Graphical Abstract

A novel flow-vector generation approach for malicious traffic detection

Jian Hou, Fangai Liu, Hui Lu, Zhiyuan Tan, Xuqiang Zhuang, Zhihong Tian

Highlights

A novel flow-vector generation approach for malicious traffic detection

Jian Hou, Fangai Liu, Hui Lu, Zhiyuan Tan, Xuqiang Zhuang, Zhihong Tian

- We proposed an approach to gradually construct flow vectors from the field vector
- Extract information irrelevant to the payload from the raw traffic as input
- Unique field value representation makes the embedded vector more effective
- The adjustable number of packets in-flow makes the model more flexible

A novel flow-vector generation approach for malicious traffic detection

Jian Hou[a,b], Fangai Liu[a], Hui Lu[b,*], Zhiyuan Tan[c], Xuqiang Zhuang[a], Zhihong Tian[b,*]

[a]*Informatization Office, Shandong Normal University, Jinan, 250014, China*
[b]*Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510300, China*
[c]*School of Computing, Merchiston Campus, Edinburgh Napier University, Edinburgh, EH10 5DT, UK*

Abstract

Malicious traffic detection is one of the most important parts of cyber security. The approaches of using the flow as the detection object are recognized as effective. Benefitting from the development of deep learning techniques, raw traffic can be directly used as a feature to detect malicious traffic. Most existing work usually converts raw traffic into images or long sequences to express a flow and then uses deep learning technology to extract features and classify them, but the generated features contain much redundant or even useless information, especially for encrypted traffic. The packet header field contains most of the packet characteristics except the payload content, and it is also an important element of the flow. In this paper, we only use the fields of the packet header in the raw traffic to construct the characteristic representation of the traffic and propose a novel flow-vector generation approach for malicious traffic detection. The preprocessed header fields are embedded as field vectors, and then a two-layer attention network is used to progressively generate the packet vectors and the flow vector containing context information. The flow vector is regarded as the abstraction of the raw traffic and is used to classify. The experiment results illustrate that the accuracy rate can reach up to 99.48% in the binary classification task and the average of AUC-ROC can reach 0.9988 in the multi-classification task.

[*]Corresponding author. E-mail address: luhui@gzhu.edu.cn(Hui Lu), tianzhihong@gzhu.edu.cn(Zhihong Tian)

Keywords: Deep Learning, Malicious Traffic, Embedding, Attention Mechanism
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

Malicious traffic usually refers to the network traffic that invades, interferes with, or grabs data without permission, which is the most common and major network security threat. Recently, McAfee's research report [1] showed that malware increased by 47% in a quarter, especially spam showed the highest increase by 250%, malware is the most often used attack vector. Compared with intrusions at the network layer, malicious traffic at the application layer generated by modern malware is faster in propagation and riskier. Besides, with the growth of encrypted network traffic, the activity of malicious traffic is more concealed.

Due to the advantage of payload content independence and detecting unknown malicious traffic, anomaly-based detection techniques have attracted more interest. As a typical representative technology based on anomaly detection methods, machine learning is also deeply applied in malicious traffic detection. Buczak and Guven [2] reviewed commonly used traditional machine learning algorithms in intrusion detection systems (IDS) and described some well-known data sets used in study. A further detailed investigation and analysis of various machine learning techniques was carried out by Mishra et al. [3], and the author also analyzed and compared the detection capability of these techniques for detecting the various categories of attacks. From the above surveys, we can see that traditional machine learning has achieved fruitful results in various intrusion detection tasks, and achieved satisfactory performance on the data sets of manually designed features.

As an advancement branch of machine learning, deep learning techniques are also used to detect malicious traffic by analyzing these manually designed features. Kim et al. [4] and Yin et al. [5] constructed detection models using LSTM and RNN respectively, and realized malicious traffic detection by learning the potential sequence relationship between features in the data set. Javaid et al. [6] and Shone et al. [7] designed different autoencoders (AE) based on deep learning in their detection models respectively, and generated more salient features of traffic through building the high-level abstraction of the manually designed features. Tian et al. [8] proposed a distributed system based on deep learning algorithm, which uses an amended residual network to detect malicious URL requests. Due to the combination of complex architecture and amount of nonlinear transformations, deep learning can automatically learn features representations from the raw traffic without being limited to manually and expert-originated features. The first malicious traffic detection method that automatically extracts features from raw traffic was proposed by [9], and some similar methods have been proposed [10, 11]. Similarly, for the irrelevance characteristics with payload content, the deep-learning-based method that uses the raw traffic as a model input has been widely applied in encrypted malicious traffic classification [12, 13, 14, 15]. A survey of deep learning approaches for malicious traffic detection was presented in [16], the author reviewed the taxonomy of detection technology, analyzed the challenges and prospects of detection technology based on deep learning.

Deep learning technology has received much more attention in malicious traffic detection or traffic classification. Reviewing the existing works, we found that more hybrid models are being proposed and more complex input data is to deal with, especially the models that use raw traffic as input. Most models classify traffic from the flow level, to accurately express a flow, the input data often needs to be preprocessed into a high-dimensional vector and be complicatedly transformed. Actually, it is possible that we only need a few data packets (such as the first N packets) in the flow [17] and some salient information (such as packet headers) in the packets [18] to be able to identify or classify the flow. In this work, we proposed a novel approach of constructing flow vectors, and then detected traffic use it. We take the field's value of the packet's header as input data and embed it as a vector. The flow vector is generated gradually by field vectors through the hierarchical attention network we built. We consider that the flow vectors which have been trained by supervised learning have included the context information inside and between packets and can be used to classify the traffic. In summary, our aim is that from limited header field values, construct a comprehensive flow-vector expression for classification. To our knowledge, the method proposed in this work is the first attempt to progressively construct the flow-level vectors from the packet fields information.

The proposed detection method, in this paper, has focused on malicious traffic detection at the flow level. The main contributions of this paper can be summarized as follows:

1. We find a method to use vectors to represent network flow. Briefly, we proposed an approach to gradually construct flow vectors from the field vector of the packet header. The header field is embedded in a field vector. All field vectors complete the extraction of field-level and packet-level features respectively through a two-layer attention network, thereupon then generate a flow vector that can be used by classification.
2. Among the existing methods with the raw traffic as the input, the proposed approach in this paper requires the least data in each packet. The input contains most of the information that can be extracted from the packet, and for the irrelevance of the payload content, we consider this method is also suitable for detecting encrypted malicious traffic.
3. We propose a field related word embedding method, which makes the embedded vector have the attributes of the field and more effectively express the meaning of the field value.
4. In the packet-level encoding and attention layer, the number of input packets is designed to be adjustable. To detect malicious traffic as soon as possible, we expect to use a few packets in the early stage of data flow to effectively detect malicious traffic. Our experiments tested the effect of the different number of packets in-flow on the accuracy.

The remainder of this paper is organized as follows. Section 2 discusses related work and analyze the enlightenment to the new method proposed in this paper. Section 3 describe the basic algorithms used in the proposed model. Section 4 details the proposed model and the process of flow vector generation. In section 5, the effectiveness of the proposed approach to malicious traffic detection is verified, and compared with the baseline algorithm. The conclusions and future work of this paper are summarized in Section 6.

2. Related Work

Many researchers have proposed a rich number of malicious traffic detection methods. In the domain, deep learning has gained several notable achievements with various network models. In [19], the author successfully applied Long Short-Term Memory (LSTM) network to intrusion detection for the first time, and applied the classifier to KDD'99 datasets. The result shows that the LSTM classifier provides superior performance. Li et al. [20] proposed an intrusion detection method using convolutional neural networks

(CNN). They converted the standard KDD'99 or NSL-KDD data form into image form as the input of the CNN classifier, and realized the automatic selection of features. Vinayakumar et al. [21] utilized a simple CNN model with multiple layers and a hybrid CNN model with LSTM Units, and Gated Recurrent Units (GRU) for network anomaly detection with the KDD'99 dataset. Experimental results show that the model has high accuracy in network intrusion detection. Aleesa et al. [22] reviewed and analyzed the research status of an intrusion detection system based on deep learning technology among four major databases. They divided deep learning technology into single and hybrid techniques, and reviewed the research results of each category in detail. The research indicated that the strongest aspect of deep learning techniques is learning feature hierarchies based on the patterns in the data, and hybrid techniques can also be used for improving detection efficiency and accuracy in several cases. Most of the above researches are based on data set composed of hand-designed features.

Deep learning as a typical approach of representation learning has achieved very good performance in many domains, and is usually used to automatically learn features from raw data. Wang et al. [9] firstly apply the representation learning approach to the malware traffic classification domain, and proposed a method using CNN to detect the malicious traffic in raw traffic data. Based on this research, Huang et al. [23] proposed a novel ant-colony-based clustering algorithm to improve the accuracy and robustness of the model. However, such a model lacks the ability to detect unknown attacks, besides, since it uses flow-level information of raw traffic, the model only learns the spatial features of the raw traffic, but not the temporal features.

The sequence of packets in a network flow contains rich temporal information, and some researchers extracted the temporal features of raw traffic at the network packet level. Wang et al. [17] adopt CNN and LSTM to design a hierarchical intrusion detection system to learn traffic features in spatial and temporal. The experimental results demonstrate its effectiveness in both feature learning and false alarm rate reduction. Xie et al. [24] also extracted features from raw traffic in packet-level and flow-level, and then build a hierarchical structure neural network model with CNNs and LSTMs to detect malware traffic. The model divided a flow into request and response, which also took both the statistical characteristics and the raw traffic data into account respectively. These models can learn the characteristics of traffic data well and can perform incremental learning. However, the volume of sample data in these models used for training is generally very large.

Table 1: Summary of relevant research works.

