Wiley
Security and Communication Networks
Volume 2024, Article ID 8725832, 16 pages
https://doi.org/10.1155/2024/8725832

WILEY

Research Article

HMMED: A Multimodal Model with Separate Head and Payload Processing for Malicious Encrypted Traffic Detection

Peng Xiao, Ying Yan, Jian Hu, and Zhenhong Zhang

Information Center of China Southern Power Grid Yunnan Power Grid Co., Ltd., Kunming 650011, China

Correspondence should be addressed to Ying Yan; yypurple2023@163.com

Received 5 September 2023; Revised 22 April 2024; Accepted 16 May 2024; Published 30 May 2024

Academic Editor: A. Peinado

Malicious encrypted traffic detection is a critical component of network security management. Previous detection methods can be categorized into two classes as follows: one is to use the feature engineering method to construct traffic features for classification and the other is to use the end-to-end method that directly inputs the original traffic to obtain traffic features for classification. Both of the abovementioned two methods have the problem that the obtained features cannot fully characterize the traffic. To this end, this paper proposes a hierarchical multimodal deep learning model (HMMED) for malicious encrypted traffic detection. This model adopts the abovementioned two feature generation methods to learn the features of payload and header, respectively, then fuses the features to get the final traffic features, and finally inputs the final traffic features into the softmax classifier for classification. In addition, since traditional deep learning is highly dependent on the training set size and data distribution, resulting in a model that is not very generalizable and difficult to adapt to unseen encrypted traffic, the model proposed in this paper uses a large amount of unlabeled encrypted traffic in the pretraining layer to pretrain a submodel used to obtain a generic packet payload representation. The test results on the USTC-TFC2016 dataset show that the proposed model can effectively solve the problem of insufficient feature extraction of traditional detection methods and improve the ACC of malicious encrypted traffic detection.

1. Introduction

In the past few years, the network has witnessed a substantial surge in the volume of encrypted traffic, primarily due to the heightened emphasis placed on privacy protection by individuals. According to the 2022 Google Transparency Report, an overwhelming majority of the most popular 100 websites worldwide, accounting for approximately 97%, have opted for the Hypertext Transfer Protocol Secure (HTTPS) protocol to facilitate data transfer. Moreover, as of 2020, there has been a substantial increase in the ratio of encrypted traffic on the Internet, with the percentage soaring from around 50% in 2014 to approximately 95% [1]. Although the widespread use of encryption protocols ensures the confidentiality and integrity of user data, it also facilitates the spread of malicious software. Based on a study conducted by WatchGuard Technologies [2], the proportion of malware traffic encrypted with TLS/SSL reached 91.5% in the second quarter of 2021. Unfortunately, previous traffic analysis tools cannot be directly used in the detection of malicious encrypted traffic because they analyze and detect nonencrypted traffic based on plaintext payloads. Therefore, it is important to develop new methods that enable malicious encrypted traffic detection. In addition, encrypted traffic is so complex that it cannot be accurately categorized or detected by a single set of features (e.g., side channel features and raw traffic features). Therefore, how to extract multiple effective features from multimodality and fuse features in order to achieve excellent encrypted traffic representation is the key to achieving high-performance malicious encrypted traffic detection.

Malicious traffic detection is an important defense technique in network security [3], and many kinds of detection techniques have been developed as of today. The traditional detection methods are deep packet inspection (DPI) and port-based identification technology [4]. The

principle of DPI is to analyze the header and payload of network traffic, discover the specific character flow (fingerprint) in the traffic, and match the traffic with the character flow to achieve network traffic classification [5, 6]. DPI needs to decrypt data packets in detecting malicious encrypted traffic, which cannot guarantee the privacy of user data. Port-based identification technology is also not suitable for malicious encrypted traffic detection because most of today's applications use dynamic ports or transmit data through encrypted protocols [7]. At present, researchers mainly use detection methods based on machine learning and deep learning, and the performance of these methods is mainly determined by feature sets and algorithms [8]. Initially, the selection of feature sets relied on the knowledge of domain experts. However, manually selected feature sets are subjective and nonintuitive features may not be found [9], which will greatly affect the performance of classification models. After that, the machine learning-based feature selection method was developed [10, 11] to address the problem of subjectivity in manual feature selection by allowing machine learning algorithms to autonomously learn a suitable feature set from all available features. The method still has limitations in that it is unable to obtain high-order features. The way to obtain the feature set at this stage is called feature engineering. With the development of deep learning, some literature [12–15] uses the raw traffic as the input of the deep learning algorithm, which self-mines features to achieve end-to-end malicious encrypted traffic detection [16]. This method not only saves the work of manual feature extraction but can also extract nonintuitive features. However, to adapt to neural networks, some important features are often lost during data preprocessing. For example, to adapt to the input format of the neural network, the data packet will be padded or truncated to ensure that the length of the data packet is consistent. However, an important feature, packet length, will be lost [17]. In addition, using the entire raw traffic as the input of the model will lead to a sharp increase in the training time of the model. It can be seen that the end-to-end detection method also has its limitations.

End-to-end representation learning using deep learning can mine high-order features of traffic but lose some simple and intuitive features, and feature engineering approaches do not obtain high-order features but do not ignore simple and effective features. Combining the advantages of the two approaches, this paper proposes a hierarchical multimode malicious encrypted traffic detection model called HMMED. It aims to use rational and efficient methods to mine effective high-order features from payloads to portray the content of flows and to extract effective features from packet headers to portray the form of flows. The model fully extracts multiple effective features from multimodality to portray traffic comprehensively. To fully extract payload features to represent payload, we first trained a transformer's encoder as a pretraining layer model using a large amount of unlabeled encrypted traffic data. The generic byte encoding with learned byte context information is obtained by the encoder. In the next layer, the encoder is reused to obtain generic payload encoding. Then, considering that there is a correlation between payloads in traffic, we use a transformer to learn the relationship between payloads. To fully extract header features to represent headers, we select four important fields from the header and use BiLSTM to learn the header sequential features. Finally, we use a transformer to fuse the header features with the payload features and input them to the softmax classifier for classification to get the malicious traffic detection results.

This paper has the following three contributions:

(1) We propose a multimodal model to detect malicious encrypted traffic. The two modes of the model are headers and payloads. The features learned in the two modes complement each other to improve the detection accuracy.

(2) We design a self-supervised pretraining model suitable for traffic detection. The interbyte and interpacket dependencies are learned through the following two training tasks: BURST_payload filling and BURST_payload homology prediction, resulting in an excellent generic byte encoding.

(3) The data packet is divided into two modes as follows: packet header and packet payload. The packet header is represented by a four-tuple, which reduces the time overhead of model training and improves the training efficiency of the model.

The rest of this paper is organized as follows. Chapter 2 introduces the related work in the field of malicious encrypted traffic detection. Chapter 3 defines the problem of malicious encrypted traffic detection in this paper and introduces the proposed HMMED model in detail. Chapter 4 is the experiment and analysis. Chapter 5 is the conclusion.

2. Related Work

In this section, we introduce some recently proposed methods for malicious encrypted traffic detection. These methods can be divided into feature engineering-based methods and original traffic data-based methods according to the way of feature extraction and feature set selection. The characteristics of the two methods and the related literature will be presented in the following.

Feature engineering involves manipulating raw data to create new features that provide a more meaningful representation of the underlying problem [18]. There is much literature based on feature engineering in the field of malicious encrypted traffic detection. Hou et al. [19] presented an algorithm that utilizes exclusively the packet header fields from the original traffic to build a distinctive characteristic representation of the traffic. The algorithm selects some fields in the IP, TCP, and UDP protocols, uses the GloVe model to embed them into field vectors, then uses a two-layer attention network to gradually generate packet vectors and flow vectors containing context information, and finally uses the flow vectors as the input for classification tasks. Wang and Thing [20] novelly proposed the concept of encrypted traffic features and divided encrypted traffic

features into the following three categories: time-related traffic features, payload-based side-channel features, and ratio features. The detection framework they designed contains two layers. The first layer consists of LSTM, ResNet, and XGBoosting. Different categories of features are fed into different models based on their characteristics. The second layer uses random forest or average ensemble to fuse the classification results from the previous layer. Zheng et al. [21] used the DBSCAN algorithm for semisupervised clustering. A small amount of data, about 10%, was used for training. In addition, the authors analyzed the difference between malicious encrypted traffic and normal traffic. However, the model's tendency to cluster as many normal samples as possible leads to a high underreporting rate. Liu et al. [22] proposed a distance-based method. The method used the Gaussian mixture model (GMM) to calculate the distance between malicious samples and defined spurious labels according to distance. Then, the final classifier was trained by the eXtreme Gradient Boosting (XGBoosting) algorithm. Dong [23] improved the support vector machine (SVM) for solving the dataset imbalance problem and proposed a cost-sensitive SVM (CMSVM) algorithm which uses active learning to dynamically assign weights to each type of traffic. Experiments show that CMSVM is more accurate than SVM, ROS, and RUS on both the MOOR-E_SET dataset and the NOC_SET dataset. Shafiq et al. [10, 11] focused on the study of feature set selection algorithms [10], proposed a wrapper-based feature set selection method called CorrAUC, which filters features based on the AUC metric [11], and proposed a feature set selection framework, which uses the bijective soft set and the proposed Corr_ACC algorithm for feature set selection.

