TLS-MHSA: An Efficient Detection Model for Encrypted Malicious Traffic based on Multi-Head Self-Attention Mechanism

JINFU CHEN, LUO SONG, SAIHUA CAI, HAODI XIE, SHANG YIN, and BILAL AHMAD, Jiangsu University, China

In recent years, the use of TLS (Transport Layer Security) protocol to protect communication information has become increasingly popular as users are more aware of network security. However, hackers have also exploited the salient features of the TLS protocol to carry out covert malicious attacks, which threaten the security of network space. Currently, the commonly used traffic detection methods are not always reliable when applied to the problem of encrypted malicious traffic detection due to their limitations. The most significant problem is that these methods do not focus on the key features of encrypted traffic. To address this problem, this study proposes an efficient detection model for encrypted malicious traffic based on transport layer security protocol and a multi-head self-attention mechanism called TLS-MHSA. Firstly, we extract the features of TLS traffic during pre-processing and perform traffic statistics to filter redundant features. Then, we use a multi-head self-attention mechanism to focus on learning key features as well as generate the most important combined features to construct the detection model, thereby detecting the encrypted malicious traffic. Finally, we use a public dataset to verify the effectiveness and efficiency of the TLS-MHSA model, and the experimental results show that the proposed TLS-MHSA model has high precision, recall, F1-measure, AUC-ROC as well as higher stability than seven state-of-the-art detection models.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Artificial intelligence;**

Additional Key Words and Phrases: Intrusion detection, multi-head self-attention, encrypted traffic, deep learning

ACM Reference format:
Jinfu Chen, Luo Song, Saihua Cai, Haodi Xie, Shang Yin, and Bilal Ahmad. 2023. TLS-MHSA: An Efficient Detection Model for Encrypted Malicious Traffic based on Multi-Head Self-Attention Mechanism. *ACM Trans. Priv. Sec.* 26, 4, Article 44 (October 2023), 21 pages.
https://doi.org/10.1145/3613960

This work was partly supported by the National Natural Science Foundation of China (NSFC) (Grant nos. 62172194, 62202206 and U1836116), the Natural Science Foundation of Jiangsu Province (Grant no. BK20220515), the China Postdoctoral Science Foundation (Grant no. 2023T160275), the Leading-edge Technology Program of Jiangsu Natural Science Foundation (Grant no. BK20202001), and the Qinglan Project of Jiangsu Province.

Authors' address: J. Chen, L. Song, S. Cai (corresponding author), H. Xie, S. Yin, and B. Ahmad, School of Computer Science and Communication Engineering, Jiangsu University, 301 Xuefu Road, Zhenjiang, Jiangsu, 212013, China; emails: jinfuchen@ujs.edu.cn, 2222108063@stmail.ujs.edu.cn, caisaih@ujs.edu.cn, {2222108039, yins}@stmail.ujs.edu.cn, bilalrouf@yahoo.com.

2471-2566/2023/10-ART44 $15.00
https://doi.org/10.1145/3613960

1 INTRODUCTION

Accurately identifying malicious network traffic is an important task for cyberspace security, which facilitates intelligent network operation, management, and network security. In recent years, with the wide use of encryption technology in network applications, encrypted traffic has become a common form of network traffic. Compared with normal network traffic, encrypted traffic can protect the confidentiality and integrity of private information to a certain degree, but it also provides protection against malicious activity in the network environment. According to ISO-OSI layers standard, traffic encryption can be classified as application layer encryption, representation layer encryption, and network layer encryption [23]. Application layer encryption refers to the data being encrypted by the applications using encryption protocols, thereby transmitting the data securely at the application layer, it is also known as regular encryption. Representation layer encryption and network layer encryption encrypting the packets by the applications using the techniques such as TLS (Transport Layer Security) and SSL (Secure Sockets Layer). TLS builds on SSL v3.0 by providing (1) a more secure MAC algorithm, (2) stricter alerts, and (3) clearer definition of the "gray area" specification. Compared with SSL, TLS makes the SSL more secure as well as make the protocol specification more precise and complete. As a result, most encrypted traffic on the network in recent years has utilized the TLS protocol.

However, many malicious programs use encryption techniques such as TLS to encrypt network traffic to evade detection by firewalls and intrusion detection systems, which poses new challenges to traditional network traffic detection methods [5, 6]. The features of the Transport Layer Security (TLS) protocol, including encrypted authentication, session key generation and management, and multiple encryption and session resumption, can provide relevant information and characteristics for detecting malicious encrypted traffic. Encryption and authentication can identify the authenticity and integrity of transmitted data, while session key generation and management can detect session reuse behavior in malicious activity. Multiple encryption and session resumption can help detect forged session behavior. Based on the above statement, one can conclude that understanding and utilizing the features of the TLS protocol is essential for detecting encrypted malicious traffic.

Recent studies have proposed **encrypted traffic analysis (ETA)** methods based on machine-learning and deep learning models. These approaches have significantly improved malicious traffic detection. For example, Niu et al. [15] proposed an approach based on adaptive online analysis to accurately determine the families of malware by analyzing the encrypted, drift, and imbalanced traffic streams. An improved adaptive random forest was utilized for processing new types of malware traffic, allowing for adaptive parameter updates. It also demonstrated sensitivity to malware families with limited samples in imbalanced traffic. The experimental results showed a remarkable achievement with a 99.66% F1-score in classifying malware families. Yang et al. [26] developed a complete framework for encrypted traffic analysis called **DEV-ETA (Detection, Explanation, and Validation of Encrypted Traffic Analysis)**. The study applied the post-hoc interpretation methods to interpret detection results and used the joint distribution of support features on the dataset to validate the interpretation. They used an ensemble learning model to train two detection models which resulted in an excellent precision score of 98%. Similarly, Zheng et al. [29] proposed a **graph convolutional network (GCN)** based encrypted malicious traffic detection method GCN-ETA. It fully considered the statistical features of network traffic and the structural information between them and achieved excellent performance in the experiments with precision, AUC, and F1-measure over 98%.