Ref.	**Method**	**Type of Features**[1]	**Features**[2]	**Pre-processing**[3]	**Datasets**	**Evaluation Metrics**
[19]	LSTM	Manu	SF	NE	KDD Cup'99	ACC = 93.82%
[21]	CNN	Manu	SF	NE	NSL-KDD	F = 90.01%
[22]	CNN & LSTM, CNN & GRU	Manu	SF	NE	KDD Cup'99	ACC = 93.82%
[9]	CNN	Raw	SLB	NE	USTC-TFC2016	ACC = 99.41%
[25]	SDAE	Raw	FB & PB	IR	ISCX2012, CTU-13	ACC = 99.48%
[17]	CNN & LSTM	Raw	PLB & FLB	IE	DARPA98, ISCX2012	ACC = 99.89%
[26]	Random Forest & CNN	Manu & Raw	SF & FLB	BE	ISCX2012	ACC = 99.13%
[18]	LSTM	Raw	FB	BE	ISCX2012, USTC-TFC2016	ACC = 99.97%
[27]	CNN & LSTM, CNN	Raw	PLB & FLB	NE	CTU-13	ACC = 99.9% (Rbot)
[23]	Heuristic CNN	Raw	SLB	IE	CTU-13	F = 94.4%
[24]	CNN & LSTM	Manu & Raw	SF & PB	NE	ISCX2012, BTHT-2018	F = 93.51%

[1] Manu: Manual design feature; Raw: Raw traffic feature.
[2] SF: statistical features; SLB: Session level Byte; FLB: Flow level Byte; PLB: Packet level Byte; FB: Field Byte; PB: Payload Byte.
[3] NE: Normalization encoding; IE: Image encoding; BE: Byte-embedding.

Packets are generally composed of packet header and payload, which is processed separately as two-part in some intrusion detection system. In [26], the two parts of information are extracted respectively and concatenated together after encoding as the input of classifier. The model has a superior performance in detection rate and false alarm rate. However, the process of extracting payload features significantly increases the computation costs. In the malicious traffic classification model designed by Diallo et al. [28], the author used the packet header and its statistical information as the main features, and payload is configured as an option for generating flow vectors. Experimental results show that accuracy degrades only marginally when payload features are not employed for classification. According to the results it is considered that the information contained in the packet header is sufficient to obtain a relatively accurate classification. The use of statistical information can improve the accuracy of the model, but it is often necessary to wait for obtaining the information of the entire flow, which is not conducive to detect malicious traffic early. In [18], the author proposed an approach for classifying malicious traffic in packet-level. Instead of the whole packet, they considered a field of the packet header as a word, and each packet as a paragraph. They used word embedding to construct the word vectors of packet header fields and leverage LSTM to learn temporal features of the packets. This approach achieved a high score of accuracy in bi-classification tasks, and the more advantage is the higher time efficiency. However, the approach is little focuses on the information of flow-level, and has no experiment about multi-classification of traffic. Summary of several the relevant research works are shown in Table 1.

In general, based on existing research, the following points can be summarized: 1) Deep learning as a cutting-edge subset of machine learning techniques has been applied to the research of malicious traffic detection commonly. Due to the excellent ability to automatically extract features from raw traffic, the input of the model based on deep learning is not limited by the features of manual design. 2) The useful information can be extracted in raw traffic at packet-level and flow-level, and the fusion of these information can improve the ability of the model to detect malicious traffic. 3) The features extracted from the packet header can directly reflect the behaviour of malicious traffic without depending on payload content, using it as input to the model can improve generalization performance.

3. Preliminary

In this section, we will provide preliminary information necessary to understand our motivations and the concepts behind the model proposed in this paper.

3.1. Fields of the packet header

The basic packet consists of a header with the sending and receiving systems' information, and a body, or payload, with the data to be transferred. According to the TCP/IP protocol architecture model, the protocols at each layer are in charge of the packet header to which it belongs, thus the structure of the packet header is usually fixed in the same protocol. Commonly used data packet headers are the Ethernet header of the data link layer, IP header of the network layer, and TCP header or UDP header are in the transport layer. In fact, the Ethernet header only reflects the information of the network structure, so we only use the other headers in this work.

3.2. Word embedding

Word embedding has been wildly used and proven to be effective in many natural language processing tasks, compared to one-hot encoding, which provides more robust representations for words by mapping them to a low-dimensional and continuous vector space. Word2Vec as the classical approach of word embedding was proposed by Mikolov et al. [25]. The skip-gram is one of the models in the approach, which is designed to predict the surrounding words with a word in the central position, and then generate the word vectors that represent the abstract of semantics and syntax features.

Similar to this study, Pennington et al. [27] constructed a new model for word representation named GloVe. Compared to Word2Vec, the GloVe model incorporated the statistical information of the global corpus, to improve performance on word analogy, and its unsupervised learning model also has a good effect on dealing with similar words. The cost function of GloVe is:

$$J = \sum\nolimits_{n,m=1}^{W} f(N_{nm})(x_i^{\top} \tilde{x}_m - \log N_{nm})^2 \tag{1}$$

where $f(\cdot)$ is a weighting function satisfying certain conditions, N_{nm} represents the number of times word m occurs in the context of word n, x and $\tilde{x}$ represent the word vector and context word vectors.

3.3. Bidirectional Gated Recurrent Units (Bi-GRU)

The GRU [29] is an effective model for processing sequence data, compared to LSTM, which uses a gating mechanism to control the update process of information without using a separate cell state. We define h_t is the new stat of GRU at time t, the state is updated by:

$$\overrightarrow{h_t} = (1 - z_t) \odot \overrightarrow{h_{t-1}} + z_t \odot \tilde{h}_t. \tag{2}$$

h_{t-1} is the previous stat. The candidate state $\tilde{h}_t$ and update gate z_t is separately computed by:

$$z_t = \sigma(W_z x_t + U_z \overrightarrow{h_{t-1}}) + b_z) \tag{3}$$

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h \overrightarrow{h_{t-1}}) + b_h) \tag{4}$$

Where x_t is the input vector at time t, r_t is the reset gate which controls the effect of the previous state h_{t-1} at the current moment on the time t, and can be computed by:

$$r_t = \sigma(W_r x_t + U_r \overrightarrow{h_{t-1}}) + b_r) \tag{5}$$

Here σ is the *sigmoid* function. W_r and U_r are weight matrices which are learned.

In summary, a sequence $(\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_T})$ is computed by formula (4)-(7) with the input sequence $X = (x_1, x_2, \cdots, x_T)$, which we called the forward coding of sequence, or annotations that incorporating the contextual information of sequence X.

In the same way, the backward coding $(\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots, \overleftarrow{h_T})$ could be generated similarly, when the sequence X is input reversely. We concatenate the forward and backward states to obtain the annotations , and then the Bi-GRU model has been constructed, which $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. Figure 1 shows the computational process of Bi-GRU. In the rest of the paper, we take the following equation to describe the update of GRU hidden state briefly:

$$h_t = BiGRU(x_t), \quad t \in [1, T] \tag{6}$$

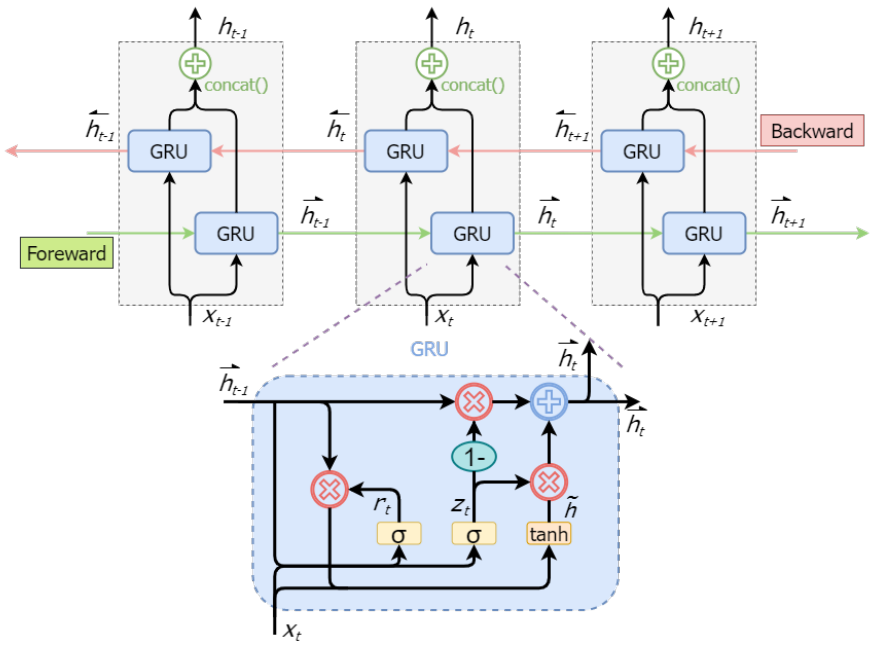


Figure 1: Structure of the Bi-GRU at time step t

3.4. Attention mechanism

The attention mechanism is inspired that human attention will always focus on the particular regions of an image, and is widely used in sequence modeling, especially in Natural Language Processing (NLP) [30]. The attention model is designed to select the more important features in the current task, and the model training can be regarded as an addressing process. The process can be summarized that, according to the query vector generated by the task, the importance weight is calculated and attached to the sequence value, and then the output with attention distribution is obtained. In terms of the calculation method, the attention mechanism is divided into soft attention, hard attention, and self-attention. Considering the differentiability and efficiency of the model, soft attention is a popular method for NLP research [31], and will be used in this work. Figure 2 is the schematic diagram of soft attention, and the attention vector s is calculated as follows:

$$s_i = \sum_{t=1}^{T} \alpha_{it} h_{it} \tag{7}$$

$$\alpha_{it} = \frac{exp(u_t^\top u_{iw})}{\sum_t exp(u_{it}^\top u_{iw})} \tag{8}$$

$$u_{it} = \tanh(W_{iw} h_{it} + b_w) \tag{9}$$

where u_{it} as a hidden representation of h_{it}, u_w is randomly initialized and jointly learned during the training process.