The method based on original traffic can also be called an end-to-end detection method. This method saves the work of manually designing features and avoids objectivity, but it also has its limitations. As highlighted in the introduction, there is a risk of losing significant information in the process. In addition, end-to-end detection methods are often time consuming [24]. Wang et al. [12] designed a model with a hierarchical structure, which for the first time successively used a CNN network and a BiLSTM network to extract the features of data packets and session flows. Experimental results show that this hierarchically constructed neural network structure can improve the model's performance. Bao et al. [13] proposed a hierarchical feature fusion and attention algorithm (HFFA) to improve performance. The method sequentially uses the BiLSTM network and attention-based BiLSTM network to extract features of packets and session flows. This method adopts the method of global hybrid pooling to fuse the payload length feature and payload content feature to avoid the problem that the global maximum pooling and the global average pooling exist in extracting feature granularity too coarse and too fine, respectively. Cheng et al. [14] proposed the RTETC method that was fed into the embedded representation of the first three packets and adopted multihead attention and 1D_CNN to extract the interaction within the traffic packet and the interaction between traffic packets. Lin et al. [15] proposed a new pretraining model for encrypted traffic detection and, for the first time, proposed a raw traffic representation model, the Datagram2Token model, which converts data packets into language-like tokens as the model input. Khodaverdian et al. [25] proposed a model combining convolutional neural networks (CNNs) and gated recurrent units (GRUs) to select appropriate migration candidate VMs and to diagnose whether a VM is latency sensitive. Khodaverdian et al. [26] proposed a hybrid model based on a combination of convolutional neural networks (CNNs) and gated recurrent units (GRUs) for classifying virtual machines in Microsoft Azure cloud services.

In addition to the challenge of extracting effective features in the field of encrypted traffic classification, there is also the problem of data imbalance. This is because the uneven distribution of samples from different classes can lead to degradation of classification performance. Soleymanpour et al. [27] presented a novel approach to deal with the problem of unbalanced data in encrypted traffic classification. By using a cost-sensitive convolutional neural network (CSCNN), the study was able to assign different misclassification costs during the training process, thus improving the classification accuracy. Experiments on the ISCX VPN-nonVPN dataset show the efficient performance of the CSCNN in traffic classification, traffic characterization, and application identification tasks, demonstrating its potential for dealing with class imbalance problems. Soleymanpour et al. [28] proposed a similar method in an earlier paper. Their experimental results show that the proposed model is about 2% higher than the deep packet method on average in classification performance.

Existing detection methods, whether based on feature engineering or end-to-end methods, cannot fully characterize traffic, resulting in insufficient feature extraction. In this paper, a multimodal feature fusion model is designed by combining the advantages of feature engineering, the end-to-end method, and the multimodal concept. The model's processing of mode one data is an end-to-end approach, while the processing of mode two data is a feature engineering approach. The hybrid method combines the advantages of the two methods in that the time spent will not be too large while ensuring the detection accuracy. In addition, for the problem of data imbalance, unlike the abovementioned research, this paper pretrains a submodel through a large amount of unlabeled data. The submodel can learn a general packet payload representation to enhance the model's recognition ability for minority classes.

3. HMMED

3.1. Problem Definition. In the field of traffic detection, there are three levels of detection methods, which are the packet level, the unidirectional flow level, and the bidirectional flow (biflow) level. Because the bidirectional flow contains all the data packets exchanged so as to have more comprehensive information, this paper implements malicious encrypted traffic detection at the bidirectional flow level. The original traffic R consists of several packets, denoted as $R = \{\text{packet}_1, \text{packet}_2, \ldots, \text{packet}_n\}$. Packets in the bidirectional flow are characterized by a consistent

five-tuple comprising the source IP, destination IP, source port, destination port, and protocol. The original flow is sliced according to the five-tuple, and the original flow is denoted as $R = \{\text{biflow}_1, \text{biflow}_2, ..., \text{biflow}_m\}$, where $\text{biflow}_i = \{\text{packet}_1, \text{packet}_2, ..., \text{packet}_t\}$. The malicious encrypted traffic detection problem to be solved in this paper can be expressed as a formula as follows:

$$f(\text{biflow}_i) \longrightarrow 0 \ldots n. \tag{1}$$

Given a bidirectional flow, the model f detects which traffic type the bidirectional flow is; each value refers to a traffic type.

3.2. Model Overview. To obtain effective features that can comprehensively portray encrypted traffic to improve the accuracy of malicious encrypted traffic detection, this paper proposes a hierarchical multimode malicious encrypted traffic detection model, named HMMED. The overall framework of the model is shown in Figure 1. The model is divided into three stages as follows: the representation learning stage, the feature fusion stage, and the classification stage.

3.2.1. Representation Learning Stage. The purpose of the representation learning stage is to obtain payload-level biflow representation and header-level biflow representation. The payload-level biflow representation focuses on flow content information. The header-level biflow characterization focuses on the flow form information (side channel information). To fully learn the payload-level biflow representation, we design a three-layer structure, which are the pretraining layer, the payload encoding layer, and the timing layer. The pretraining layer uses a large amount of unlabeled encrypted traffic to train a transformer's encoder. The encoder inputs a byte embedding sequence, learns the relationship between bytes, and outputs a generic byte encoding sequence. The payload encoding layer reuses the encoder from the pretraining layer and outputs the generic payload encoding. The generic payload encoding is the input unit to the transformer model in the timing layer. The transformer model learns the relationship between payloads and outputs through payload-level biflow representation. To fully learn the header-level biflow representation, we select four important fields to represent the header and design a timing layer. The header-level biflow representation containing the sequential information is obtained through the BiLSTM model in the timing layer.

3.2.2. Feature Fusion Stage. The feature fusion stage consists of two layers of structure, i.e., the transform layer and the feature fusion layer. In the transformer layer, we combine payload-level biflow representation with header-level biflow representation to make the payload representation and header representation of the same packet physically adjacent to each other. In the feature fusion layer, the transformer model is used to learn the correlation between payload representation and header representation. The model outputs the fused biflow representation.

3.2.3. Classification Stage. The fused biflow representation is input into the softmax classifier to complete the final encrypted traffic classification.

3.3. Data Preprocessing. The data preprocessing module is used to convert the original traffic data into a fixed-format matrix that the model can accept. This module mainly includes traffic segmentation, traffic cleaning, traffic slicing, and traffic encoding.

(1) Traffic segmentation: the purpose of traffic segmentation is to divide the data packets in the PCAP file into different biflows according to the five-tuple (<protocol type, source IP, source port, destination IP, destination port>), which is convenient for subsequent marking and processing. The segmentation process uses pkt2flow, which segments packets into biflows based on the source address, destination address, source port, and destination port and saves each biflow to a single PCAP file.

(2) Traffic cleaning: the purpose of traffic cleaning is to remove the dirty data in each biflow after segmentation, including ARP packets, DNS packets, and retransmission packets, to minimize their impact on the classification.

(3) Traffic slicing: since the length of each packet is different and the input to the model is fixed, we select the first 20 packets of each biflow for slicing. For packets longer than 400 bytes, the tail is truncated. For packets shorter than 400 bytes, the tail is padded with 0x00 to achieve the desired length. In addition, because this model handles the headers and payloads differently, the headers and payloads are separated to form a sequence, respectively. A biflow consists of a header flow and a payload flow, which can be expressed as follows: $\{(p\_\text{payload}_1, p\_\text{payload}_2, ..., p\_\text{payload}_n), (p\_\text{head}_1, p\_\text{head}_2, ..., p\_\text{head}_n)\}$. The first 20 data packets of each biflow are selected for slicing processing; the data packets with a length greater than 400 are truncated from the tail, and the data packets with a length of less than 400 are filled with 0X00 at the end.

(4) Traffic encoding: the payload is converted to a hexadecimal numeric representation. Each packet header is represented by a corresponding four-tuple. The biflow can be expressed as follows: $\{(d\_\text{payload}_1, d\_\text{payload}_2, ..., d\_\text{payload}_n), (t\_\text{head}_1, t\_\text{head}_2, ..., t\_\text{head}_n)\}$.