It is important to mention here that all the aforementioned methods are focused on features of network traffic while ignoring the importance of the features in the network traffic. For the detection of encrypted malicious traffic, it is necessary to consider the differences in feature importance and combine several conditions. For example, for the detection of encrypted malicious traffic, the

TLS extended combination features are more important than the features of the IP layer. Therefore, it is often difficult for the methods to detect the encrypted malicious traffic once these methods do not consider the weight relationship between features. Meanwhile, the manual-based feature weight assignment is too inefficient to meet the requirements of efficient traffic detection.

In order to solve the above problems about the feature weight learning and thus further improving the detection performance on encrypted malicious traffic, based on the **TLS**, this paper proposes an efficient detection model based on a **M**ulti-**H**eaded **S**elf-**A**ttention mechanism called **TLS-MHSA**. In the TLS-MHSA, we first use the multi-head self-attention mechanism to learn the weight of different key features in the TLS traffic, thereby mapping different types of features to same low-dimensional space and cross-modeling for the features in the low-dimensional space. And then, we use a multi-head self-attention mechanism to identify highly correlated feature combinations, thereby constructing the key high-order features and using them in the model. The main contributions of this paper are as follows:

(1) We propose an efficient encrypted malicious traffic detection model called TLS-MHSA based on a multi-head autonomous attention mechanism; it detects the malicious traffic by learning the key features in TLS traffic and automatically assigns high weights to key features.
(2) We introduce a multi-head self-attention layer to the DNN model to effectively process the characteristics of encrypted traffic as well as make the proposed detection model more suitable for the application of encrypted traffic.
(3) We evaluate the proposed TLS-MHSA model using four publicly available datasets, with the experimental results showing that the TLS-MHSA model outperforms the state-of-the-art detection models in detection precision on these four publicly available datasets.

The rest of this paper is structured as follows. In Section 2, the related works about machine learning and deep learning in intrusion detection are introduced and critically discussed. In Section 3, the formulation and preliminary algorithm of TLS-MHSA is presented. In Section 4, we conducted the experiments on four publicly available datasets to evaluate the performance of the proposed method in Section 3. Finally, conclusions and future work are drawn in Section 5.

2 RELATED WORK

Traditional intrusion detection mainly uses rule-matching algorithms to detect malicious network attacks. However, the detection accuracy seriously depends on constructed rules. With the development of artificial intelligence, deep learning models are gradually applied to intrusion detection.

Before applying the deep learning in network traffic anomaly detection, scholars used machine learning methods to solve the problem of anomaly detection. For example, Anderson et al. [3] added the features of background traffic to the algorithm model and used logistic regression as well as 10-fold cross-validation to conduct the experiments, experimental results verified the proposed method achieved the detection precision of 90.3%. Anderson and McGrew [2] developed a supervised machine learning-based encrypted malicious traffic detection method using the TLS handshake metadata in encrypted traffic packets; the DNS context linked to the encrypted stream as well as other information of the features. Shekhawat et al. [18] exploited the features related to connections, SSL, and certificate log files and obtained relatively good results compared with previous works.

Compared with machine learning-based approaches, the use of deep learning algorithms allows automatic feature extraction from original network traffic without complex feature processing. Based on this advantage, some deep learning-based approaches have recently been proposed to detect encrypted malicious traffic. Amoli et al. [1] proposed a real-time unsupervised **network-based intrusion detection system (NIDS)** that uses the two independent engines without any

prior knowledge to detect new and complex attacks in both encrypted and plaintext traffic. The advantage of NIDS is that it is an unsupervised method and can detect unknown attacks, but it cannot classify different types of encrypted attack traffics.

Yu et al. [28] proposed an encrypted malicious traffic detection system based on multiple autoencoders. The system analyzed the features of encrypted protocols from the handshake phase to the authentication phase, and extended the feature vector to higher dimensions in order to extract the features of network traffic. The proposed system used a multilayer network of autoencoders to extract the features and train the classification models. The experimental results showed that this system can achieve higher detection precision and lower loss rate. However, the weight relationship between traffic features and the different combinations of encrypted traffic features are not considered in the system.

TSCRNN, proposed by Lin et al. [11], is a novel spatiotemporal feature-based identification scheme for encrypted traffic. It utilizes CNN to extract abstract spatial features and introduces stack bidirectional LSTM to learn temporal characteristics. The experimental results demonstrate the effectiveness of TSCRNN, achieving precision rates of up to 99.4% and 95.0% in Tor/non-Tor binary classification tasks and 16 classification tasks, respectively.

In summary, the deep learning based methods proposed in previous studies for malicious encrypted traffic analysis heavily rely on the domain-specific expertise provided by the humans or directly utilize the raw traffic features, and they are unable to make judgments about the importance of the features.

Indeed the most informative features will vary depending on the deep learning model applied, and some of these models are not intuitive. In addition, feature interactions are also difficult to interpret when relying on human expertise. In contrast, we use a multi-head self-attention model to automatically learn the feature weights based on TLS features and background traffic features, thereby finding the key features and accomplishing efficient malicious encrypted traffic analysis. The comparisons of the afore-mentioned encrypted traffic analysis detection methods are shown in Table 1. (In the column of Automatic rate of extraction, “Low” indicates manually selecting the features as well as using the network traffic feature dataset, “Medium” indicates manually selecting the features, and “High” represents neither.)

3 ENCRYPTED MALICIOUS TRAFFIC DETECTION METHOD BASED ON MULTI-HEAD SELF-ATTENTION MECHANISM

3.1 The Framework of TLS-MHSA

In this study, the TLS protocol information of malicious and benign encrypted traffic is statistically analyzed to provide the information for constructing the model. The framework of the proposed TLS-MHSA model is shown in Figure 1.