In this paper, we briefly express the calculation process of the ith attention module as:

$$s_i = ATT(h_{it}), \quad t \in [1, T] \tag{10}$$

4. Proposed approach

In this section, we introduce the proposed malicious traffic detection approach. The inspiration of our approach is that subject to the network protocol, the fields of packet header have a strict order in it, just like a potential grammar rule for the built sentence, and catch hold of this point, we consider each packet header as a sentence which constructed by fields as words.

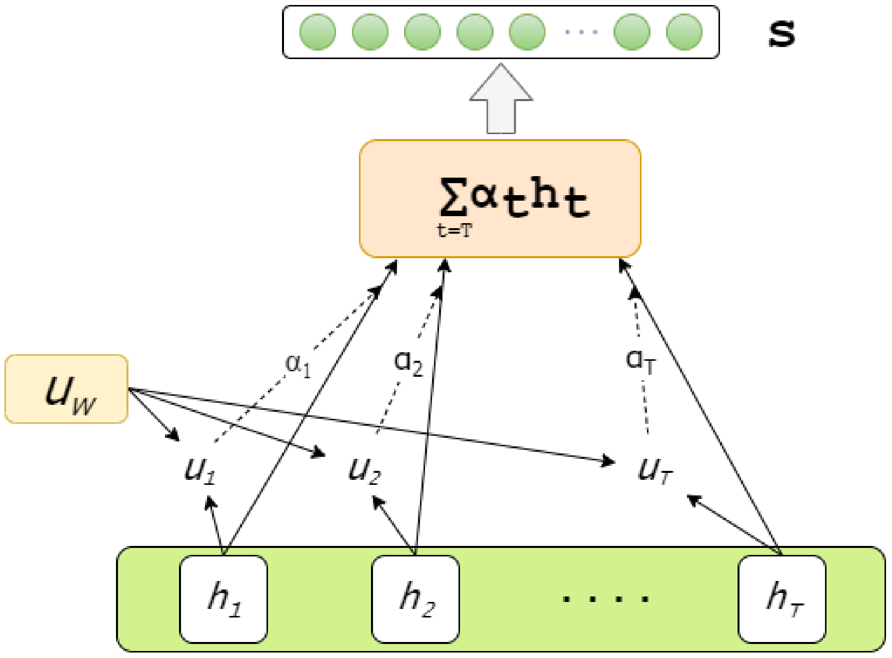


Figure 2: Structure of the soft attention

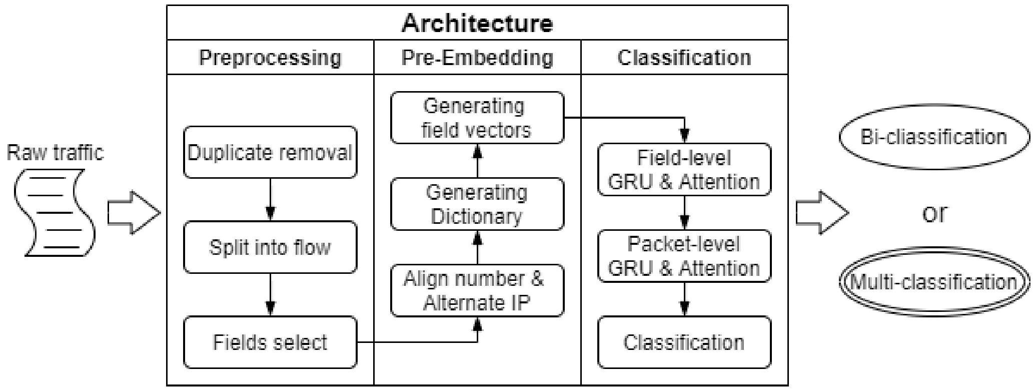


Figure 3: the overview structure of the approach

Figure 3 shows the overview structure of the approach from the perspective of workflow. Given a TCP/IP network raw traffic, our approach aims to sample and classify each flow as malicious or normal with maximizing accuracy and minimizing false alarm rate. Overall, the approach is divided into three parts: preprocessing, pre-embedding and classification. Features of raw traffic are extracted in the preprocessing module, that is, the field value of the data packet header, and organized in the unit of flow. Some selected features obtain their vectorized representation in the pre-embedding module, and as an input to the classification model. In the third module, we apply a hierarchical attention network to construct a flow vector from its packet field vectors, and then classify the sample flow using it. In what follows we detail the operation of each module, and we elaborate on the design of some key steps.

4.1. Preprocessing

In this work, we use the field values extracted from the packet header to express a network flow. The pre-processing module is responsible for extracting and labeling selected data from the raw traffic, without any conversion operation, and the labeled data will be used as the source data of the following experiments.

Affected by the network environment, in most cases, there will be some packets that do not offer substantial information to the flow in the datasets directly captured from the network [32], such as that of duplicates and retransmission. These packets usually do not contribute to the salient characteristics of the traffic [33], so we choose to remove that. After de-duplication, the raw traffic should be split into flows, which have the same IP-address pair, port-number pair, and IP protocol, and labeled as a sample. Considering that some fields contain little traffic behavior information, we only select part of the fields, and ignore the fields that do not have the meaning of actual network behavior, such as checksum fields, which is the mathematical expression of byte value in the packet header. Table 2 lists the header fields. The features of raw traffic can be represented as:

$$
\mathbf{v_{ij}} = \begin{bmatrix} [v_{11}, & v_{12}, & \cdots, & v_{1j}, & \cdots, & v_{1M}] \\ [v_{21}, & v_{22}, & \cdots, & v_{2j}, & \cdots, & v_{2M}] \\ \cdots & & & & & \\ [v_{i1}, & v_{i2}, & \cdots, & v_{ij}, & \cdots, & v_{iM}] \\ \cdots & & & & & \\ [v_{L1}, & v_{L2}, & \cdots, & v_{Lj}, & \cdots, & v_{LM}] \end{bmatrix} \quad \begin{matrix} i \in [1, L] \\ j \in [1, M] \end{matrix}
$$

L is the number of packets that are selected in the flow, and M is the number of fields in a packet.

Table 2: List of candidate header fields.

Protocol	Candidate Header Field
IP	version, header length, type of service, total length, identifier, flags, fragmented offset, time to live (TTL), protocol, source address, destination address;
TCP	source port, destination port, sequence number, acknowledge number, window size value, header lengh, flags, urgent pointer;
UDP	source port, destination port, lengh;

The data source used by our model is raw traffic, which can be captured directly from the network or be "pcap" or "pcapng" files that have already been generated as a date set. In this paper, considering the verification of model performance and comparison with existing work, we employ the public data sets (detailed in Section 5.1) as the source of the traffic to be detected, and the generation algorithm of labeled flow is shown in Algorithm 1.

4.2. Pre-Embedding

Word embedding vectors performed well for extracting syntactic and semantic relations [34] in many NLP tasks. Inspired by this study, we apply field embedding to generate the vector representation of the fields, which we called field vector. We treat a field's value as a word so that the header composed of fields can be seen as a sentence composed of words. We select the appropriate field from the candidate fields according to the dectection

Algorithm 1: Extracting Labeled flow from Data sets

```
1  Require: Data set CTU-13, Data set ISCX2012;
   Input   : $*.pcap$ files, $File_n, n \in [1, 14]$
   Output: Files containing labeled flow: $V_{Neris}$, $V_{Rbot}$, $V_{Virut}$, $V_{BFS}$,
           $V_{HDos}$, $V_{Inf}$, $V_{Normal}$
2  Extract 54-byte from each packet; %For reducing storage and
     computing cost.;
3  Remove duplicate packets ;
4  Extract field values from packets;
5  Split into flow;
6  for $i \in [1, L]$ do
7      if All 5-tuples are not empty then
8          for $j \in [1, M]$ do
9              Express a flow $V_f$ with fields according to 5-tuples: $\boldsymbol{v_{ij}}$ ;
10     else
11         Discard this packet;
12 if $File_n \in CTU - 13$ then
13     Labeled by CTU-13's annotations:
         $\boldsymbol{V} = concat(label, flatten(\boldsymbol{v}))$;
14     Merge the same malicious traffic into one file;
15     return $\boldsymbol{V}_{Neris}, \boldsymbol{V}_{Rbot}, \boldsymbol{V}_{Virut}$ ;
16 else
17     Labeled by ISCX2012's $*.xml$ files:
         $\boldsymbol{V} = concat(label, flatten(\boldsymbol{v}))$;
18     return $\boldsymbol{V}_{BFS}, \boldsymbol{V}_{HDos}, \boldsymbol{V}_{Inf}, \boldsymbol{V}_{Normal}$;
```

tasks, and use the word embedding method to vectorize it. The vectorization process is shown schematically in figure 4.

