows in the pcap files and label them using records in the label files. Note that the labeled files contain a few problems. For example, the packet numbers recorded in them are not identical to the actual packet number in pcap files. Besides, the time records in them do not exactly correspond to the timestamps in the pcap files. Therefore, we have to remove all incorrect and confused records. We chose most attack samples and randomly chose a small subset of legitimate ones to generate a relatively balanced dataset. Then, we divided the preprocessed dataset into training and testing set using a ratio of 70% and 30%, respectively. Our preprocessing results are shown in Table 2.

4.2. Evaluation Metrics. Three metrics are used to evaluate the performance of TR-IDS: Accuracy (ACC), Detection Rate (DR), and False Alarm Rate (FAR), which are frequently used in the evaluation of intrusion detection. ACC is a good metric to evaluate the overall performance of a system. DR is used to evaluate the attack detection rate. FAR is used to evaluate misclassification of normal traffic. The three metrics are formulated as follows:

$$\begin{aligned} Accuracy\,(ACC) &= \frac{TP + TN}{TP + FP + FN + TN} \\ DetectionRate\,(DR) &= \frac{TP}{TP + FN} \\ FalseAlarmRate\,(FAR) &= \frac{FP}{FP + TN}, \end{aligned} \tag{3}$$

where TP is the number of instances correctly classified as A, TN is the number of instances correctly classified as Not-A, FP is the number of instances incorrectly classified as A, and FN is the number of instances incorrectly classified as Not-A.

4.3. Experimental Setup. Scapy, Pytorch, and Scikit-learn are the software frameworks for our implementation. The operating system is CentOS 7.2 64bit OS. Server is PR4712GW/X10DRFF-iG with 2 Xeon e5 CPUs with 10 cores and 64GB memory. Four Nvidia Tesla K80 GPUs are used to accelerate the training of CNN. In our all experiments, the Text-CNN contains convolution filters with three different size, i.e., 3, 4, and 5, and there are 100 channels for each. The stride is 1 and no padding is used. The mini-batch size is 100 and optimizer is Adam with default parameters. The parameters of RF are set by default, except the number of trees, which is set as 200.

4.4. Experimental Results. In this section, we show the experimental results of our methods. We set the number of extracted payload features as 50 and the truncated length of bytes in each payload as 1000. Table 3 shows the result of 5-class classification on ISCX2012 and Table 4 shows the confusion matrix of the classification. It is obvious that our method can nearly identify all attacks of Infiltration, BFSSH, and HttpDoS but confuses a few DDoS attacks with the normal traffic. The reason is that some network flows of DDoS are really similar to normal traffic; thus, it is unrealistic to identify each flow in a DDoS attack.

Since ISCX2012 dataset was published much later than DARPA1998, there are much fewer available corresponding experimental results. Although some existing methods are evaluated on it, they have different preprocessing procedures and even use different proportions of the dataset. Thus, it is unfair to compare our methods with these methods. In this case, in order to demonstrate the effectiveness of our method, we implemented five other methods. The first four ones are support vector machine (SVM), fully-connected network (NN), convolutional neural network (CNN), and random forest (RF-1), and their inputs are statistical features combined with 1000 raw bytes. The fifth one is running random forest on just statistical features (RF-2). Table 5 compares TR-IDS with the five methods. Note that the performance of RF-1 is inferior to that of RF-2, which means the features of raw bytes may even deteriorate the performance of intrusion detection. The superior performance of TF-IDS demonstrates the effectiveness of the proposed feature extraction techniques.

Table 2: Preprocessing results of the ISCX2012 dataset.

Category	Count	Percentage	Training count	Testing count
Normal	10000	28.28%	6971	3029
Infiltration	9925	28.07%	6930	2995
BFSSH	7042	19.92%	4911	2131
DDoS	4963	14.04%	3513	1450
HttpDoS	3427	9.69%	2424	1003
Total	35357	100%	24749	10608

Table 3: Performance of TR-IDS on ISCX2012 (%).