As is shown in Figure 1, the framework of TLS-MHSA is composed of three modules, namely data pre-processing, model training, and traffic detection. In the pre-processing data module, the original network traffic is structured and filtered to extract the better features. This step aims to provide high-quality features for model training as well as reduce the interference from irrelevant features. The extracted features are then transformed into input data for model training, ensuring that the model receives the most relevant and useful information during the training process. In the model training module, the encrypted malicious traffic detection model is constructed and pre-trained firstly based on the multi-head self-attention mechanism. Then the structure and parameters of the model are adjusted and optimized through the evaluation results to obtain the trained detection model. In the traffic detection module, the features of original network traffic under test are extracted using data pre-processing module. The extracted features serves as input to the TLS-MHSA model that was obtained from the model training module. And then, the

Table 1. The Characteristics of Encrypted Traffic Detection Methods

Methods	Automatic rate of extraction	Detection effect	Feature weight learning
Amoli et al. [1]	Medium	Medium	No
Multi-AEs [28]	Low	Medium	No
TSCRNN [11]	Low	Medium	No
Stergiopoulos et al. [20]	Medium	High	No
IARF [15]	Low	High	No
DEV-ETA [26]	Low	High	No
GCN-ETA [29]	Medium	High	No
PEAN [13]	Medium	High	Yes
E-minBatch GraphSAGE [10]	Low	High	No
TLS-MHSA(proposed method)	Medium	Excellent	Yes

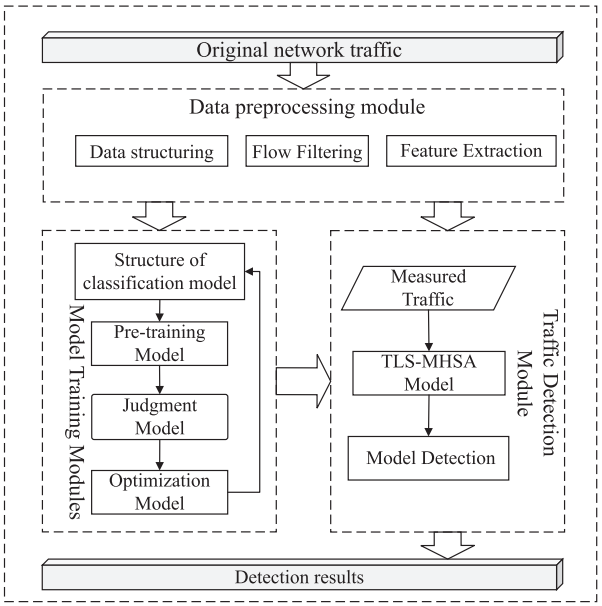


Fig. 1. The framework of TLS-MHSA model.

TLS-MHSA model utilizes the extracted features to perform the detection of the encrypted malicious traffic, thereby obtaining the final detection results.

3.2 Data Pre-processing

Data pre-processing is an important step in data mining. It helps improve the accuracy of the model, and also reduce the time and resources required to train the model. Thus, because the original format of the network traffic is usually the pcap type, it is required to pre-process the original network traffic to meet the input requirements of deep learning model. In addition, the feature extraction operation is also performed to make the extracted features more suitable for detecting the network traffic after encryption by TLS protocol, thereby improving the detection precision of the TLS-MHSA model. The specific pre-processing process on encrypted network traffic is shown in Figure 2. Firstly, the Joy tool [7] is used to parse the encrypted network traffic into .json format for further processing. Secondly, the processed encrypted network traffic in .json format is cleaned and filtered using a Python script to remove the irrelevant network traffic present in the dataset

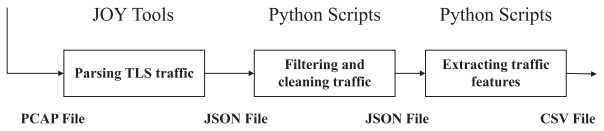


Fig. 2. The process of pre-processing on encrypted network traffic.

and only keep the TLS-encrypted traffic. The order of pre-processed network traffic dataset is disordered, which allows the model to have a stronger generalization ability after learning. Finally, the features of the filtered encrypted network traffic are extracted by the Python script and then convert these features into .csv format, which are directly used as input for the proposed TLS-MHSA.Additionally, we apply a special process to the features of the encrypted TLS protocols. This involves listing the TLS protocol feature variables and converting them to boolean type. We also transform the parameters and TLS extension information in the protocol into binary form. This process ensures that the encrypted TLS protocol features are distinct from non-encrypted protocol features, making them more compatible with the TLS-MHSA model and enhancing overall feature compatibility.

The utilization of Joy tool is justified by its ability to parse network packets and extract a wealth of valuable information, including source IP addresses, destination IP addresses, port numbers, protocol types, timestamps, and more. These kinds of information are crucial for network traffic analysis and malicious network activities detection. [19] and [24] demonstrate the utilization of Joy tool for pre-processing the network traffic. The Joy tool is mainly used to extract features (such as IP address, port number, and protocol type) from raw network traffic packets and convert the packets into JSON format. We further process the extracted features in JSON format using a special scripts developed the team. Specifically, our approach extracts features of encryption protocols, metadata statistics, and time intervals through scripts and integrates these features into CSV files for subsequent modeling and training.

Based on the study of the features of TLS encryption malicious traffic [26] and the traditional feature extraction practices [4] for traffic detection, in addition to the metadata and other features with specific values, all features with binary states should be listed and converted into bool type. For the use of cipher suites, the used features of TLS extension and other features are set to 1 for their vector, while the unused features are set to 0. Feature extraction can be divided into metadata feature selection and TLS protocol encryption traffic feature selection; they are described as follows.

Metadata features are the features in the metadata shared by all traffic as well as the features of traffic traditionally used for traffic detection, such as IP addresses, ports, inbound bytes and outbound bytes, and the like. Because IP addresses are not meaningful for the identification of malicious traffic in most cases and also can easily mislead the judgment of the model. The information of all IP addresses should be removed to prevent them from participating in the feature extraction process. In the extracted features, some metadata is only affected by the data being transmitted but out of control of malicious attackers (such as the number of inbound bytes and outbound bytes, the number of inbound packets and outbound packets, etc.), as well as the normal and malicious traffic will also differ somewhat in the behaviors reflected by these metadata; therefore, these metadata need to be saved to make them participating in feature extraction. In this study, we draw on the traditional standard port matching identification method to contain the ports as one of the traffic features. Considering some differences in the behavior between malicious traffic and benign traffic, some features related to window order statistics are also considered in the feature extraction process, where the relationship between adjacent packets is captured by the Markov transfer matrix.