Empirically, before fields selection and vectorization, the following points need to be noted:

1. Different fields perform embedded encoding respectively. It means that even if there is the same value in the two fields, they should be encoded as different vectors. It is noticeable that different fields values are given different meanings even if they are the same.
2. Both "Sequence Number" and "Acknowledgement Number" fields are

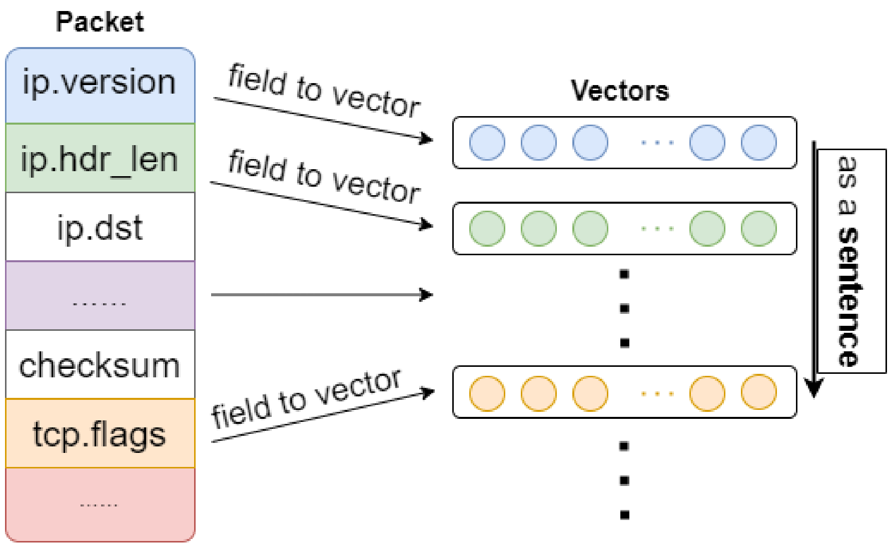


Figure 4: Schematic diagram of vectorization process

represented by a 4-byte number. Generally, the raw number of these fields is very large and with characteristics of randomicity. Instead of a raw number, we choose the relative number to express the relationship between packets. For example, given a "Sequence Number" of a packet, the "relative Sequence Number" is calculated by subtracting the "raw Sequence Number" of the first packet in the flow. Through doing this, the "tcp.seq" and "tcp.ack" values are limited to a relatively small range, especially the first few packets in the flow.

3. Ip address pair are fixed in a flow and provide very limited information for detecting network behavior, and even cause a certain degree of overfitting. Therefore, IP addresses are used as alternative features to generating vectors in this paper.

In this work, considering the similarity between field values in the packet header, we utilize the GloVe model to precode our fields feature and generate the vector dictionary of all fields obtained in the preprocessing. After applying the embedding to fields of the packet header, each field is embedded, and reshaped to an N-dimensional vector. Finally, to further explore the impact of packets and fields number on the identification of malicious traffic, we set the number of fields and packets in-flow to be adjustable, and which must consistent with the input of the classification model. The pre-embedding process is expressed as:

$$x_{ij} = GloVe(v_{it}), \quad i \in [1, L], j \in [1, M] \tag{11}$$

4.3. Classification

Our goal is to detect whether network traffic is malicious at the flow level using the field vector obtained above. Inspired by the literature [35], we take the advantage of hierarchical attention framework to build the flow-level vector progressively from the field vectors above obtained, and then be classified by a classification layer. The module consists of five layers, and take charge of the following operations respectively: field encoding, field attention, packet encoding, packet attention, and classifying. The overall framework of the classification module is shown in Figure 5. Each of these components is detailed in the following subsections.

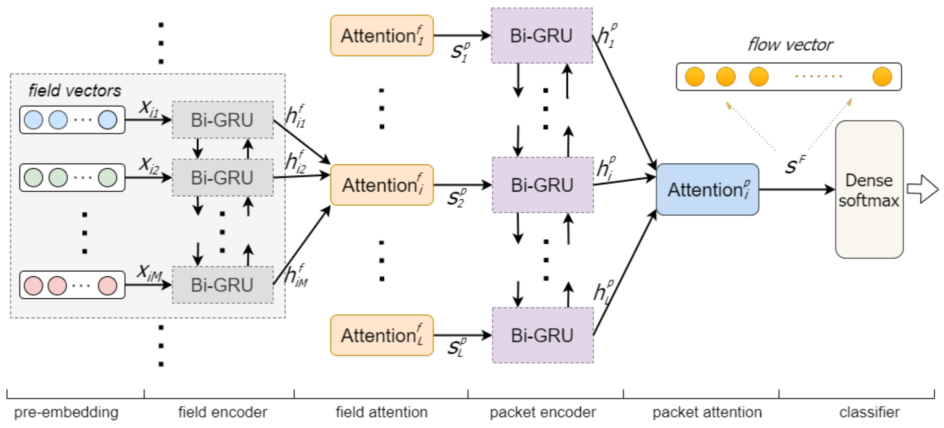


Figure 5: The framework of classification process

4.3.1. Field encoding layer

The field vectors are input to the classification module in the field encoding layer. Although having a few association attributes through pre-embedding, the input field vectors still cannot be used as candidate input expression vectors for the packet vector. This is because the pre-embedded vector is generated through unsupervised learning and is rarely associated with the classification task information. In this layer, we use supervised Bi-GRUs to further encode the field vectors, and get annotations of fields by summarizing information from both directions for fields. The field vectors are input to Bi-GRUs respectively according to the data package they belong to. There are L groups Bi-GRU, where L is the number of packets we chose in a flow. Take the input sequence x_{ij}, The output of this layer will be computed using equation (6) as follows:

$$h_{ij}^{f} = BiGRU(x_{it}), \quad i \in [1, L], j \in [1, M] \tag{12}$$

4.3.2. Field encoding layer

Not all fields contribute equally to the representation of the packet feature. Hence, we introduce the attention mechanism to extract such fields that are important to the meaning of the packets and aggregate the representation of those informative fields to form a packet vector. The same as the field encoding layer, there are L attention units. For input h_{ij}, the packet vector can be computed with equation (10) as:

$$s_{i}^{p} = ATT(h_{ij}), \quad i \in [1, L], j \in [1, M] \tag{13}$$

where s_i^p is the packet vector, which Is an abstract representation of a packet.

4.3.3. packet encoder layer

As mentioned above, we consider each flow as a paragraph which constructed by packets. From this layer, we aim to build the flow vector by packet vector which is output in filed attention layer. Similar to the field encoding layer, in this layer, we will apply Bi-GRU to get annotations of packets from the packet vectors. Given the previous layer output s_i^p, we compute the output of the packet encoder layer as:

$$h_{i}^{p} = BiGRU(s_{i}), \quad i \in [1, L] \tag{14}$$

where h_i^p is the packet vector, which summarizes the neighbor packets around packet i but still focuses on packet i.

4.3.4. Packet attention layer

After being encoded by Bi-GRU, the annotations of packets have been incorporated contextual information of the packet vectors, and can be used for some classification tasks [36]. In this work, to select the packets that contribute the most to the correct classification of traffic, we use the attention mechanism one more time. Finally, the flow vector is computed by the following equation:

$$s^{f} = ATT(h_{i}^{p}), \quad i \in [1, L] \tag{15}$$

where s^f is the flow vector that summarizes all the information of packets.

4.3.5. Classification layer

Following the encoding layer and attention layer at field-level and packet-level, the classification layer aim to classify the network flow by using the flow vector generated in the previous layer. In this layer, we apply a dense layer to compress the dimension of the flow vector, and a *softmax* classifier to predict the classification of flow, the co-operation can be expressed as:

$$p = softmax(dense(s^f)) \tag{16}$$

where p is the predicted classification, dense is the fully connected network, and *softmax* classifier function is similar to equation (8).

Based on the five layers described above, the network flow is classified by the flow-level vector generated from field-level vectors step by step. We consider that the flow-level vectors obtained by the trained model have the ability to represent important characteristics of flows for classifying network flows, and the trained model can serve different classification tasks with training the classification layer again.

5. Experiments and evaluations

In this section, we describe the experiments about validating the effectiveness of the detection model, and compare performance with previous works as a baseline.

5.1. Data sets

In this work, raw traffic is used as the source data for model input, thus the source datasets must contain raw traffic, and preferably public data sets as the baseline. Based on the above requirements, the following data sets are chosen.

Dataset CTU-13 was published by the team members of the Stratosphere IPS project [37], which is supported by the CTU University of Prague in the Czech Republic, and so far, the data set is still adopt by many studies [38, 39, 40]. This dataset contains malicious, mixed, and normal traffic. There are thirteen scenarios in CTU-13, and each scenario includes the malicious traffic raw pcap file generated by a specific bot or malware separately. Our method is based on network traffic at the flow level, thus the malicious traffic caused by icmp-dos cannot be detected. Besides, deep learning models are usually trained with a large amount of data, and then the malicious samples with a small amount of data in this data set will not be selected. Finally, we chose nine scenarios in this work, which contain malicious traffic generated by three types of malware, including Neris, Rbot, and Virut. According to the literature[37] description, we filter the malicious traffic in each scenario in strict accordance with the traffic IP address and the protocol used by the malicious traffic, and then merge the scenarios according to the botnet.

Data set ISCX2012 was created by Shiravi et al. and published in the paper [41]. This data set includes 7 days of network activity which consists of malicious and normal traffic. Among them, malicious traffic is divided into four categories according to its behavior, namely Infiltrating the network from inside, HTTP Denial of Service, Distributed Denial of Service, and Brute Force SSH. Besides, there are two profiles developed by authors in this data set, which was used to dynamically generate a new malicious traffic detection data set with normal as well as malicious network behavior. Although having an increasing age, this data set is still widely used in many studies [42, 43, 44]. Considering the types of existing malicious traffic in the CTU-13 dataset, in this work, we select four scenarios of the ISCX2012 dataset, except Distributed Denial of Service. In our data sets, the normal activity traffic data is derived from Saturday data in this dataset, and we filtered the malicious traffic of the dataset according to the XML file provided in the literature [41].

Table 3 shows the details of network flows extracted for different malicious flows. We can see that the number of malicious traffic in the two data sets is imbalanced. Therefore, in the multi-classification experiment, we use the random sampling method to establish a relatively balanced data set according to different experimental contents.