3.4. Representation Learning Stage

3.4.1. Obtaining Payload-Level Biflow Representation. The vast majority of data in encrypted traffic is random bytes with no real meaning. However, the handshake packets, headers of application packets, etc. in the traffic are still transmitted in plaintext and contain a lot of valid information that can be used to categorize malicious flows [29]. We extracted plaintext features for malicious encrypted

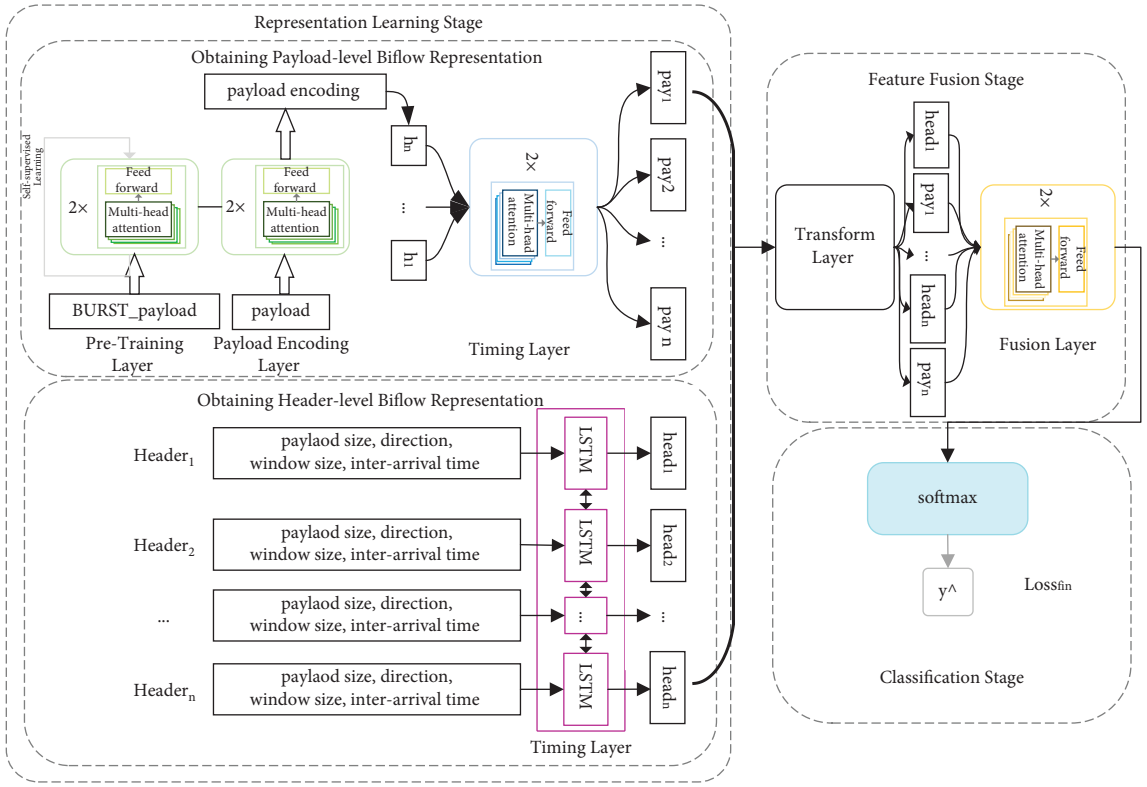


Figure 1: Overall framework of the HMMED model.

traffic detection and obtained good results. However, on the one hand, extracting plaintext features requires prior knowledge, and on the other hand, the plaintext features are often not discrete but correlated. For example, the Cipher Specs field of Client Hello and the Cipher Suite field of Server Hello in the TLS protocol are in the relationship of option and selection [30]. Moreover, this relationship is diverse and dynamic, and it is difficult to design suitable features to represent it. Therefore, in this paper, deep learning models with representational learning capabilities and the ability to handle nonindependent random variables are chosen to mine features. In addition, considering that the IP header and TCP header contain biased data such as IP address and port, which seriously affects the generalization ability of the model, we retain only the transport layer payload and learn the content features from the payload. Interaction behavior between protocol fields occurs not only between payloads but also within the payload. Therefore, we designed a three-layer structure to gradually learn the interaction behavior patterns within and between payloads.

(1) Pretraining layer. The purpose of the pretraining layer is to learn the interaction patterns between bytes. Since traditional deep learning is highly dependent on the training set size and data distribution, resulting in a less generalized model that is difficult to adapt to unseen encrypted traffic; we chose to use a large amount of unlabeled encrypted traffic data in the pretraining layer to pretrain a model that can capture the generic relationship of bytes.

The pre-training layer consists of two phases. In the embedding phase, the inputs to the pretrained model are obtained through the byte embedding generation method. In the training phase, two training tasks, Burst_payload homology prediction and Burst_payload filling, are designed to learn the interaction patterns within the payload from the transition context instead of the semantic context. BURST is a concept introduced in the literature [15], which is defined as a consecutive set of packets from a server to a client or from a client to a server. Burst_payload is a continuous set of packet payload from a server to a client or from a client to a server. In this paper, instead of using all the data in BURST, only the payload part of a set of packets in BURST is used.

(1) Embedding phase

The byte embedding generation method uses three types of embeddings to generate byte embedding, namely token embedding, position embedding, and direction embedding. The byte embedding generation process is shown in Figure 2.

Token embedding: we implement token embedding by one-hot encoding. One token is two adjacent bytes in the payload. Four tokens, [CLS], [PAD], [SEP], and [MASK], are added. [CLS] indicates the beginning of the sequence and is inserted at the top of the BURST_payload. [SEP] is used to separate payloads and is inserted at the end of a payload to indicate the end of the payload and the beginning of another. If there is only one payload, there is no need

to insert [SEP]. [PAD] serves as padding. The maximum length of BURST_payload is specified to be 512, i.e., the maximum number of tokens is 512, and if it is not enough, it is padded with [PAD]. [MASK] is used to mask the payload bytes and is used in the BURST_payload filling task.

Positional embedding: the direction embedding indicates the direction of the data packet. 0 represents the data packet from the server to the client, and 1 represents the data packet from the client to the server.

Direction embedding: the positional embedding represents the positional information of tokens and uses a sinusoidal position encoding to represent the positional information of each token.

(2) Training phase

Transformer is a model that is based entirely on the self-attention mechanism proposed in 2017 [31], which has two major advantages of parallel computing and the shortest and maximum path length. The transformer model consists of an encoder and a decoder. Within the encoder, there are multiple layers, all of which are identical. Each layer is comprised of two sublayers: a multi-head self-attention mechanism and a feed-forward neural network. The decoder, like the encoder, is stacked with several identical layers. However, in contrast to the encoder, the decoder includes an additional sublayer known as the encoder-decoder attention layer, which is inserted between the two existing sublayers. This paper does not use the decoder part, only the encoder. The multihead attention mechanism is the most important component of the transformer, which uses the query generated by the input, the key, and the value to calculate the attention score of each input. Multihead refers to the parallel computing of different self-attention mechanisms to learn the dependencies between inputs from multiple perspectives. The calculation process of the multi-head attention mechanism is as follows:

$$\text{Attention}(Q, K, V) = \text{soft max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

$$\text{head}_i = \text{Attention}\left(QW_i^Q, K_i^K, VW_i^V\right), 1 \le i \le h, \tag{3}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right)W^O. \tag{4}$$

d_k is the dimension of k;W^Q, W^K, W^V, and W^O are different linear transformation matrices.

This paper uses the transformer as the pretraining model. The multihead attention mechanism enables the model to learn the transition context relationship between bytes and the dependencies between the same directional payloads by two training tasks. The two training tasks are the Burst_payload filling task and the Burst_payload homology prediction task. The details are as follows:

Burst_payload filling task (BFT): BFT aims to learn the dependencies between bytes in the payload. The input of the model is a byte embedding sequence. The BFT randomly masks 15% of tokens in the input sequence. Then, these masked tokens are used as supervisory signals, and the model recovers the masked tokens through contextual information. The model structure is shown in Figure 3(a). We measure the training error using cross-entropy. Assuming k token embeddings are randomly masked, this is the formula of the loss function:

$$\text{Loss}_{\text{BFT}} = -\sum_{i=1}^{k} \log\left(p\left(\text{mask}_i = o_i \mid \widetilde{X}; \theta\right)\right). \tag{5}$$

θ represents all parameters of the model; $\widetilde{X}$ represents the sequence input to the model after masking.

Burst_payload homologous prediction task (BSP): BSP aims to learn the dependencies between the same directional payloads. The BSP randomly selects a payload and replaces it with a probability of 0.5. The task is to predict whether a replacement event has occurred for Burst_payload, which is a binary classification task. The model structure is shown in Figure 3(b). The model uses cross-entropy to measure the prediction error. The following is the formula of the loss function:

$$\text{Loss}_{\text{BSP}} = -\log\left(p\left(y \mid \widehat{X}; \theta\right)\right). \tag{6}$$

θ represents all parameters of the model; $\widehat{X}$ represents the sequence after replacing.