In addition to the above features, the distance of the average byte, the standard deviation of bytes, and the entropy value of bytes are retained as the metadata of traffic to improve the detection precision of malicious attack encrypted traffic for the model.

The features of encrypted traffic of TLS protocol mainly refer to the information fields and TLS extension in the protocol. Based on the statistical results of this information, the cipher suites, TLS extensions, and TLS extension-related information are extracted as the features in this study. As an important part of the handshaking process when TLS protocol establishes a connection, the information of TLS certificates is also valuable as the features of traffic. In addition, the non-critical fields of TLS protocol (such as the length of the client key) are also retained as additional features to make the extracted TLS encrypted traffic features more comprehensive, thereby enabling them to play a better effect in the subsequent detection process.

After the extraction of features, the network traffic is saved as .csv file to make it able to be directly used as the input of the deep learning model. The extracted features include both discrete features (such as the category and port of TLS certificate) and continuous features (such as average incoming and outgoing bytes and the average length of customer secret key), and the features are represented by 0 or 1, which indicate whether the TLS extension were used or not. The relatively sparse and high dimensional features easily make the model over-fitted and thus further lead to a low detection rate of the model. It is therefore necessary to select the key features from the sparse features to form a subset of features that serve as a true representation of the overall feature set of the encrypted malicious traffic detection.

3.3 Encrypted Malicious Traffic Detection Model Construction

For the encrypted network traffic, the important information such as the payload is encrypted, which causes only the protocol information and statistical information to be able to be used in the detection of encrypted malicious traffic. In relation to the detection of encrypted malicious traffic, the selection of relevant key features is very important, which promotes us to research a method to seek for the key features in the encrypted network traffic.

To achieve this goal, the attention mechanism is often used which has been applied in the field of image classification [12], text summarization [25], target detection [9], and recommendation systems [27]. In 2018, Vaswani et al. [22] improved the attention mechanism and proposed a multi-head self-attention model to model the complex dependencies between words in the machine translation relationship; it solves the problem of over-focusing on its own position when encoding information in the ordinary self-attention mechanism model, and combines the encoded information in different subspaces to enhance the expressive power of the model. The success of multi-head self-attention mechanism in these domains demonstrates its effectiveness in the feature weighting problem. Meanwhile, we also tried various deep neural network architectures. However, these architectures either cannot directly process time series data or require a lot of feature engineering, which leads to the poor performance of these model. In addition, these methods also cannot solve the feature weight problem.

In summary, we finally chose the multi-head self-attention model. The advantages of choosing the multi-head self-attention model are as follows: (1) Through simultaneously attending to different aspects of the input to extract richer features, multi-head self-attention model can increase the expressive power of the model without increasing the network depth; (2) The multi-head self-attention model performs well in processing sequence data, which is suitable for processing the sequence data in encrypted network traffic.

The complete structure of the proposed encrypted malicious traffic detection model is shown in Figure 3, which contains an input layer, an embedding layer, a multi-head self-attention layer, and an output layer. Firstly, the pre-processed traffic feature vector x is fed into the input input layer,

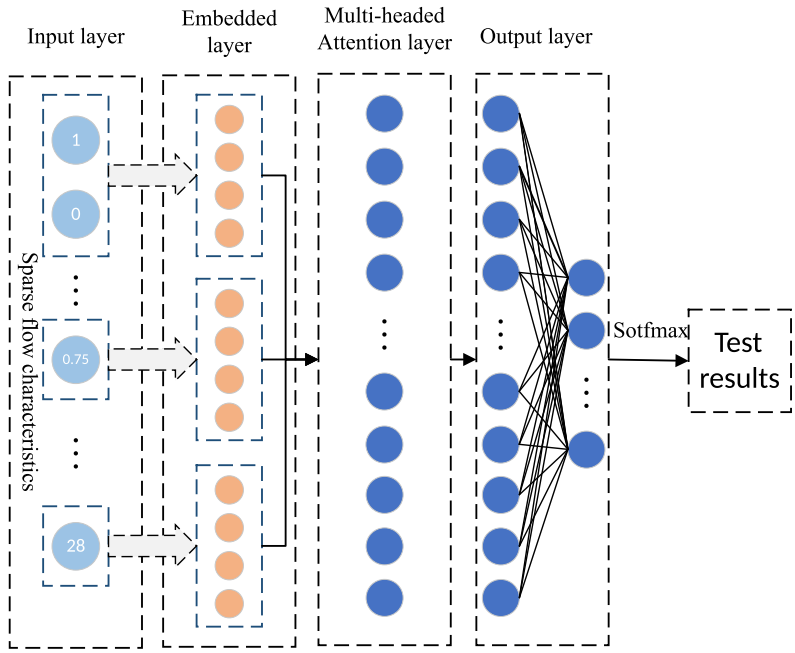


Fig. 3. The structure of TLS-MHSA model.

and all features are mapped to the same low-dimensional space through the embedding layer, and the low-dimensional vector is output. The vectors are then combined into different higher-order features by mapping them into multiple subspaces through a multi-head self-attention mechanism. Additionally, the multiple multi-head self-attention layers are stacked to obtain the multiple feature combinations, while the effectiveness of the feature combinations is judged by attention mechanism. Finally, the feature combination vectors obtained from the upper layer are input into the fully connected layer to output the detection results. The details of each layer are described as follows.