As mentioned above, our approach is also suitable for the detection of encrypted malicious traffic. We select DataCon-eta Dataset to verify our proposed approach in this paper. DataCon-eta Dataset is an open dataset used in the direction of encrypted malicious traffic detection in the 2020 DataCon big data security analysis competition [45]. The data set comes from malware and normal software collected from February to June 2020. Among them, malicious traffic is the encrypted traffic generated by malware (all exe types), and white traffic is the encrypted traffic generated by normal software (all exe types). We used all the traffic in the data set, except 61 UDP flows, the detail is shown in Table 4.

Table 3: Traffic composition in the data sets.

Dataset	Botnet	Activity[1]	Number of flows	Protocol
CTU-13	Neris	Spam, CF	111267	TCP
	Rbot	DDos	35959	TCP & UDP
	Virut	Spam, PS	33670	TCP
ISCX2012	-	BFS	4960	TCP
	-	HDos	3577	TCP
	-	Inf.	9977	TCP
	-	Normal	114936	TCP & UDP

[1] CF: Click Fraud; PS: Port Scan; BFS: Brute Force SSH; HDos: HttpDos; Inf.: Infiltrating.

Table 4: The detail of DataCon-eta data set.

Data set	Label	Number of flows	
		train	test
DataCon-eta	black	42817	51393
	white	18522	12996

5.2. Metrics

Several metrics are used to evaluate the performance of the proposed model: accuracy (ACC), precision (P), Recall (Rec) and $F-score$. ACC is a commonly used metric, which reflects the overall performance of the model. P measures the number of correct classifications that are predicted to be true malicious traffic. Rec is the fraction of real malicious traffic correctly detected. $F-score$ is calculated from P and Rec, which offers a summarized performance score. The calculation formula of the metrics are as follows:

$$\begin{aligned} ACC &= \frac{TP+TN}{TP+TN+FP+FN} \\ P &= \frac{TP}{TP+FP} \\ Rec &= \frac{TP}{TP+FN} \\ F-score &= 2 \times \frac{P * Rec}{P+Rec} \end{aligned} \tag{17}$$

We consider the label of malicious traffic as "true" and benign traffic as "false" in the detection tasks. Where, TP is the number of instances correctly predicted as "true"; FP is the number of instances incorrectly predicted to be "true"; FN is the number of instances incorrectly predicted to be "false"; TN is the number of instances correctly predicted to "false".

In addition, we use Receiver Operating Characteristics (ROC) and Area Under Curve of ROC (AUC-ROC) to evaluate the performance of a multi-class model in this paper. AUC-ROC represents the ability of a model to distinguish between positive and negative examples. Its value is between 0 and 1, and the larger the model, the better the performance.

5.3. Experimental design

In this section, we will gradually verify the effectiveness and performance of the method through experiments. The experimental design will be considered from the following aspects. First, the effectiveness of malicious traffic detection is usually our most concern. We design a binary classification model to detect malicious traffic, and use the above two data sets for training and testing. Second, considering that the flow-level vectors built by our model have the ability to detect malicious traffic generated by different malware from traffic activity, our second experiment tested the multi-classification effectiveness of the model. Thirdly, we show the process of constructing the optimal detection model by parameter modulation. In the last experiment, we verified the effectiveness of the algorithm on the datacon-eta data set and made a brief comparison with traditional machine learning algorithms. All experiments were performed on the same host, and Table 5 provides a detailed description of our experimental configuration. In the last experiment, we verified the effectiveness of the algorithm on the previously mentioned encrypting malicious traffic data set, and compared it with the traditional machine learning algorithm in performance.

Table 5: Experimental environment configurations.

Item	**Configuration**
Operating System	Ubuntu 18.04.5 LTS
Hardware	Intel(R) Xeon(R) Gold 6132, 256G
GPU	GeForce RTX 2080Ti GPU
Python Version	Python 3.8.8
Framework	Pytorch 1.9.1

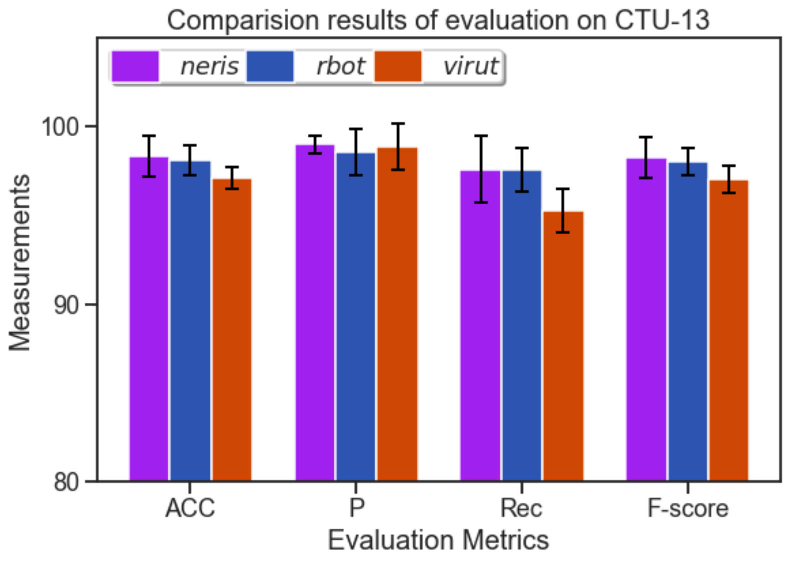
5.4. Evaluations

5.4.1. Experiment of bi-classification

In the first experiment, we tested the effectiveness of the proposed model when each type of malicious traffic is trained separately, and the benign traffic is combined from the two data sets. We applied ten-fold cross-validation to each malicious traffic, and Table 6 shows the optimal results, which reflects the effectiveness of our proposed approach. At the same time, we compare the stability of the model in the above experiment, and Figure 6 shows the identification performance comparison of each malicious traffic in different data sets. The colored bar represents the average value of each evaluation metrics, and The black error bars are calculated by the ten-fold crossover experiment. According to the experimental evaluation results shown in the image, the proposed approach has good performance in binary classification tasks.

5.4.2. Experiment of 4 classification

The finer-grained detection of malicious traffic is generally related to the multi-classification performance of the model. In the second experiment, we trained the model on two data sets and test the multi-classification performance respectively, and the experiment is divided into 3 scenarios, including



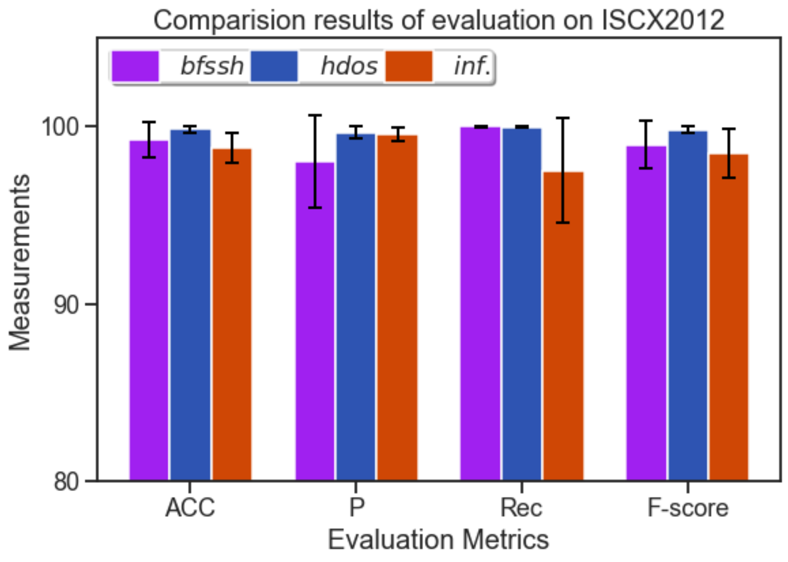


Figure 6: Performance of different malicious traffic in Data Set CTU-13 and ISCX2012

Table 6: Performance evaluation results.

Dataset	**Classification**	**ACC %**	**P %**	**Rec %**	**F-score**
CTU-13	Neris	99.20	99.32	98.87	0.9910
	Rbot	99.16	98.85	99.37	0.9911
	Virut	99.12	99.40	98.90	0.9915
ISCX2012	BFS	100	100	100	1.0000
	HDos	99.95	99.91	100	0.9995
	Inf.	99.45	99.09	99.67	0.9938

[1] BFS: Brute Force SSH; HDos: HttpDos; Inf.: Infiltrating.

4 classifications for each data set and 7 classifications for two data sets. In order to obtain a balancing data set between various types of traffic, we random re-selected the traffic in the data set according to the Table 3, and generated three subsets as shown in Table 7. In each scenario, we used five-fold cross-training to obtain the optimal model.

Table 7: Data sets in multi-classification experiments.

Scenario	**Data Set**	**Number of flows**		
		overall	**train**	**test**
1	**CTU-13**	295832	264595	31237
2	**ISCX2012**	30839	27899	2939
3	**CTU-13 & ISCX2012**	130290	121258	9032

Although we have considered the balance of data sets, in order to more comprehensively represent the characteristics of network traffic, all flows in Table 4 were used in the experiment. ROC curve and AUC-ROC value to evaluate the multi-classification model, which is rarely affected by the size and balance of the train set, and generally will not change significantly with the change of the proportion of positive and negative samples. In the first two scenarios, we trained four classification models in the two data sets respectively, and Figure 7 shows the ROC curve and AUC-ROC value for the performance evaluation of the proposed approach. The results show that the proposed approach is effective in multi-classification scenarios with different data sets, and different data sets have different sensitivity to the approach.