The final loss of the pretrained model is the sum of the losses of the two tasks, $\text{Loss}_0 = \text{Loss}_{\text{BFT}} + \text{Loss}_{\text{BSP}}$. It should be emphasized that the two training tasks train an encoder. The addition operation can make Loss_0 the smallest when Loss_{BFT} and Loss_{BFP} both are as small as possible, which ensures that both tasks can achieve better performance.

Algorithm 1 shows the training process of the pretrained model: a pseudocode.

(2) Payload encoding layer. The payload encoding layer is a transition layer whose purpose is to obtain the packet payload encoding for use in the timing layer. The payload

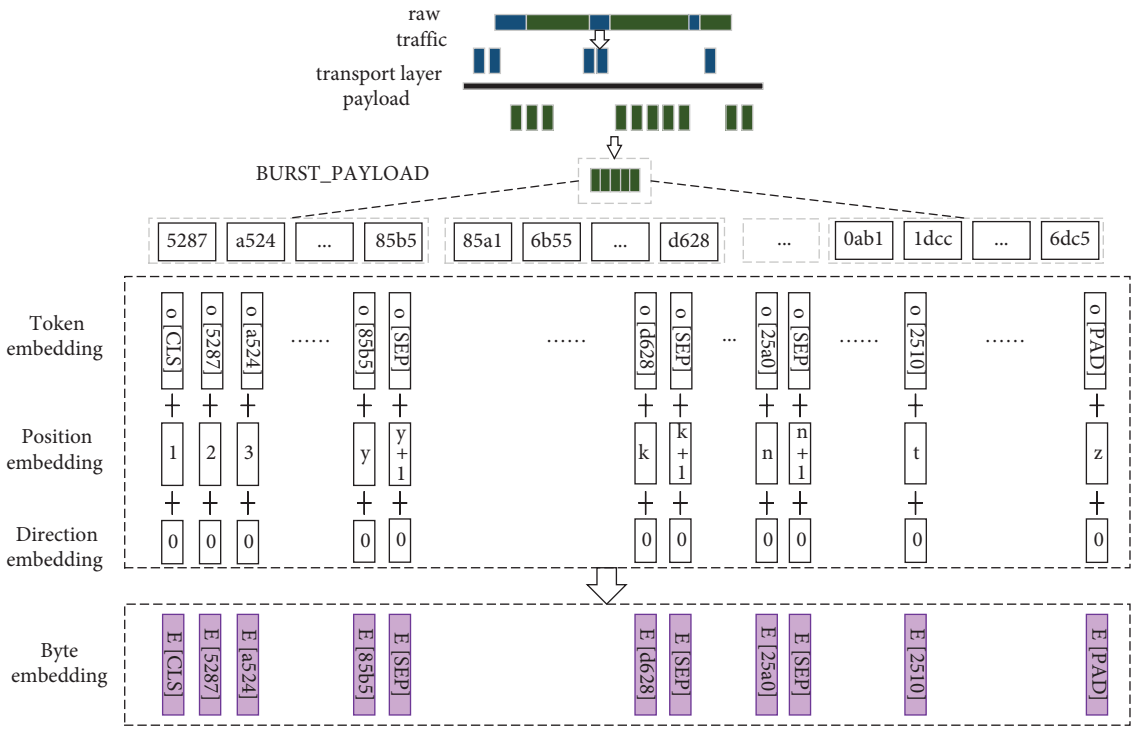


Figure 2: Byte embedding generation.

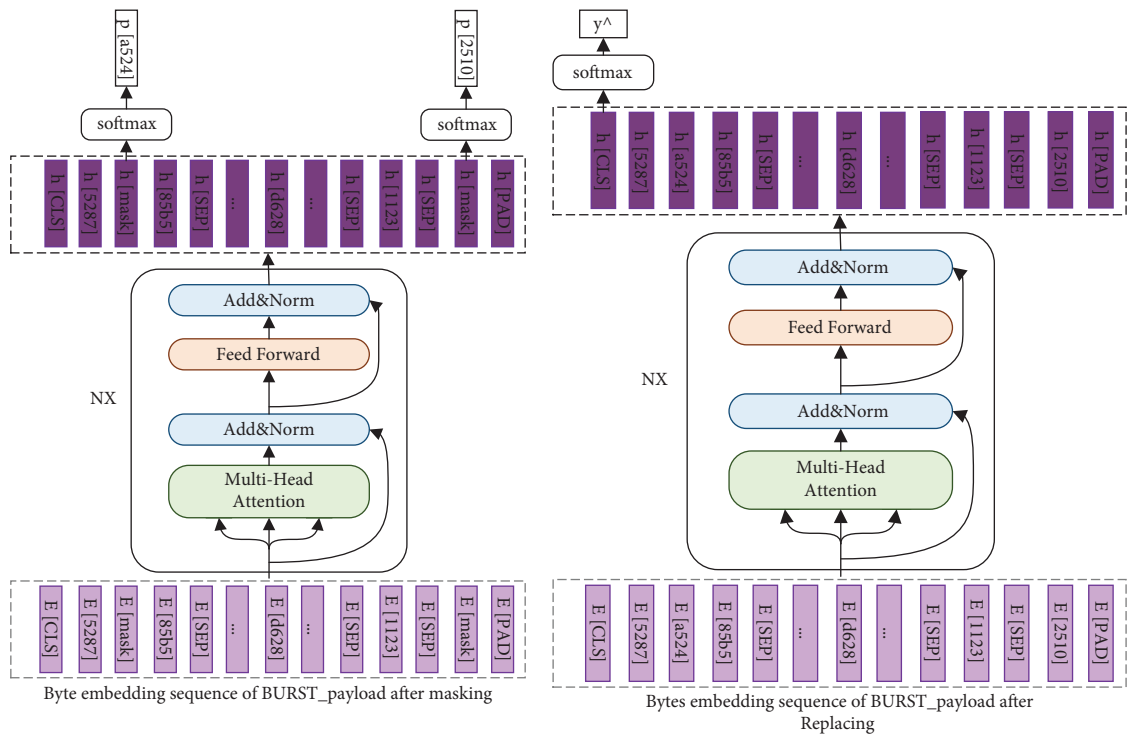


Figure 3: Pretraining layer model. (a) BFT model; (b) BSP model.

encoding layer reuses the model from the pretraining layer to obtain payload encoding with byte interaction information. The payload encoding layer still uses the byte embedding generation method in the pretraining layer to obtain the standard model inputs and reuses the encoder from the pretraining layer. The encoder output corresponding to

INPUT: raw traffic raw_input, transformer layers num_layers, number of attention mechanism heads num_heads, output dimension hidden_size, learning rate learning_rate, batch size batch_size, epoch epochs
OUTPUT: model parameter θ

(01) According to five-tuple and the definition of the Burst_payload, extract samples Burst_payload from raw_input;
(02) According to the byte embedding generation method, transform the byte sequence of Burst_payload into the vector sequence LS = $[E_1, E_2, ..., E_n]$;
(03) **for** $i = 0$ to epochs **do**:
(04) **for** each Burst_payload in raw_input **do**:
(05) mask 15% of tokens and the masked sample is LS_MASK;
(06) X = LS_MASK; # LS_MASK is input into the model
(07) $K_h = X * W_h{}^K$, $V_h = X * W_h{}^V$, $Q_h = X * W_h^Q$; # $h = (1, ..., \text{num\_heads})$
(08) $\text{head}_h = \text{soft max}(Q_h K_h{}^T / \sqrt{d_k}) V_h$;
(09) $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_{\text{num\_heads}}) W^O$;
(10) $\widetilde{X} = \text{LayerNorm}(X + \text{MultiHead}(Q, K, V))$;
(11) $\text{FFN}(\widetilde{X}) = \max(0, \widetilde{X} * W_1{}^{\text{FFN}} + b_1^{\text{FFN}}) W_2^{\text{FFN}} + b_2^{\text{FFN}}$;
(12) $\text{OUTPUT} = \text{LayerNorm}(\widetilde{X} + \text{FFN}(\widetilde{X}))$;
(13) extract the corresponding output of the masked tokens from OUTPUT to form the sequence OUTPUT_MASK_LS;
(14) $P = \text{soft max}(\text{OUTPUT\_MASK\_LS})$; #predict
(15) $\text{Loss}_{\text{BFT}} = -\sum_{i=1}^{k} \log(p(\text{mask}_i = o_i \mid \widetilde{X}; \theta))$; #loss of filling tasks
(16) randomly select a payload in LS and replace it with a probability of 0.5. The replaced sample is LS_REPLACE;
(17) X = LS_REPLACE; # LS_REPLACE is input into the model
(18) execute step 07–12;
(19) extract the output OUTPUT_CLS_VECTOR corresponding to [CLS] from OUTPUT;
(20) $P = \text{soft max}(\text{OUTPUT\_CLS\_VECTOR})$; #predict
(21) $\text{Loss}_{\text{BSP}} = -\log(p(y \mid \widehat{X}; \theta))$; # loss of homologous predict tasks
(22) $\text{Loss}_0 = \text{Loss}_{\text{BFT}} + \text{Loss}_{\mathbf{BSP}}$; # add the two losses
(23) calculate the gradient of parameter W and update the parameters θ;
(24) end for
(25) end for
(26) return θ

Algorithm 1: The training process of the pretrained model.