In the TLS-MHSA model, the pre-processed sparse and high-dimensional traffic features are firstly represented as sparse vectors and transmitted to the input layer, which will connect these features together and take them as the input of the model. The categories of the feature vectors are given in formula 1, where A represents the total number of feature categories and x_i represents the i-th feature category. The input layer converts the input features into corresponding feature vectors depending on the feature class, e.g., discrete features are converted into one-hot vectors.

$$x = [x_1; x_2; \ldots x_A \,] \tag{1}$$

The main role of an embedding layer is to transform the sparse feature vectors into suitable dense feature vectors for learning, because neural network is not suitable for learning large-scale sparse vectors. The feature representations of the encrypted network traffic are obtained in the pre-processing phase are sparse and high-dimensional, this will reduce the detection precision. Therefore, we map these feature vectors processed by the input layer into low-dimensional space and represent each feature with multiple low-dimensional vectors. The conversion process is shown in formula 2, where O_i represents the embedding matrix corresponding to feature type I, and x_i is the unique thermal encoding representation vector of features corresponding to feature type.

$$e_i = O_i \, x_i \tag{2}$$

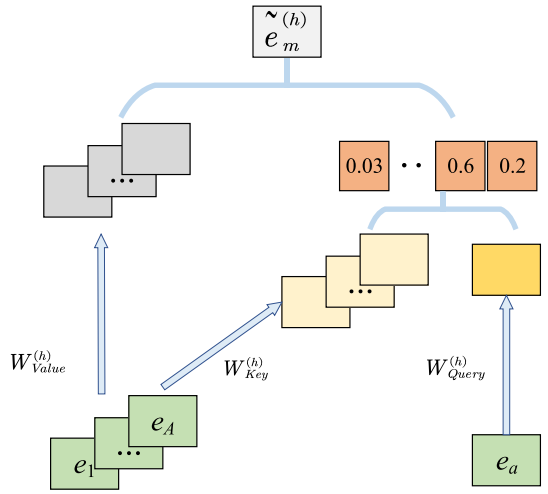


Fig. 4. The structure of multi-head self-concentration layer.

Note that the features of malicious attack encrypted traffic detection have multiple values, such as the TLS encrypted traffic, can support multiple cipher suites. Thus, in this study, the category variable x_i is not a one-hot vector but a multi-hot vector. Hence, we show the multi-valued feature type table S as the average of corresponding feature vectors to make the features normalized, thereby compatible with this multi-valued input, which is shown in formula 3, where v represents the number of multi-valued feature types, and x_i represents the multi-thermal encoded representation vector of features corresponding to the multi-valued feature types.

$$e_i = \frac{1}{v} O_i x_i \tag{3}$$

In addition, the malicious attack encrypted traffic detection features contain both discrete features (such as TLS certificate class) and continuous features (such as average inbound and outbound bytes). In order to enable discrete and continuous features to be combined, we map continuous features into the low-dimensional dense feature vector space and represent continuous features as the result of multiplying their feature values with their corresponding embedding vectors. The continuous features can be represented in formula 4, where v_a is the embedding vector of feature type a, and x_a is the scalar value.

$$e_a = v_a x_a \tag{4}$$

Through the above operations, the embedding layer connects the embedding vectors corresponding to different feature types and also serves as the input of the multi-head self-attention layer. That is, continuous and discrete features will be mapped to a low-dimensional space of uniform dimensionality, thereby allowing for the modeling of higher-order combinatorial features

The role of the multi-head self-attention layer is to capture the correlation between multi-flow features as well as select meaningful features for higher-order combinations, and the complete structure is shown in Figure 4. The core operation of attention mechanism is to obtain the attention weights of feature combinations directly by some operations, and then judge the importance of feature combinations through the weights. Then the multi-head self-attention mechanism uses multiple groups of self-attention to process the input of the network traffic feature vector, and then splices the processing results of all groups together for linear transformation to obtain the final result. What's more, we will use the feature a as an example to explain how to find the important

higher-order features of a. The evaluation of the effectiveness of network traffic features requires determining the correlation between multiple features, and formula 5 [22] defines the correlation between features a and b under the attention head h.

$$\alpha_{a,b}^{(h)} = \frac{\exp\left(S^{(h)}\left(e_a, e_b\right)\right)}{\sum_{l=1}^{A} \exp\left(S^{(h)}\left(e_a, e_l\right)\right)} \tag{5}$$

In formula 5, $S^{(h)}$ is the attention scoring function, where the common dot product model shown in formula 6 is chosen in this paper to define the relevance scores of features a and b under the attention head h. In formula 6, $W_{\text{Query}}^{(h)}$, $W_{\text{Key}}^{(h)}$ both belong to the transformation matrix, and the located dimensions of them are $d' \times d$, which is the transformation matrix that maps the original embedding space R^d to new space $R^{d'}$. And the network traffic feature vectors of x_a and x_b can be represented as e_a and e_b, respectively, which means that the feature vector is changed from the d-dimensional space to a d' -dimensional space. Then the feature representation of e_a is updated by formula 7.

$$S^{(h)}\left(e_a, e_b\right) = \left(W_{\text{Query}}^{(h)}\, e_a\right)^T \left(W_{\text{Key}}^{(h)}\, e_b\right) \tag{6}$$

$$\widetilde{e}_a^{(h)} = \sum_{k=1}^{M} \alpha_{a,b}^{(h)} \left(W_{\text{Value}}^{(h)}\, e_b\right) \tag{7}$$

Formula 7 describes the process of updating attention weights of feature a in subspace h by guiding the combination of all associated features using coefficients $\alpha_{a,b}^{(h)}$, i.e., each feature is represented as a weighted sum of all other associated features. The dimension of $W_{Value}^{(h)}$ is also $d' \times d$. The feature $\widetilde{e}_a^{(h)}$ in the d'-dimensional space is a combined feature obtained by crossing feature x_a with its associated correlated features in subspace h, which representing a new combined feature learned through this research method. Because there is also the possibility that a feature involves several different feature combinations. Thus, we draw on the experience in the transformer and expand the self-attention from one head to multiple heads using the multi-head self-attention mechanism, thereby learning different feature crossover information from the subspace represented by different heads. The feature crossover results learned from different heads can be connected together according to formula 8 together, where symbol $\oplus$ represents the operation of tandem and H represents the number of heads used by multi-head attentional self-mechanism.