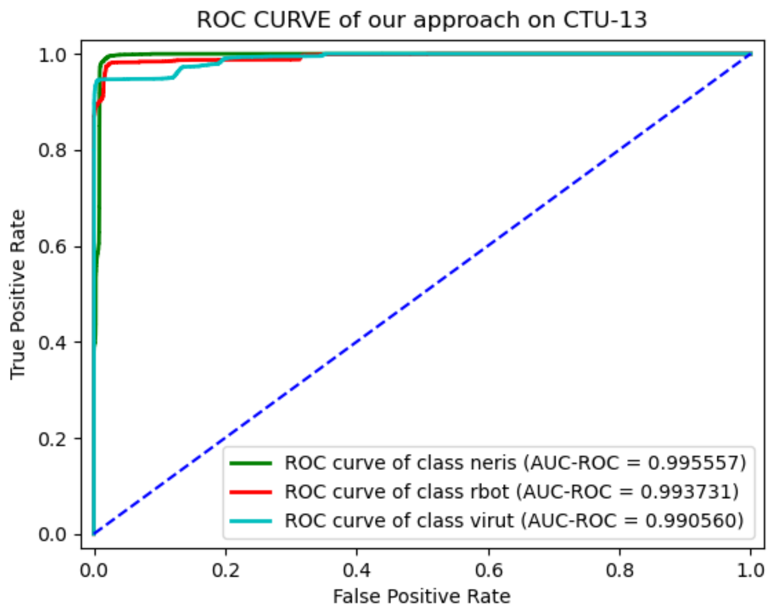
As is shown in Figure 7, our approach achieves the best performance on dataset ISCX2012, especially for BFSSH traffic detection. In experiment scenario 1, the experimental results for the CTU-13 data set are relatively inferior to ISCX 2012 data set in scenario 2. The details can be obtained from the confusion matrix of the evaluation results shown in Figure 8. Considering the idea of the approach proposed in this paper, the experimental results illustrate the following points:

Firstly, the method proposed in this paper is based on the flow as the sample, and the more salient the behavior characteristics of a single flow, the easier it is to be identified. In this experiment, the data set ISCX2012 is classified based on malicious traffic activities, while CTU-13 is classified based on the botnet, in which the behaviors of different malware may overlap. Therefore, on the whole, the detection performance of the data set ISCX2012 is better than that of the CTU-13 data set. As an example, both Neris and Virut share spam as an activity attack in data set CTU-13, which may be one of the reasons for the low detection rate for these two types of malicious traffic.

Secondly, For Rbot malware traffic, its main attack activity is UDP DDos. According to the five-tuple rule, the number of UDP packets that can be constructed into a stream in the data set is relatively too small, and the generated traffic is not enough to complete effective deep learning network training. However, its salient UDP characteristics are easy to be detected by the model, and the experimental results show a misjudgment of UDP traffic in normal traffic.

Thirdly, considering that malicious traffic usually uses unconventional general TCP/UDP ports, in this work, the port number is used as one of the characteristic values to participate in the training of the model, and which makes it easier to detect some malicious traffic that uses special ports, such as BFSSH traffic.

We compared the accuracy of the proposed approach with [38] and [42] as a baseline, which also use deep learning algorithms. As shown in Table 8,



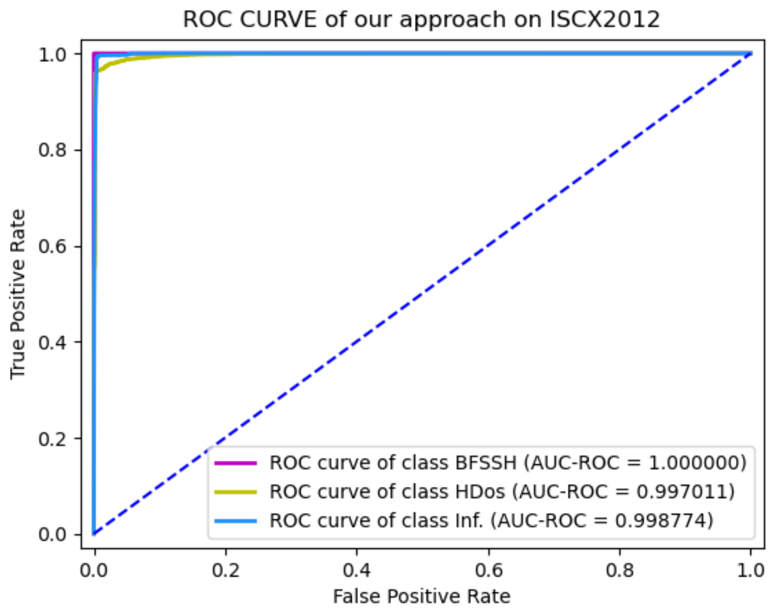


Figure 7: Detection performance in the first two data set respectively

the method proposed in this paper also shows a similar trend on the CTU-13 data set with [38], but the accuracy rate has been greatly improved. Table 9 shows the comparison between our approach and [42], and the accuracy rate has also improved.

Table 8: Comparison with baseline results in CTU-13.

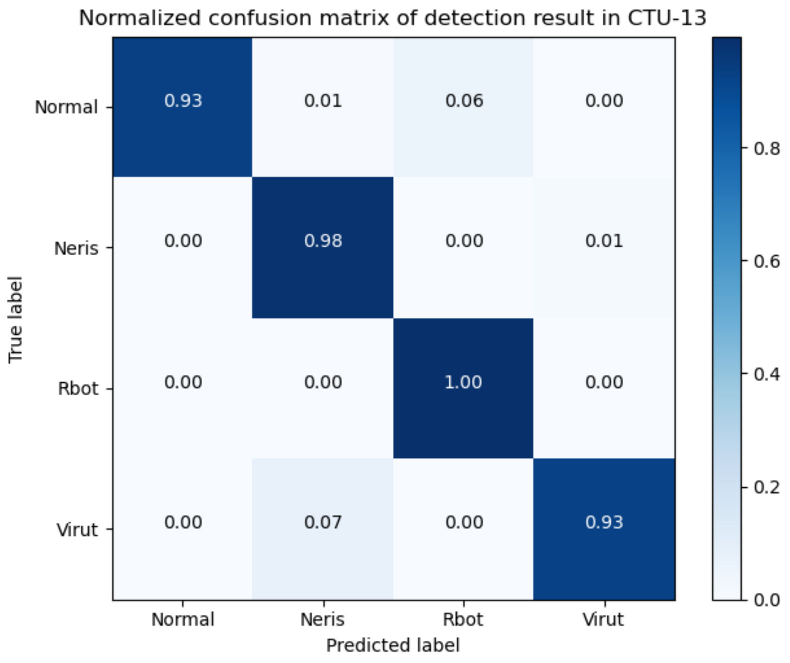
Botnet	**Ref. [38]**			**our approach**		
	Acc.	**Rec.**	**F-score**	**Acc.**	**Rec.**	**F-score**
Neris	0.635	0.635	0.714	**0.955**	0.989	**0.956**
Rbot	0.999	0.999	1.000	0.970	1.000	0.971
Virut	0.547	0.547	0.606	**0.925**	0.93	**0.925**

Table 9: Comparison with baseline results in ISCX2012.

Botnet	**Ref. [42]**		**our approach**	
	AUC-ROC	**F-score**	**AUC-ROC**	**F-score**
BFSSH	1.000	0.959	1.000	**0.999**
HDos	0.984	0.569	0.989	**0.963**
Inf.	0.962	0.720	0.992	**0.987**

In the last scenario of this experiment, we combined the two data sets to test the multi-classification performance of the proposed approach in the mixed data set. The newly generated data set contains all malicious traffic of ISCX2012 and part of malicious traffic randomly extracted from the CTU-13 data set, as shown in Table 7. We also used the ROC curve and confusion matrix to evaluate the detection performance of the trained model, and the results are shown in Figure 9 and Figure 10.

From the evaluation results in Figure 9, the 7 classification model still has good classification performance, and the ROC curve for different types of



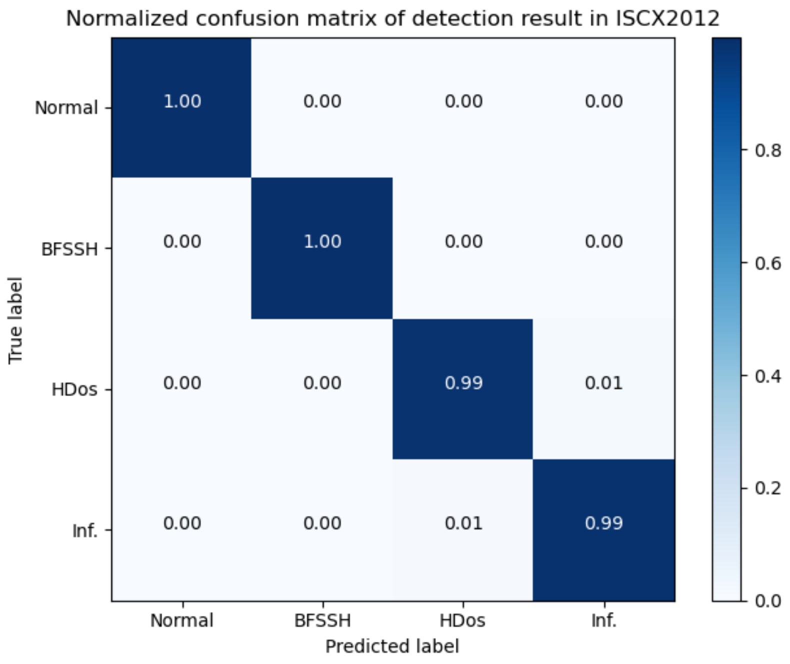


Figure 8: Confusion matrix of detection results in the first two data set respectively