E[CLS] is the payload encoding. In addition, to further obtain the packet payload encoding for the encrypted traffic detection task, this paper feeds the encoder output corresponding to E[CLS] to the downstream task, which will further fine-tune the encoder. The model structure is shown in Figure 4.

(3) Timing layer. The purpose of the timing layer is to learn the interaction behavior patterns between payloads to obtain payload-level biflow representation. The payload-level biflow representation with byte interaction information and payload interaction information portrays the traffic comprehensively. Considering the advantage of transformer's parallelizable operation, we still use transformer's encoder in this layer to learn the sequential information between payloads. The payload encoding sequence of a biflow is input to the model, and the length of the sequence is fixed to 20. If the length is less than 20, the sequence is padded to 20. The model structure is shown in Figure 5.

3.4.2. Obtaining Header-Level Biflow Representation. As mentioned above, there are many biased data in the packet header, such as the port of the transport layer and the IP address of the IP layer [32]. If they are directly input into the model, the model will have problems with performance expansion and lack of generalization, which will lead to poor performance when classifying unfamiliar biflows. In order to avoid the use of biased data and improve the speed of model training, this paper uses four-tuple (<load size, window size, interarrival time, and data packet direction>) to represent the data packet header.

The reason for selecting the payload size and packet direction is that the payload size distribution of malicious traffic is often different from that of normal traffic and the payload size distribution is related to the packet direction. For example, when browsing the web, the client's request packet to the server is usually short, while the server's response packet to the client is very long. But malware is completely the opposite. The server only sends a small amount of control commands to the client, and the client sends a massive amount of data packets to the server for data return. There is also a large difference in the distribution of time intervals between malicious and normal traffic. According to [30], the time interval of about 80% of malicious traffic is within 0.1 s, while only about 25% of benign traffic is within 0.1 s. Therefore, we choose these four side channel features to express the formal information of traffic as a supplement to the traffic content features.

In this paper, the BiLSTM model is used to learn the sequential information in four-tuples and obtain the header-level biflow representation. The input data format of the

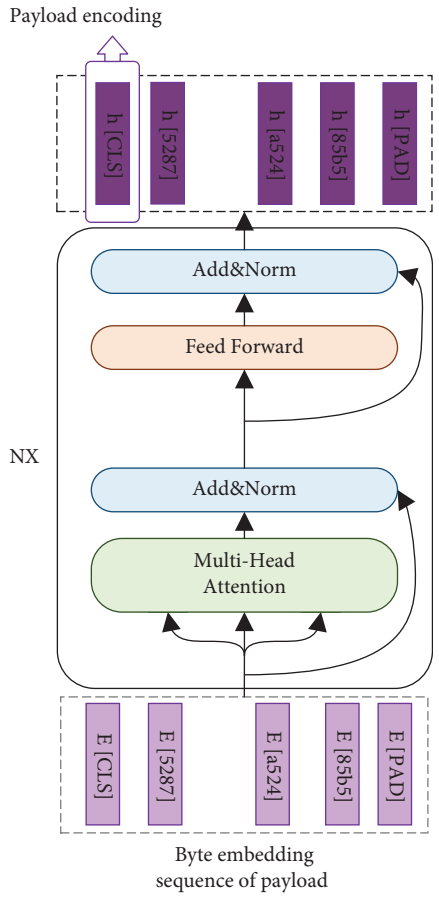


Figure 4: Payload encoding model.

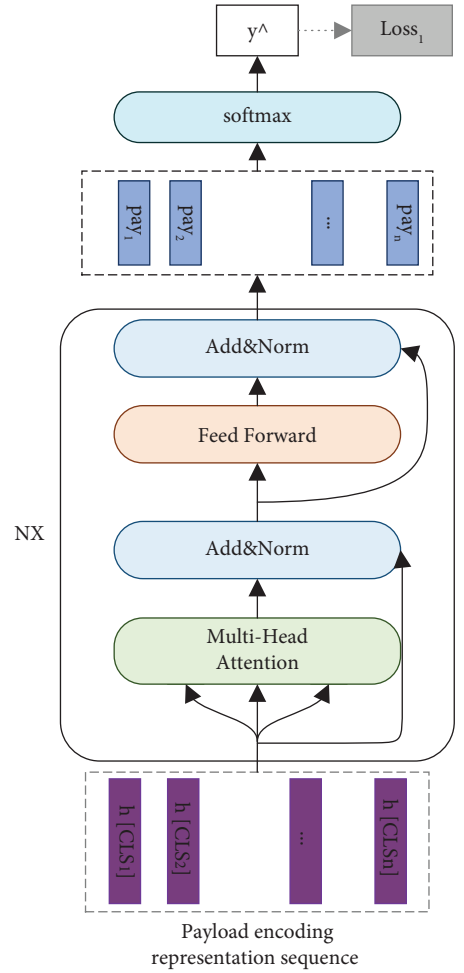


Figure 5: Timing layer model.

BiLSTM model is shown in Figure 6. Among them, t_head_i is the header vector. The elements of the vector are f_1, f_2, f_3, and f_4, which, respectively, refer to the payload size, window size, interarrival time, and packet direction. $\{t\_\text{head}_1, t\_\text{head}_2, \ldots, t\_\text{head}_n\}$ is the input sequence of the model.

The BiLSTM model outputs the header-level biflow representation $H_{\text{header}} = \{\text{head}_1, \text{head}_2, \ldots, \text{head}_n\}$.

$$\left\{\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_n}\right\} = \overrightarrow{\text{LSTM}}\left(\{t\_\text{head}_1, t\_\text{head}_2, \ldots, t\_\text{head}_n\}\right), \tag{7}$$

$$\left\{\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_n}\right\} = \overleftarrow{\text{LSTM}}\left(\{t\_\text{head}_1, t\_\text{head}_2, \ldots, t\_\text{head}_n\}\right), \tag{8}$$

$$\text{head}_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i}. \tag{9}$$

The symbol $\oplus$ indicates splicing. The dimension h_i of hi is 94.

The model for obtaining the header-level biflow representation is shown in Figure 7.

3.5. Feature Fusion Stage. In the representation learning stage, we get the payload-level biflow representation $\{\text{pay}_1, \text{pay}_2, \ldots, \text{pay}_n\}$ and the header-level biflow representation $\{\text{head}_1, \text{head}_2, \ldots, \text{head}_n\}$. The feature fusion stage consists of the transform layer and the fusion layer. In the transform layer, payload-level biflow representation and header-level biflow representation are combined so that the payload representation and header representation of the same packet are physically adjacent to each other. At the fusion layer, the transformer model is utilized to learn the correlation between payload representation and header representation. The model outputs the fused traffic representation H.

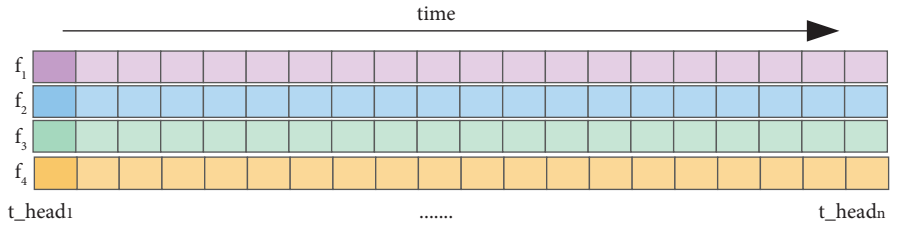


Figure 6: BiLSTM model input data format.