$$\widetilde{E}_a = \widetilde{e}_a^{(1)} \oplus \widetilde{e}_a^{(2)} \oplus \cdots \oplus \widetilde{e}_a^{(H)} \tag{8}$$

In order to make the model learn higher-order traffic features while retaining lower-order original traffic features, we add the classical residual network to the multi-head self-attention layer. The residual network and self-attention layer are combined to build the backbone network of the TLS-MHSA model. Specifically, in the input of the model, the original features are embedded as low-dimensional vectors and nonlinearly transformed using residual blocks. And then, the transformed feature matrix is fed into the multi-head self-attention layer to learn the correlation between cross features. Finally, the obtained feature vectors are subjected to convergence and fully connected layers, and then the residual connections are used to consider the original features and newly learned cross features, and leave some original features. The procedure is shown in formula 9, where W_{Res} is used to align the dimension of e_a with $\widetilde{e}_a$, and $\text{ReLU}\ (t) = \max\left(0, t\right)$ is the nonlinear activation function.

$$E_a^{Res} = ReLU\left(W_{Res} e_a + \widetilde{e}_a\right) \tag{9}$$

Through the operation of the multi-head self-attention layer, the feature e_a can be transformed into a new feature representation e_a^{Res}, which is the higher-order crossover feature corresponding

to the original feature e_a after transformation. Meanwhile, the multi-head self-attention neural network with the residual network (composed of multi-head self-attention layers) can perform the superposition of multiple multi-head self-attention layers to enable the learning of arbitrary-order feature combination information, thereby constructing the encrypted network traffic detection model.

The output layer takes a set of feature vectors $\{\mathbf{e}_{\mathbf{a}}^{Res}\}_{a=1}^{A}$ output by multi-head self-attention layer as the input, where feature vectors contain both the original encrypted network traffic features retained in the residual module and the combined features learned through the multi-head self-attention mechanism. The output layer is a detector consisting of a fully connected layer and a softmax layer together. The fully connected layer maps the input feature vector into the sample marker space R^c through a linear transformation to obtain a vector $z \in R^C$, where C is the total number of network traffic categories to be detected. Then, the softmax function is used for detection, and the function is shown in formula 10, where z_i is the probability that the input network traffic sample belongs to traffic category i. The softmax function can convert the multiple output values into a probability distribution ranging from [0, 1] to 1. Next, the softmax layer outputs the probability that the network traffic samples to be detected belong to each traffic category and selects the network traffic category with the highest probability value as the final detection result.

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}} \tag{10}$$

Finally, we perform the model training by calculating the loss values to update the important parameters in the model. The loss function used is a multi-category cross-entropy loss function, and it is shown in formula 11, where p_i denotes the probability vector of the predicted detection results , and y_i denotes the category of the actual sample labels. We also use the Adam optimizer to optimize the model and save the model parameters after training the optimal model.

$$Loss = -\sum_{i=1}^{K} y_i \log(p_i) \tag{11}$$

Combined with the contents described above, the encrypted malicious traffic detection method based on the multi-head self-attention mechanism is summarized in Algorithm 1. The input of this algorithm is the .pcap format encrypted network traffic dataset X, the training parameters of epoch and batch_size. The dataset X is divided into a training set X' and test set X'' after pre-processing operations such as filtering and extracting features. The training set is trained according to the training parameters, and the training process is as follows. Firstly, the features extracted from the training set are mapped to a low-dimensional space of the same dimension through the embedding layer and are transformed into a low-dimensional dense feature vectors. These feature vectors are linearly transformed into a vector representation under the attention space. Then, the correlations between features are calculated, and the features with stronger correlations are combined into a new cross-feature in each self-attention subspace by weighted summation after normalizing the attention distribution. Next, the new features obtained by updating each subspace are spliced to obtain the new combined features with the highest importance, and the original feature information is retained through residual concatenation to learn new features further. Finally, all learned valid features are mapped to all network traffic categories by full concatenation, and the detection result $label'$ is obtained by softmax function, while the loss value between the real label and $label'$ is calculated as well as the model parameters are optimally updated using the Adam algorithm, thereby obtaining the final detection model. With the trained model, the test set X'' is detected to obtain the final encrypted malicious traffic detection results.

ALGORITHM 1: TLS-MHSA

Input:
.pcap dataset X, batch_size, epoch
Output:
malicious attack encrypted network traffic

X', X'' = pre-processing(X);
for each epoch in X' **do**
  for each batch_size in X' dataset **do**
    for each feature a, b in X' dataset **do**
      $e_a, e_b = embedding\,(a, b)$;
      /*Mapping features a and b to the same low-dimensional space through the embedding layer to obtain low-dimensional dense vectors e_a and e_b */
      for attention space h in attention space $[1, 2, \ldots, H]$ **do**
        $\alpha_{a,b}^{(h)} = attention\,(e_a, e_b)$;
        /* Compute the correlation $\alpha_{a,b}^{(h)}$ between feature vectors e_a and e_b in a specific attention space h by means of a multi-head attention mechanism */
        $\widetilde{E}_a^{(h)} = update(\alpha_{a,b}^{(h)})$;
        /* The new feature $\widetilde{e}_a^{(h)}$ obtained by combining feature e_a and feature e_b updated by weighted summation */
      end for
      $\widetilde{E}_a = \widetilde{e}_a^{(1)} \oplus \widetilde{e}_a^{(2)} \oplus \cdots \oplus \widetilde{e}_a^{(H)}$;
      /* The results of the new features under each subspace are stitched together to obtain the final new feature with the highest importance $\widetilde{e}_a$ */
      $E_a^{Res} = ReLU(W_{Res} e_a + \widetilde{e}_a)$;
      /* Use residual networks to reserve some of the original features for the next layer of learning */
    end for
    $z = mapping(\, e_1^{Res}, \ldots, \, e_A^{Res})$;
    /* The cross features are mapped by linear transformation to obtain a vector $z \in R^C$, c is the total number of traffic categories */
    $label' = softmax\,(z)$;
    /* Input the vector z to the softmax function to obtain the detection result labels */
    Compare the predicted class $label'$ with the actual class label and calculate the value of the loss function;
    Updating parameters using Adam on back propagation;
    Model ← model after updating the weights & biases;
  end for
end for
Using the best model to process X'' dataset and generate malicious attack encrypted network traffic detection results Z
return Z;

4 EXPERIMENTAL RESULTS

In this section, we validate the effectiveness of TLS-MHSA on available public datasets, and analyze the experimental results.