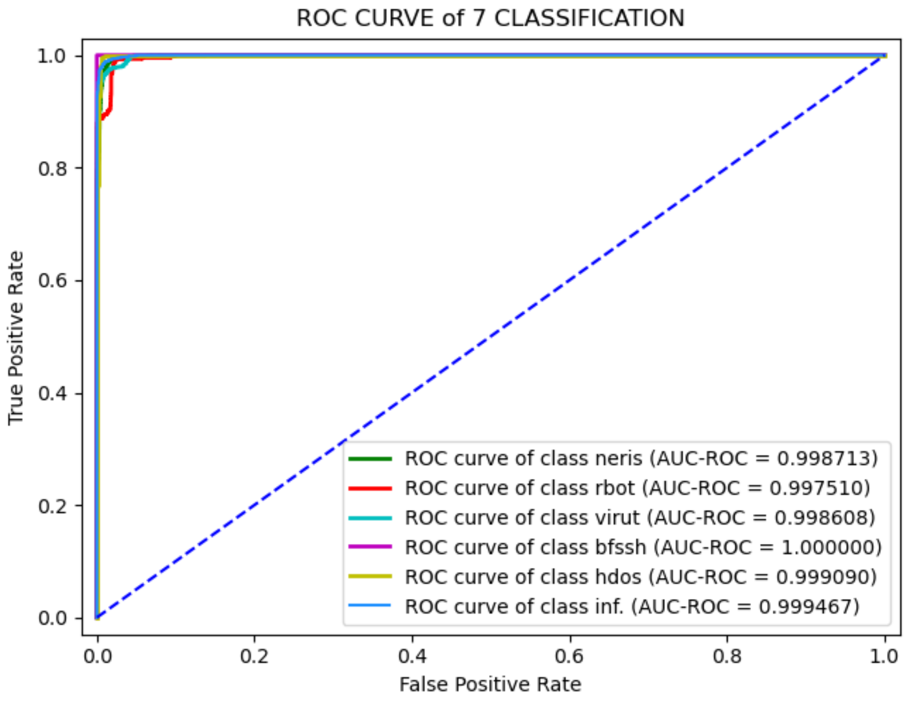


Figure 9: Performance evaluation of 7 classification models

malicious traffic detection maintains a sharp similar to the four classification models. So we can draw a conclusion that the detection performance of the proposed method is rarely affected by the number of malicious traffic types, and as shown in Figure 10, the same conclusion can also be obtained from the confusion matrix of the test results. Therefore, we believe that the proposed approach can effectively extract the relatively independent salient features of malicious traffic of different activity types, and the constructed flow vector can express the salient characteristics of flow behavior.

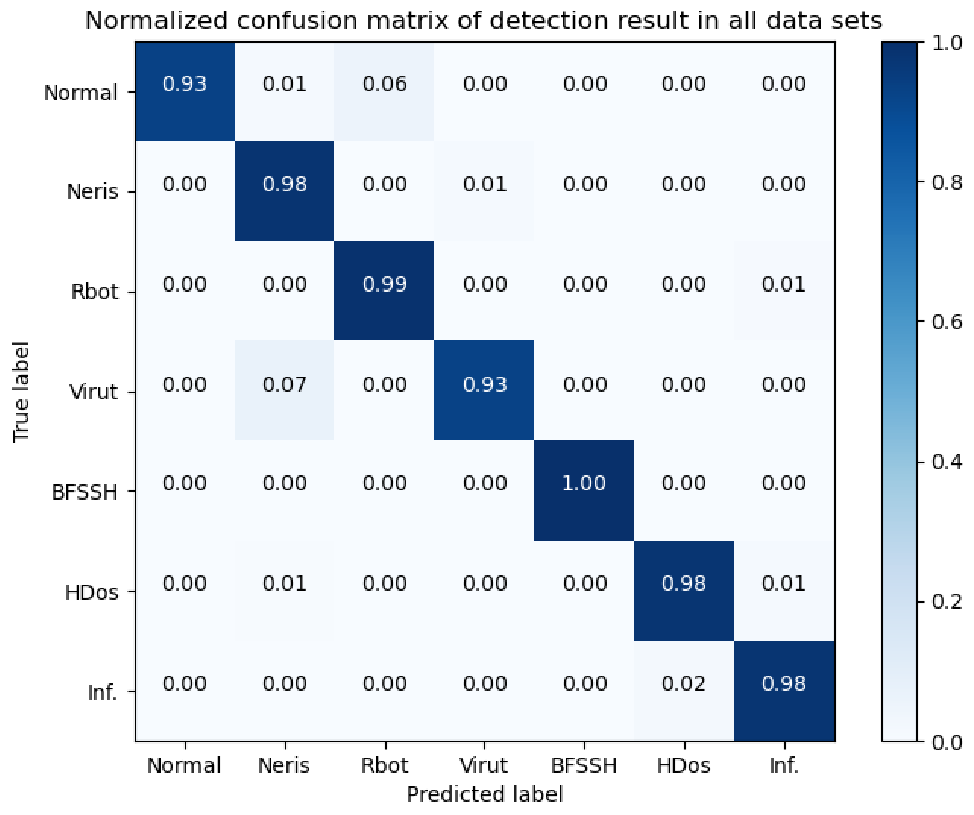


Figure 10: Confusion matrix of 7 classification test results

5.4.3. Experiment of 7 classification

In order to improve the efficiency of the model, under the premise of ensuring accuracy, we hope that the number of packets constructing the flow vector is as small as possible. In this experiment, we use the 7 classification model as the basic model for testing to evaluate the impact of the number of data packets contained in each stream on the detection accuracy. We made statistics on the median and mean value of data packets contained in different types of traffic, as shown in Table 10.

Table 10: Statistics for data sets.

No.	**Type**	**Intra flow packet statistics**			
		Maximum	**Minimum**	**Average**	**Median**
1	**Neris**	100	1	8.3	3
2	**Rbot**	100	1	9.9	3
3	**Virut**	100	1	7.2	4
4	**BFS**	100	2	18.8	21
5	**HDos**	100	1	4.4	3
6	**Inf.**	100	1	1.8	1
7	**Normal**	100	1	21.7	10

[1] BFS: Brute Force SSH; HDos: HttpDos; Inf.: Infiltrating.

According to these statistics, we carried out 7 scenarios and limited the maximum number of packets to 5, 10, 20, 30, 50, 70 and 100 respectively, and the data sets used the 7 classification data sets in Experiment 2. In this experiment, only the above parameters are adjusted, and other parameters of the model remain unchanged. Figure 11 shows the change in accuracy during parameter tuning.

5.4.4. Experiment on encrypted traffic

As mentioned above, the data set DataCon-eta is divided into two parts: train set and test set, and each subset is divided into black samples and white samples respectively. All files labeled black are encrypted malicious traffic generated by malware, and all encrypted traffic in files labeled white are regarded as normal traffic. In this experiment, we use the train set data training to find the optimal model, and evaluate the performance of the

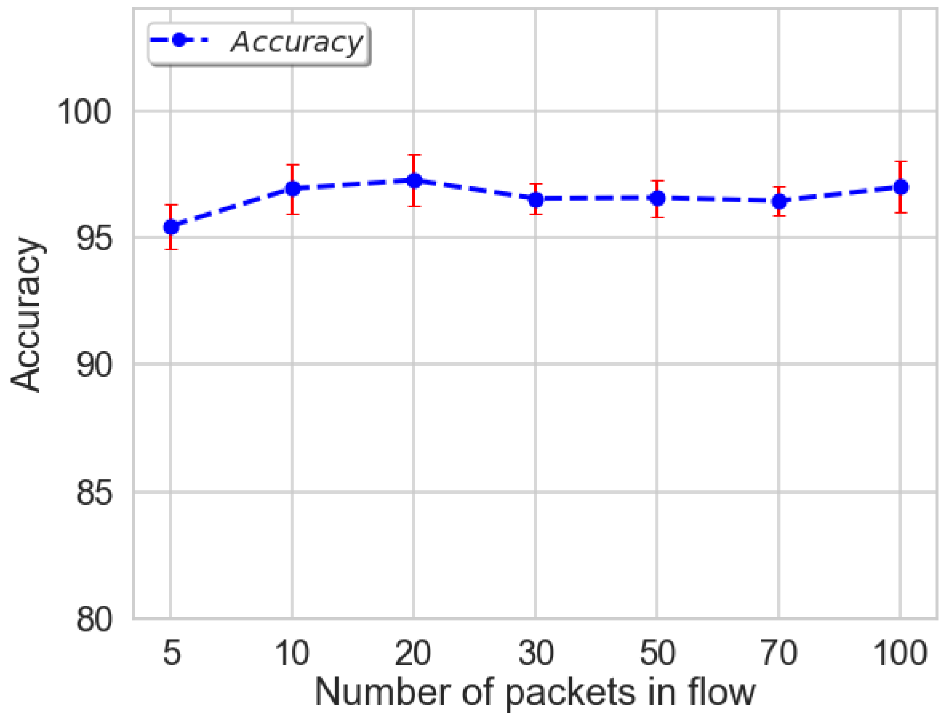


Figure 11: influence of flow length

model on the test set. As shown in Table 4, the black and white samples of the train set data are not balanced. Therefore, we randomly select some data from the black samples to participate in the training, and carry out training by setting different hyperparameters to find the optimal detection model. We find the optimal model by setting different hyperparameters. Table 11 shows the accuracy under different learning rates and batch size parameters. We got the best accuracy 92.734% when the learning rate is equal to 0.05 and batchsize is equal to 128.

Table 11: Accuracy under different hyperparameters (%).

LR \ BS	**16**	**32**	**64**	**128**	**256**	**512**
0.005	91.508	90.549	89.651	87.402	89.435	88.968
0.01	91.664	92.218	91.984	91.990	89.539	89.155
0.02	92.366	92.133	91.863	91.602	90.735	88.231
0.05	91.872	91.540	91.926	**92.734**	91.088	90.836
0.08	90.603	90.797	91.327	91.588	91.181	91.328

BS: Batch Size; LR: Learning Rate.