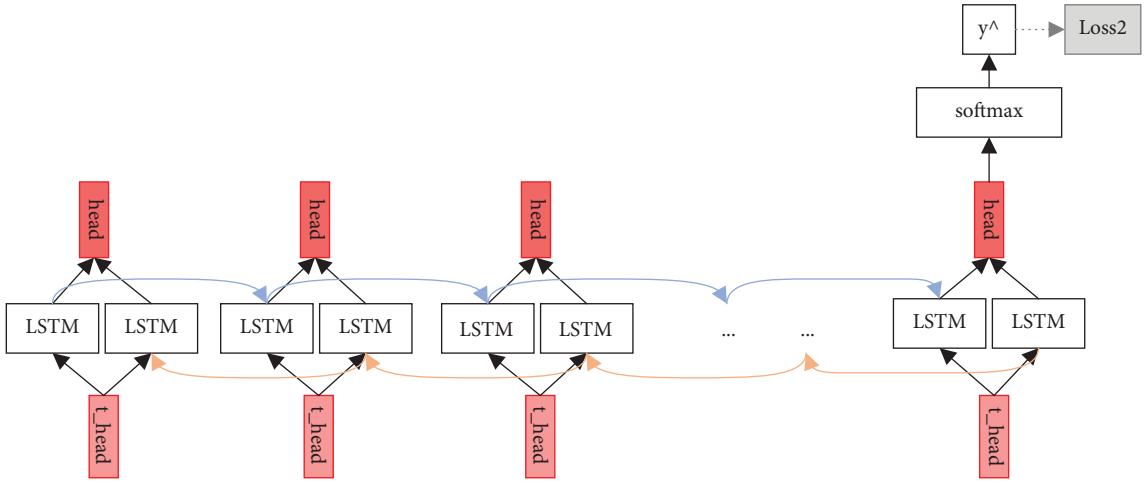


Figure 7: The model of obtaining the header-level biflow representation.

3.5.1. Transform Layer. First, the transform layer decapsulates the payload-level biflow representation and header-level biflow representation. Then, the two representations are rearranged and combined. In this paper, in order to identify whether the representation comes from the header or the payload, a mark is added to each representation. [1, 0] indicates the representation from the header, and [0, 1] indicates the representation from the payload. In addition, we add an all-zero vector #CLS# in front of the biflow representation, indicating the start of the biflow. The biflow representation is expressed as $\text{Bi\_flow} = (\#\text{CLS}\#, \text{head}_1, \text{pay}_1, \text{head}_2, \text{pay}_2, \ldots, \text{head}_n, \text{pay}_n)$, and the dimension is 96, $\text{Bi\_flow}_{\text{dim}} = 96$. The transform process is shown in Figure 8.

3.5.2. Fusion Layer. After obtaining the biflow representation, we input it into the encoder of the transformer for feature fusion. The encoder outputs the fused biflow representation H. The process for calculating H is as follows:

$$\widetilde{\text{Bi\_flow}} = \text{LayerNorm}(\text{Bi\_flow} + \text{MultiHead}(Q, K, V)), \tag{10}$$

$$H = \text{LayerNorm}(\widetilde{\text{Bi\_flow}} + \text{Forward}(\widetilde{\text{Bi\_flow}})), \tag{11}$$

where Q, K, and V represents the query, key, and value in the self-attention mechanism. $Q = \text{Bi\_flow} * W_Q$, $K = \text{Bi\_flow} * W_k$, *and* $V = \text{Bi\_flow} * W_V$.

3.6. Classification Stage. During the classification stage, the softmax classifier takes the fused biflow representation H as input and produces the ultimate classification outcome. In order to ensure that HMMED can learn the optimal header-level biflow representation and payload-level biflow representation, when training the model, we first calculate the losses (Loss_1 and Loss_2) of the header timing layer and the loader timing layer, respectively, and then calculate the classification loss (Loss_3) after feature fusion. Finally, the HMMED is trained with the sum of the three loss values, $\text{Loss}_{\text{fin}} = \text{Loss}_1 + \text{Loss}_2 + \text{Loss}_3$.

4. Experiment and Analysis

4.1. Experimental Environment and the Dataset

4.1.1. Experimental Environment and Parameter Settings. This article is an experimental operation on the Linux 64 bit operating system. The processors are an Intel Xeon E5-2698

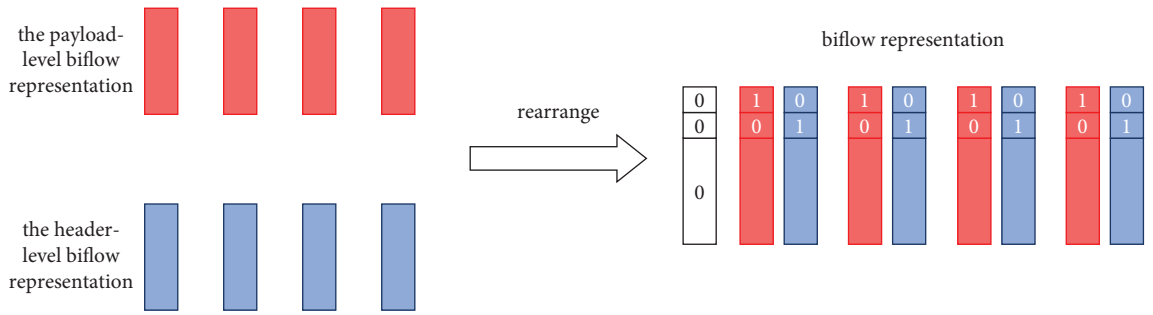


Figure 8: The transform processing.

and an NVIDIA GeForce RTX3080 with 128 GB of memory. The experimental environment language is Python, and the deep learning framework is TensorFlow. There are mainly NumPy, Pandas, Keras, Sklearn, Pydot, Seaborn, Matplotlib, and other related libraries.

The parameters of the model are set as follows: In the pretraining layer, this work uses a 2-layer 4-head transformer (i.e., stacking 2 transformer encoders, each encoder uses a 4-head self-attention mechanism). In the timing layer, this work is also implemented using a 2-layer, 4-head transformer. The timing layer of the second mode uses the BiLSTM model, and the dimension of the hidden state is 96. In addition, in the training process, the model uses the cross-entropy loss function to adjust the model parameters and uses the Adam optimizer, which can not only adapt to the sparse gradient but also alleviate the problem of gradient oscillation. The initial learning rate is set to 0.01, the batch size is set to 64, the number of iterations is 20, and all the activation functions in the model are ReLU. In this paper, the training set and the test set are set according to the ratio of 7 : 3.

4.1.2. Dataset Introduction. The dataset used in this paper is the USTC-TFC2016 dataset designed by the literature [33]. The dataset includes 10 kinds of malicious encrypted traffic and 10 kinds of benign traffic. The sample datasets are all in PCAP file format, and the size is 3.71 GB. Table 1 shows the number of samples of different types of data after preprocessing. Among them, the left side shows 10 kinds of malicious encrypted traffic, which is selected from the malicious traffic collected by CTU researchers in the real network environment from 2011 to 2015. On the right are 10 kinds of normal traffic, generated by IXIA BPS simulation equipment.

4.2. Evaluation Criteria. The assessment of the experimental results employs several evaluation indicators, namely the precision rate (precision), the accuracy rate (accuracy), the F1 value (F1_score), and the recall rate (recall). Among these indicators, TP indicates the count of samples accurately identified as the target traffic, FP represents the count of samples incorrectly identified as the target traffic, FN signifies the count of samples not correctly identified as the target traffic, and TN denotes the count of samples correctly identified as nontarget traffic.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{12}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{13}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{14}$$

$$\text{F1}_{\text{score}} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{15}$$

This paper uses the receiver operating characteristic curve (ROC) [34] and evaluates it with the area under the ROC curve (AUC) to examine the model's capability to identify samples at a certain threshold. The larger the AUC is, the better the diagnostic value of the model is, where M is the number of positive samples and N is the number of negative samples. Finally, the confusion matrix is used to summarize the prediction effect of the classification model in this paper.

$$\text{AUC} = \frac{\sum_{k\,\text{posthiveClass}} \text{rank}_i - M(1+M)/2}{M \times N}. \tag{16}$$

4.3. Experimental Results and Analysis

4.3.1. Comparison with Unimodal Models. In order to explore the detection effect of the HMMED model, this paper compares the detection results of the single-modal model with the hierarchical multimodal model proposed in this paper. The model that only uses the payload in the packet as input is called the HMMED-1 model, and the model that only uses the packet header as input is called the HMMED-2 model. The accuracy, F1 score, and recall rate of the model are analyzed, respectively. Table 2 shows the comparison between the model in this paper and other single-modal models in accuracy, F1 score, and recall rate under the same dataset. It can be seen that the model HMMED is superior to other single-modal models in all aspects.