Type	ACC	DR	FAR
Infiltration	99.87	99.77	0.06
BFSSH	99.99	99.95	0.00
DDoS	98.09	95.93	0.40
HttpDoS	99.90	99.70	0.07
Total	99.13	99.26	1.18

Table 4: Confusion matrix of the 5-class classification task.

	Normal	Infiltration	BFSSH	DDoS	HttpDoS
Normal	**2993**	1	0	33	2
Infiltration	0	**2988**	0	4	3
BFSSH	0	1	**2130**	0	0
DDoS	56	1	0	**1391**	2
HttpDoS	0	2	0	1	**1000**

Table 5: Comparison with other algorithms (%).

Type	ACC	DR	FAR
SVM	86.16	81.48	1.95
NN	90.99	91.17	9.45
CNN	95.75	96.61	6.41
RF-1	97.21	96.81	1.74
RF-2	98.59	98.24	2.67
TR-IDS	**99.13**	**99.26**	**1.18**

4.5. Sensitivity Analysis. In this section, we show the results of sensitivity tests on the two important hyperparameters, i.e., the truncated length of payloads for feature extraction and the number of features extracted from payloads. We first fixed the truncated length of payloads as 1000 and then varied the extracted feature number from 5 to 100. After that, we fixed the extracted feature number and varied the truncated length from 500 to 3000. As we can see in Table 6, our methods are not sensitive to the two hype-parameters. The best value of feature number locates at the middle of 5 and 100, i.e., 50. The reason is that too few features cannot contain enough information of the entire payload, and too many features bring noise to the final classification algorithm. For the truncated length of payloads, we find that large length contributes to a better performance; it is because a small length may result in the loss of information of payloads. Nevertheless, a large length also leads to a high computational cost.

Table 6: Influence of the payload length and feature number (%).

Payload length	feature number	ACC	DR	FAR
1000	5	98.68	98.92	1.91
1000	10	98.71	98.94	1.88
1000	20	98.84	99.20	2.01
1000	50	**99.13**	**99.26**	**1.18**
1000	100	99.09	99.24	1.48
500	50	99.02	99.35	1.81
1000	50	99.13	99.26	1.18
1500	50	99.14	98.25	1.17
2000	50	99.18	98.36	1.25
3000	50	**99.21**	**99.40**	**1.12**

5. Conclusion

In this paper, we propose a novel intrusion detection framework, i.e., TR-IDS, which utilizes both manually designed features and payload features to improve the performance. It adopts two modern NLP techniques, i.e., word embedding and Text-CNN, to extract salient features from payloads. The word embedding technique retains the semantic relations between each byte and reduces the feature dimension, and then Text-CNN is used to extract features from each payload. We also apply the sophisticated random forest algorithm for the final classification. Finally, extensive experiments show the superior performance of our method.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was previously presented at the 4th International Conference on Cloud Computing and Security (ICCCS 2018). This work is supported by the National Natural Science Foundation of China (Grants nos. 61105050, 61702539, and 60970034).

References

[1] J. Cui, Y. Zhang, Z. Cai, A. Liu, and Y. Li, "Securing display path for security-sensitive applications on mobile devices," *Computer, Materials & Continua*, vol. 55, no. 1, pp. 17–35, 2018.

[2] A. Pradeep, S. Mridula, and P. Mohanan, "High security identity tags using spiral resonators," *Computers, Materials and Continua*, vol. 52, no. 3, pp. 185–195, 2016.

[3] M. Roesch et al., "Lightweight intrusion detection for networks," *Lisa*, vol. 99, pp. 229–238, 1999.

[4] J. Cannady, "Artificial neural networks for misuse detection," in *National Information Systems Security Conference*, vol. 26, 1998.

[5] H. Brahmi, I. Brahmi, and S. Ben Yahia, "OMC-IDS: At the cross-roads of OLAP mining and intrusion detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7301, no. 2, pp. 13–24, 2012.