4.1 Experimental Dataset

This study used four publicly available datasets, namely Stratosphere, CICIDS-2018, CIRA-CIC-DoHBrw-2020, and CICIDS2017-CTU13. To validate the proposed detection framework proposed in this study. The datasets include:

(1) Stratosphere: this dataset consists of malicious traffic compiled by Roques et al. [16] and normally captured encrypted traffic from the Stratosphere IPS project [21].
(2) CICIDS-2018: this dataset consists of encrypted traffic from the CSE-CIC-IDS2018 dataset [17].
(3) CIRA-CIC-DoHBrw-2020: this dataset consists of encrypted traffic from the CIRA-CIC-DoHBrw-2020 dataset [14].
(4) CICIDS2017-CTU13: this dataset consists of normal encrypted traffic from the CICIDS2017 dataset [17] and malicious encrypted traffic from CTU13 [8].

In practical applications, the use of raw PCAP files from the CICIDS2017 or CTU13 datasets alone is insufficient to meet the experimental requirements. This limitation arises from the relatively limited number of categories present in their respective raw PCAP files. Therefore, we combine these two datasets to create the fourth dataset.

4.2 Experimental Setup

The hardware environment for the experiment is two 20 cores and 40 threads CPUs, 128G RAM and 2 RTX3090 GPUs, and the software environment is Win10 Professional, TensorFlow 2.4.2 , and Python 3.7.

The parameters of the model are set as follows: the batch _size is set to 64, the number of neurons is set to 32, the size of epochs is set to 20, and the learning rate is set to 0.001. In addition, we use 3 layers of multi-head self-attention mechanism in the model.

During the model training process, we partitioned the dataset into training dataset, validation dataset, and test dataset with a ratio of 8:1:1. The validation dataset was used to evaluate the performance of the model under different hyperparameter combinations. We then selected the optimal hyperparameters. Finally, we evaluated the performance of the trained model using the test dataset and compared the experimental results with those obtained from the validation dataset to verify the generalization ability of the model and avoid overfitting issues.

The TLS-MHSA model was benchmarked against seven different methods for effective comparisons. The methods include:

(1) PEAN [13], a Packet-level End-to-end Attentive Network method for encrypted traffic classification.
(2) IARF [15], an improved adaptive random forests-based encrypted malicious network traffic detection method.
(3) DEV-ETA [26], an interpretable detection framework for encrypted malicious network traffic.
(4) GCN-ETA [29], an encrypted malicious network traffic detection method using graph convolutional neural network.
(5) E-minBatch GraphSAGE [10], an improved method based on the E-GraphSAGE algorithm to solve the problem of attack detection in the complex industrial Internet environment.
(6) DTC [20], a decision tree-based malicious network traffic detection method.
(7) RF [18], a random forest-based encrypted network traffic detection method.

4.3 Evaluation Standards

This study used four standard evaluation metrics (shown in Table 2), namely precision, recall, F1-measure, and AUC-ROC to evaluate the experimental results for experimental evaluation; the definitions of the used metrics in these four-evaluation metrics are as follows. Note that these evaluation metrics were applied in previous studies.

True Positive (TP): it indicates the number of positive samples that are correctly detected.

Table 2. The Description of used Evaluation Metrics

Metrics	Formula	Description
Precision	$Precision = \frac{TP}{TP+FP}$	Correctness of the detected encryption attack samples
Recall	$Recall = \frac{TP}{TP+FN}$	Percentage of samples correctly detected
F1-measure	$F_1 = \frac{2\times Precision\times Recall}{Precision+Recall}$	Consider the overall effectiveness of Precision and Recall together
AUC-ROC	$AUC = \int_0^1 \frac{TP}{TP+FP} d\left(\frac{FP}{TP+FP}\right)$	Area under the ROC curve

True Negative (TN): it indicates the number of negative samples that are correctly detected.

False Positive (FP): it indicates the number of samples that were incorrectly detected as positive.

False Negative (FN): it indicates the number of positive samples that are incorrectly classified as other classes.

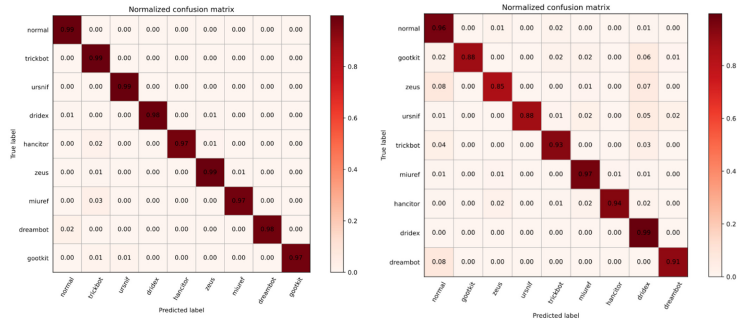
4.4 Analysis of Experimental Results

In order to test the detection capability of the proposed TLS-MHAS model, we performed extensive experiments on the datasets with seven other benchmarked techniques or models. The experimental results are presented based on the confusion matrix, which is shown in Figure 5. In the confusion matrix, the horizontal coordinates represent the precision type of the model and the vertical coordinates represent the true type of the data. The values on the confusion matrix are normalized, and the diagonal lines represent the probability of correct detection by the model.