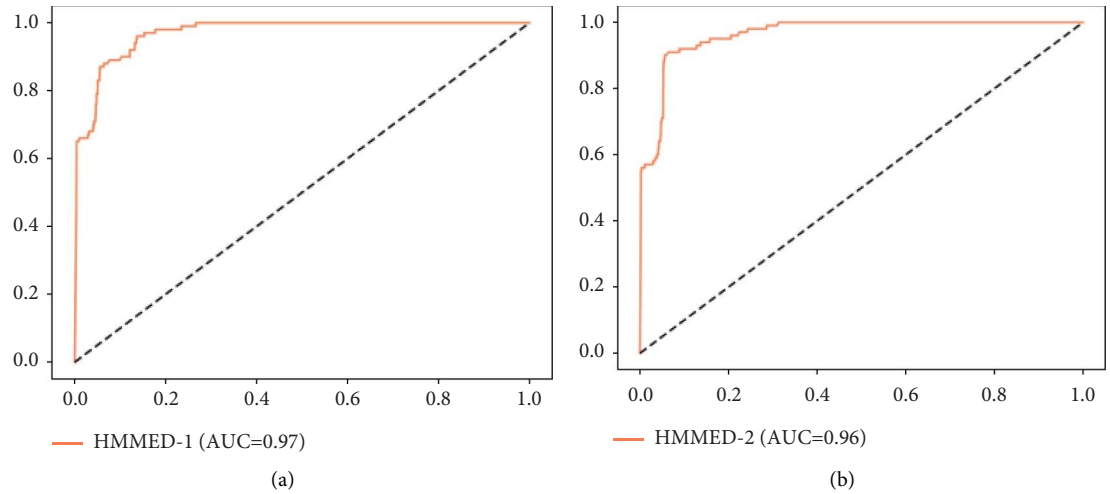
Table 2 shows that the multimodal model HMMED is higher than the single-modal model in both accuracy and F1 value. This is due to the fact that the multimodal model integrates payload content features and side-channel features that describe traffic form, which makes up for the problem of incomplete traffic description in a single mode. In addition, in order to verify the performance of the proposed model in encrypted traffic classification, the AUC

Table 1: Dataset introduction.

Classification	Traffic type	Number of samples
Malicious encrypted traffic	Cridex	8095
	Geodo	6691
	Htbot	5952
	Miuref	4756
	Neris	7653
	Nsis-ay	6014
	Shifu	9576
	Tinba	8304
	Virut	6034
	Zeus	5661
Normal network traffic	BitTorrent	7462
	Facetime	5089
	FTP	6119
	Gmail	5101
	MySQL	6886
	Outlook	7324
	Skype	6059
	SMB	5312
	Weibo	4357
	World of Warcraft	7368

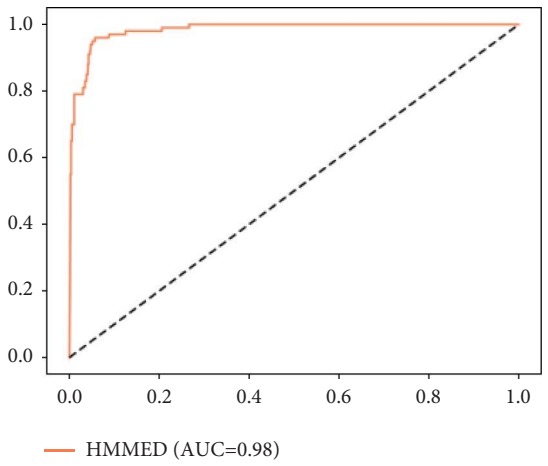
Table 2: Performance comparison with single-mode models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1_score (%)
HMMED-1	97.52	96.57	98.32	97.43
HMMED-2	96.63	95.67	96.54	99.93
HMMED	99.24	98.89	99.36	99.12



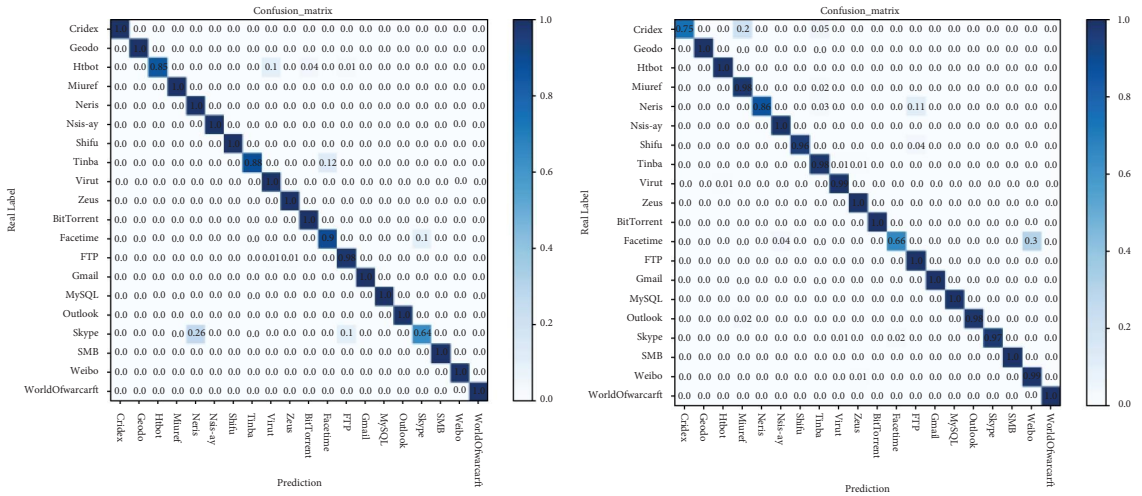
(a) (b)

Figure 9: Continued.

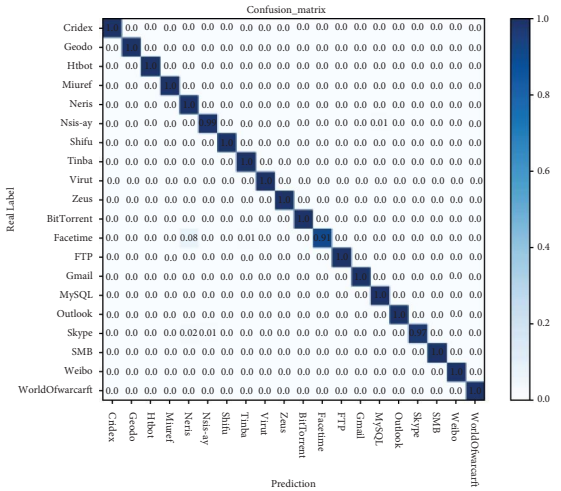


(c)

Figure 9: ROC curves. (a) The ROC curve of HMMED-1; (b) the ROC curve of HMMED-2; (c) the ROC curve of HMMED.



(a) (b)



(c)

Figure 10: Confusion matrices. (a) The confusion matrices of HMMED-1; (b) the confusion matrices of HMMED-2; (c) the confusion matrices of HMMED.

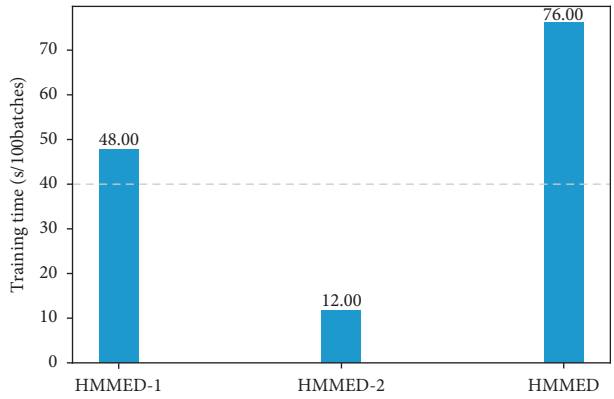


Figure 11: Model training time comparison.

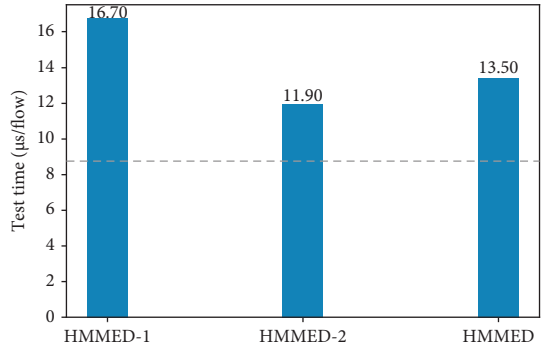


Figure 12: Model test time comparison.

scores of the above three models are calculated, respectively. The AUC score of the model (HMMED) in this paper is as high as 98.85%, which shows the superiority of HMMED. Figures 9 and 10 are the ROC curves and confusion matrices of the three models, respectively.

Figures 11 and 12 show the training time and test time of each model, respectively. It can be seen from the diagram that the training phase of HMMED consumes more time than other models, which is due to the pretraining. However, once the model is trained well, the test time of HMMED is also very ideal.

4.3.2. Comparison with Advanced Algorithms. To further validate the superiority of the model, the encrypted traffic classification results of this model and other models in recent years under the same dataset (USTC-TFC2016) are compared, as shown in Table 3.

(i) LeNet-5 [33] adopts an end-to-end detection method, which converts traffic into a $78 \times 78 \times 1$ grayscale image and uses the CNN to classify the grayscale image.