In addtion, we compared the proposed approach with several other methods in generalization. There is no matching related paper in the existing research work based on the DataCon-eta data set. Therefore, we selected two machine learning methods and common statistical features to carry out comparative experiments. As shown in Table 12, our approach has higher detection performance than traditional machine learning methods that use statistical features.

6. Conclusions

In this paper, we introduced a novel flow-vector generation approach for detecting malicious traffic based on a hierarchical attention network. The flow vector is gradually constructed by the field vector embedded from the

Table 12: Performance comparison.

Methods	Features	ACC	P	Rec	F-score
RF	Packet length	0.880	0.977	0.870	0.921
RF	Flow statistics	0.920	0.968	0.930	0.949
RF	Hosts statistics	0.814	0.972	0.790	0.872
NB	Server IP	0.784	0.987	0.740	0.846
NB	Certificate	0.870	0.964	0.870	0.914
Our method	**Field value**	**0.927**	0.948	**0.962**	**0.955**

RF: Random Forest; NB: Naive Bayes.

packet header field in the raw traffic, thus the flow vector contains the field level and packet level context information. The header field can reflect the comprehensive information of the packets, and the combination of multiple packets in the flow can reflect the behavior of the network traffic. Firstly, we use the word embedding method to embed the header field into the field vectors, so that it can preliminarily learn the context information. Then, two layers of attention mechanism are employed to construct packet vectors and flow vectors respectively. Finally, due to the high dimension of the generated flow direction, we use full join and function for dimensionality reduction classification. We consider that the flow vector can express the behavior and pattern of traffic, and distinguish benign and malicious traffic through it. We selected part of the data from two public data sets and combined them into an experimental data set. The experimental results showed the effectiveness of the approach. The approach proposed in this paper only uses a few information of packet header and is independent of payload content information, so we consider that it may also be suitable for online or encrypted malicious traffic detection. For online detection tasks, the pre-embedding process needs to be completed first, because the generation of the field vector dictionary requires a lot of data to train. For encrypted malicious traffic detection tasks, the flow vector can be used as one of the features of the detection model to improve the detection accuracy. We will focus on the two above problems in future work.

Declaration of competing interest

The authors have no financial conflict of interest that might be construed to influence the results or interpretation of this manuscript. Thank you again for your consideration of this manuscript.

Acknowledgments

This research was supported by National Natural Science Foundation of China under Grant (No.U20B2046, No.61772321), and the National Key Research and Development Program of China under Grant (2021YFB2012402), Shandong Natural Science Foundation (ZR202011020044), and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme(2019). The author would like to thank Qianxin Company for sharing the dataset-eta dataset.

References

[1] R. Samani, Advanced threat research report, `https://www.mcafee.com/enterprise/en-us/lp/threats-reports/oct-2021.html`, 2021.

[2] A. L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Communications Surveys & Tutorials 18 (2016) 1153–1176.

[3] P. Mishra, V. Varadharajan, U. K. Tupakula, E. S. Pilli, A detailed investigation and analysis of using machine learning techniques for intrusion detection, IEEE Communications Surveys & Tutorials 21 (2019) 686–728.

[4] J. Kim, J. Kim, H. L. T. Thu, H. Kim, Long short term memory recurrent neural network classifier for intrusion detection, 2016 International Conference on Platform Technology and Service (PlatCon) (2016) 1–5.

[5] C. Yin, Y. Zhu, J.-l. Fei, X.-Z. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.

[6] A. Y. Javaid, Q. Niyaz, W. Sun, M. Alam, A deep learning approach for network intrusion detection system, in: EAI Endorsed Trans. Security Safety, 2016.

[7] N. Shone, T. N. N. . . Íc, V. D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Transactions on Emerging Topics in Computational Intelligence 2 (2018) 41–50.

[8] Z. Tian, C. Luo, J. Qiu, X. Du, M. Guizani, A distributed deep learning system for web attack detection on edge devices, IEEE Transactions on Industrial Informatics 16 (2020) 1963–1971.

[9] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, 2017 International Conference on Information Networking (ICOIN) (2017) 712–717.

[10] H. Liu, B. Lang, M. Liu, H. Yan, Cnn and rnn based payload classification methods for attack detection, Knowl. Based Syst. 163 (2019) 332–341.

[11] G. Mar'in, P. Casas, G. Capdehourat, Deepmal - deep learning models for malware traffic detection and classification, ArXiv abs/2003.04079 (2020).

[12] Z. Weng, T. Chen, T. Zhu, H. Dong, D. Zhou, O. Alfarraj, Tlsmell: Direct identification on malicious https encryption traffic with simple connection-specific indicators, Computer Systems Science and Engineering 37 (2021) 105–119. doi:10.32604/csse.2021.015074.

[13] Y. Fang, K. Li, R. F. Zheng, S. Liao, Y. Wang, A communication-channel-based method for detecting deeply camouflaged malicious traffic, Computer Networks 197 (2021) 14. doi:10.1016/j.comnet.2021.108297.

[14] P. Wang, S. Li, F. Ye, Z. Wang, M. Zhang, Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan, ICC 2020 - 2020 IEEE International Conference on Communications (ICC) (2020) 1–7.

[15] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, Z. Tian, A novel web attack detection system for internet of things via ensemble classification, IEEE Transactions on Industrial Informatics 17 (2021) 5810–5818.

[16] M. A. Ferrag, L. Maglaras, S. K. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, J. Inf. Secur. Appl. 50 (2020).

[17] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, M. Zhu, Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection, Ieee Access 6 (2018) 1792–1806.

[18] R.-H. Hwang, M.-C. Peng, N. Van-Linh, Y.-L. Chang, An lstm-based deep learning approach for classifying malicious traffic at the packet level, Applied Sciences-Basel 9 (2019). doi:10.3390/app9163414.

[19] R. C. Staudemeyer, Applying long short-term memory recurrent neural networks to intrusion detection, South African Computer Journal 56 (2015) 136–154.

[20] Z. Li, Z. Qin, K. Huang, X. Yang, S. Ye, Intrusion detection using convolutional neural networks for representation learning, in: ICONIP, 2017.

[21] R. Vinayakumar, K. P. Soman, P. Poornachandran, Applying convolutional neural network for network intrusion detection, 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2017) 1222–1228.

[22] A. Aleesa, B. Zaidan, A. Zaidan, N. M. Sahar, Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions, Neural Computing and Applications 32 (2020) 9827–9858.

[23] H. Huang, H. Deng, Y. Sheng, X. Ye, Accelerating convolutional neural network-based malware traffic detection through ant-colony clustering, J. Intell. Fuzzy Syst. 37 (2019) 409–423.

[24] J. Xie, S. Li, X. chun Yun, Y. Zhang, P. Chang, Hstf-model: An http-based trojan detection model via the hierarchical spatio-temporal features of traffics, Comput. Secur. 96 (2020) 101923.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NIPS, 2013.

[26] E. Min, J. Long, Q. Liu, J. Cui, W. Chen, Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest, Secur. Commun. Networks 2018 (2018) 4943509:1–4943509:9.

[27] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: EMNLP, 2014.

[28] A. F. Diallo, P. Patras, Adaptive clustering-based malicious traffic classification at the network edge, IEEE INFOCOM 2021 - IEEE Conference on Computer Communications (2021) 1–10.

[29] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Ba Hdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, Computer Science (2014).

[30] A. P. Parikh, O. Täckström, D. Das, J. Uszkoreit, A decomposable attention model for natural language inference, in: EMNLP, 2016.

[31] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR abs/1409.0473 (2015).

[32] E. Papadogiannaki, S. Ioannidis, Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware, Sensors 21 (2021) 21. doi:`10.3390/s21041140`.

[33] M. A. Ambusaidi, X. He, P. Nanda, Z. Tan, Building an intrusion detection system using a filter-based feature selection algorithm, IEEE Transactions on Computers 65 (2016) 2986–2998.

[34] T. Mikolov, W. tau Yih, G. Zweig, Linguistic regularities in continuous space word representations, in: NAACL, 2013.

[35] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. H. Hovy, Hierarchical attention networks for document classification, in: NAACL, 2016.

[36] C. Liu, L. He, G. Xiong, Z. Cao, Z. Li, Fs-net: A flow sequence network for encrypted traffic classification, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications (2019) 1171–1179.

[37] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Comput. Secur. 45 (2014) 100–123.

[38] G. Marín, P. Casas, G. Capdehourat, Deep in the dark - deep learning-based malware traffic detection without expert knowledge, 2019 IEEE Security and Privacy Workshops (SPW) (2019) 36–42.

[39] Y. Yu, J. Long, Z. Cai, Session-based network intrusion detection using a deep learning architecture, in: MDAI, 2017.

[40] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, M. Colajanni, Deep reinforcement adversarial learning against botnet evasion attacks, IEEE Transactions on Network and Service Management 17 (2020) 1975–1987.

[41] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (2012) 357–374.

[42] F. Folino, G. Folino, M. Guarascio, F. S. Pisani, L. Pontieri, On learning effective ensembles of deep neural networks for intrusion detection, Inf. Fusion 72 (2021) 48–69.

[43] M. Wang, Y. Lu, J. Qin, A dynamic mlp-based ddos attack detection method using feature selection and feedback, Comput. Secur. 88 (2020).

[44] T. Aldwairi, D. Perera, M. A. Novotny, An evaluation of the performance of restricted boltzmann machines as a model for anomaly network intrusion detection, Comput. Networks 144 (2018) 111–119.

[45] Qianxin, Datacon-eta dataset, `https://datacon.qianxin.com/opendata/openpage?resourcesId=6`, 2020.