[6] F. Jemili, M. Zaghdoud, and M. B. Ahmed, "A framework for an adaptive intrusion detection system using Bayesian network," in *Proceedings of the ISI 2007: 2007 IEEE Intelligence and Security Informatics*, pp. 66–70, May 2007.

[7] M. Blowers and J. Williams, "Machine Learning Applied to Cyber Operations," in *Network Science and Cybersecurity*, vol. 55 of *Advances in Information Security*, pp. 155–175, Springer New York, New York, NY, 2014.

[8] C. Kruegel and T. Toth, "Using Decision Trees to Improve Signature-Based Intrusion Detection," in *Recent Advances in Intrusion Detection*, vol. 2820 of *Lecture Notes in Computer Science*, pp. 173–191, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[9] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 5, pp. 649–659, 2008.

[10] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Systems with Applications*, vol. 39, no. 1, pp. 424–430, 2012.

[11] E. Min, Y. Zhao, J. Long, C. Wu, K. Li, and J. Yin, "SVRG with adaptive epoch size," in *Proceedings of the 2017 International Joint Conference on Neural Networks, IJCNN 2017*, pp. 2935–2942, USA, May 2017.

[12] E. Min, J. Cui, and J. Long, "Variance Reduced Stochastic Optimization for PCA and PLS," in *Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, pp. 383–388, Hangzhou, December 2017.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–11, 2018.

[15] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive Fractional-Pixel Motion Estimation Skipped Algorithm for Efficient HEVC Motion Estimation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–19, 2018.

[16] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology*, vol. 37, pp. 51–89, 2003.

[17] B. Chigier, "Automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, p. 19, 1997.

[18] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pp. 1555–1565, USA, June 2014.

[19] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014.

[20] A. Liaw and M. Wiener, "Classification and regression by randomforest," *The R Journal*, vol. 2, no. 3, pp. 18–22, 2002.

[21] V. Matthew Mahoney and K. Philip Chan, "Learning models of network traffic for detecting novel attacks," Tech. Rep., 2002.

[22] F. Peter Brown, V. Peter Desouza, L. Robert Mercer, J. Vincent Della Pietra, and C. Jenifer Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[23] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*, vol. 3224 of *Lecture Notes in Computer Science*, pp. 203–222, Springer, Berlin, Germany, 2004.

[24] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: a multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.

[25] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, 2016.

[26] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, New York, NY, USA, December 2015.

[27] Y. Yu, J. Long, and Z. Cai, "Session-Based Network Intrusion Detection Using a Deep Learning Architecture," in *Modeling Decisions for Artificial Intelligence*, vol. 10571 of *Lecture Notes in Computer Science*, pp. 144–155, Springer International Publishing, Cham, Germany, 2017.

[28] Yang Yu, Jun Long, and Zhiping Cai, "Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders," *Security and Communication Networks*, vol. 2017, pp. 1–10, 2017.

[29] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proceedings of the 2017 International Joint Conference on Neural Networks, IJCNN 2017*, pp. 3854–3861, USA, May 2017.

[30] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection on denial-of-service attacks based on computer vision techniques," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 64, no. 9, pp. 2519–2533, 2015.

[31] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of Recurrent Neural Networks for Botnet detection behavior," in *Proceedings of the 2016 IEEE Biennial Congress of Argentina, ARGENCON 2016*, Argentina, June 2016.

[32] W. Wang, Y. Sheng, J. Wang et al., "HAST-IDS: Learning Hierarchical Spatial-Temporal Features using Deep Neural Networks to Improve Intrusion Detection," *IEEE Access*, 2017.

[33] https://www.import.io/post/how-to-win-a-kaggle-competition/.

[34] Y. Goldberg and O. Levy, "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method, 2014," https://arxiv.org/abs/1402.3722.

[35] M. Tomas, K. Chen, G. Corrado, and D. Jeffrey, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[36] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defence Applications*, pp. 1–6, IEEE, July 2009.