(ii) The LSTM + Word embedding [35] splits the data packet header into 33 fields, uses the word embedding mechanism to embed fields, and then uses LSTM to further learn the sequential information between fields.

(iii) ET-BERT [15] also adopts an end-to-end detection method. It uses pretraining ideas to achieve malicious traffic detection. First, a transformer model that can learn general datagram-level representation is pretrained, and then the pretrained model is fine-tuned according to specific downstream tasks to achieve specific classification tasks.

According to Table 3, the model proposed in this paper shows significantly high performance in the combined comparison of accuracy, precision, recall, and F1_score. The performance of the model proposed by [35] is second. On the one hand, this is because the model does not perform anonymous operations, and biased data such as MAC, IP, and port are exposed to the model, resulting in performance inflation. On the other hand, it also shows that the packet header fields contain effective information that can classify malicious flows. Both ET-BERT and LeNet-5 are end-to-end models that learn features directly from raw data. However, ET-BERT outperforms LeNet-5, suggesting that the transformer model for pretraining is more capable of learning valid information than the simple CNN model. HMMED uses the pretraining idea, uses the pretrained model in a well-designed downstream model structure, and supplements the flow information learned from the second modality, which gives HMMED an excellent flow representation capability. HMMED exhibits optimal performance.

Table 3: Comparison of detection results of different detection models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1_score (%)
LeNet-5 [33]	99.17	~	~	~
LSTM + word embedding [35]	99.88	99.99	99.86	99.93
ET-BERT [15]	99.29	99.30	99.30	99.30
HMMED	99.98	99.94	99.96	99.95

5. Conclusion and Prospect

In this paper, a hierarchical multimodal malicious encrypted traffic detection framework (HMMED) is proposed. The data of the first mode come from the packet load. For the data of the first mode, we use the end-to-end method to learn the general packet payload representation through the pretraining model and then use the transformer's encoder to learn the payload timing features. The data of the second mode come from the packet head. For the data of the second mode, we use the feature engineering method to process and use the four-tuple to represent the packet head. BiLSTM is used to learn the header timing features. After that, the two data features are fused, and the fused hidden features are used to complete the detection of malicious encrypted traffic. According to the experimental findings, the performance of the proposed model surpasses both the single-mode model and the advanced algorithm. The accuracy of this model is as high as 99.98% in the encrypted traffic detection task. In the future, we will try to increase the number of modes and learn more comprehensive hidden features of encrypted traffic from more modes to improve the accuracy of classification.

Although the method proposed in this paper has improved the performance of malicious encrypted traffic detection, there are still several aspects that can be optimized.

(1) Robustness of adversarial attacks: considering that malicious actors may adopt adversarial strategies to avoid detection, and future work can study how to enhance the robustness of the model against these attacks.

(2) Continuous learning and adaptation: explore the online learning or continuous learning mechanism of the model so that it can adapt to the rapid evolution of network threats.

(3) Model interpretability: improving the interpretability of the model is crucial for network security analysts. Future research can explore how to make the decision-making process of the HMMED model more transparent to help analysts understand the behavior of the model and build trust.

Data Availability

The dataset used to support the findings of this paper is publicly available on the Internet: USTC-TFC2016.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by "Research and Application of Key Technologies for CloudEdge Collaborative Security in New Power Systems Based on Zero Trust Architecture (Subject 4), grant no. YNKJXM20210198."

References

[1] "HTTPS encryption on the web – Google transparency Report," https://transparencyreport.google.com.

[2] Internet Security Report, in *WatchGuard Technologies*, https://www.watchguard.com/wgrd-resource-center/security-report-q2-2021.

[3] S. Dong, Y. Xia, and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4197–4212, 2021.

[4] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[5] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proceedings of the Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, pp. 271–282, 2014.

[6] Y. Xia, S. Dong, T. Peng, and T. Wang, "Wireless network abnormal traffic detection method based on deep transfer reinforcement learning," in *Proceedings of the 2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pp. 528–535, 2021.

[7] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: a survey," *Sustainable Cities and Society*, vol. 60, 2020.

[8] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and bot-IoT attacks traffic identification for Internet of things in smart city," *Future Generation Computer Systems*, vol. 107, pp. 433–442, 2020.

[9] Z. Wang, K. W. Fok, and V. L. L. Thing, "Machine learning for encrypted malicious traffic detection: approaches, datasets and comparative study," *Computers and Security*, vol. 113, 2022.

[10] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[11] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Computers and Security*, vol. 94, 2020.

[12] 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*, vol. 6, pp. 1792–1806, 2018.

[13] W. Bao, L. Sha, and X. Cao, "Malicious encrypted traffic identification based on feature fusion," *Computer Systems and Applications*, vol. 32, pp. 358–367, 2023.

[14] J. Cheng, R. He, E. Yuepeng, Y. Wu, J. You, and T. Li, "Real-time encrypted traffic classification via lightweight neural networks," in *Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[15] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Et-Bert, "A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the Proceedings of the ACM Web Conference*, pp. 633–642, 2022.

[16] Q. Xia, S. Dong, and T. Peng, "An abnormal traffic detection method for IoT devices based on federated learning and depthwise separable convolutional neural networks," in *Proceedings of the 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 352–359, 2022.

[17] P. Lin, K. Ye, Y. Hu, Y. Lin, and C.-Z. Xu, "A novel multimodal deep learning framework for encrypted traffic classification," *IEEE/ACM Transactions on Networking*, vol. 31, no. 3, pp. 1369–1384, 2023.

[18] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*, O'Reilly Media, Inc, 2018.

[19] J. Hou, F. Liu, H. Lu, Z. Tan, X. Zhuang, and Z. Tian, "A novel flow-vector generation approach for malicious traffic detection," *Journal of Parallel and Distributed Computing*, vol. 169, pp. 72–86, 2022.

[20] Z. Wang and V. L. L. Thing, "Feature mining for encrypted malicious traffic detection with deep learning and other machine learning algorithms," *Computers and Security*, vol. 128, 2023.

[21] R. Zheng, J. Liu, W. Niu, L. Liu, K. Li, and S. Liao, "Preprocessing method for encrypted traffic based on semi-supervised clustering," *Security and Communication Networks*, vol. 2020, pp. 1–13, 2020.

[22] J. Liu, Z. Tian, R. Zheng, and L. Liu, "A distance-based method for building an encrypted malware traffic identification framework," *IEEE Access*, vol. 7, pp. 100014–100028, 2019.

[23] S. Dong, "Multi class SVM algorithm with active learning for network traffic classification," *Expert Systems with Applications*, vol. 176, 2021.

[24] A. Lichy, O. Bader, R. Dubin, A. Dvir, and C. Hajaj, "When a RF beats a CNN and GRU, together—a comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification," *Computers and Security*, vol. 124, 2023.

[25] Z. Khodaverdian, H. Sadr, and S. A. Edalatpanah, "A shallow deep neural network for selection of migration candidate virtual machines to reduce energy consumption," in *Proceedings of the 2021 7th International Conference on Web Research (ICWR)*, pp. 191–196, IEEE, Tehran, Iran, 2021.

[26] Z. Khodaverdian, H. Sadr, S. A. Edalatpanah, and M. Nazari, "An energy aware resource allocation based on combination of CNN and GRU for virtual machine selection," *Multimedia Tools and Applications*, vol. 83, no. 9, pp. 25769–25796, 2023.

[27] S. Soleymanpour, H. Sadr, and M. Nazari Soleimandarabi, "CSCNN: cost-sensitive convolutional neural network for encrypted traffic classification," *Neural Processing Letters*, vol. 53, no. 5, pp. 3497–3523, 2021.

[28] S. Soleymanpour, H. Sadr, and H. Beheshti, "An efficient deep learning method for encrypted traffic classification on the web," in *Proceedings of the 2020 6th International Conference on Web Research (ICWR)*, pp. 209–216, IEEE, Tehran, Iran, 2020.

[29] O. Roques, S. Maffeis, and M. Cova, "Detecting malware in TLS traffic," in *Proceedings of the the IEEE Conference on Local Computer Networks 30th Anniversary*, LCN'05, 2019.

[30] Y. Gu, H. Xu, and X. Zhang, "Multi-granularity representation learning for encrypted malicious traffic detection," *Chinese Journal of Computers*, vol. 46, pp. 1888–1899, 2023.

[31] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[32] J. Dai, X. Xu, H. Gao, X. Wang, and F. S. H. A. P. E. Xiao, "SHAPE: a simultaneous header and payload encoding model for encrypted traffic classification," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1993–2012, 2023.

[33] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, 2017.

[34] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, 2020.

[35] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Applied Sciences*, vol. 9, no. 16, p. 3414, 2019.