It can be seen from Figure 5(a) that the diagonal values of the proposed TLS-MHSA model in confusion matrix are close to 95%, which shows the TLS-MHSA model has an excellent detection performance. It can also be seen from Figure 5(b) that the overall performance of the PEAN approach is inferior to the TLS-MHSA model when dealing with smaller samples. The reason for this phenomenon is that the byte sequence and length sequence of packets in per stream are used in the PEAN approach, and the Transformer is used to learn the packet order relationship, which can focus on the content of the data from different perspectives, but it can not learn enough information about the small category traffic from the byte sequence, length sequence and the order relationship. It is shown in Figure 5(c) that the IARF method does not perform well in all categories when processing fewer samples, and its overall performance is not as good as the TLS-MHSA model, which indicates that the IARF method easily occurred overfitting in the online learning process.

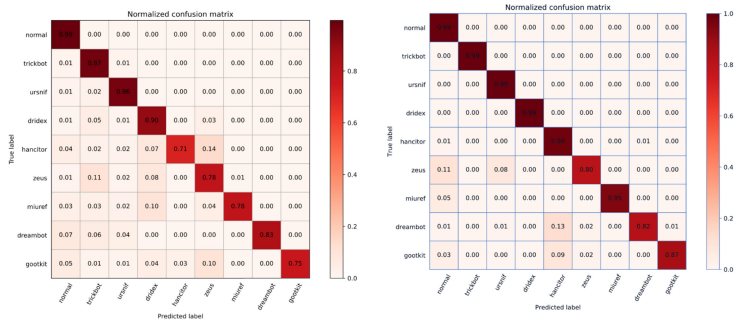
Meanwhile, it can be known from Figure 5(d) that the GCN-ETA method also does not perform as well as the TLS-MHSA model in detecting the network traffic with the types of Zeus, dreambot, and gootkit, while GCN-ETA method has similar results on other samples with the TLS-MHSA model. Similarly, it is shown in Figure 5(e) that the DEV-ETA method does not perform as well as TLS-MHSA model in detecting the network traffic with the types of Zeus, dreambot, and gootkit, which indicates that the DEV-ETA method is less capable of overfitting. In addition, it can be seen from Figure 5(f) that the E-minBatch GraphSAGE method is not as effective as the TLS-MHSA model in detecting network traffic in small sample types, but it is more effective in other categories, which indicates that the E-minBatch GraphSAGE method has a larger problem in detecting small samples. It can be seen from Figure 5(g) that the DTC method has similar drawbacks with the DEV-ETA method. In addition, it can be seen from Figure 5(h) that the RF method is comparable to GCN-ETA method in detecting the network traffic with the types of Zeus, dreambot, and gootkit.

It can be seen from the experimental results presented that the models that do not implement a feature weighting learning system were able to learn or identify the relevant features in a normal traffic and large samples of malicious traffic. Thus, they cannot effectively detect malicious traffic



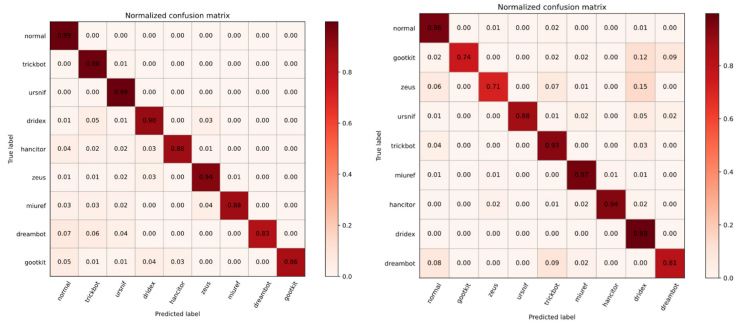
(a) TLS-MHSA

(b) PEAN



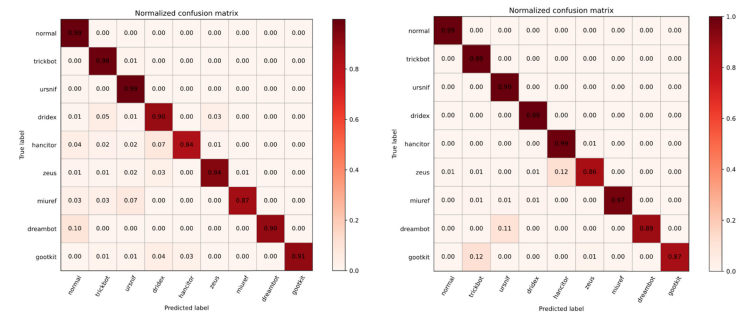
(c) IARF

(d) GCN-ETA



(e) DEV-ETA

(f) E-minBatch GraphSAGE



(g) DT

(h) RF

Fig. 5. The confusion matrix of eight compared methods.

in small sample classes. Compared with them, using a multi-head self-attention mechanism in our proposed TLS-MHSA model avoids this problem. In the TLS-MHSA model, the used multi-head self-attention mechanism layer can better utilize the information of small-scale data through adaptive interaction, which improves the performance of the model in the case of few samples; In addition, the used DNN model can better handle the high-dimensional data through multilayer nonlinear transformations, thus providing better performance in the small-sample cases. Overall, the proposed TLS-MHSA model combines the advantages of multi-head self-attention and DNN models to detect malicious traffic in small sample classes with better performance.

In order to verify the detection capability and stability of the proposed model, we performed a comparative analysis with the seven methods (including PEAN, GCN-ETA, DEV-ETA, IARF, E-minBatch GraphSAGE, DT, RF) with our proposed methods in this experiment. Each experiment is conducted 30 times, and the average experimental result (including Precision, Recall, F1-measure, and AUC-ROC) on detection capability is shown in Table 3, and the stability of each model is shown in Figure 6.

It can be seen from Table 3 that for the Stratosphere dataset, in comparison with GCN-ETA, DEV-ETA, IARF, DT, and RF, the TLS-MHSA model improves the precision by 1.17%, 1.16%, 1.66%, 1.21%, 2.98%, 0.39%, and 1.57%; improves the recall by 1.13%, 1.03%, 1.13%, 1.22%, 4.01%, 0.40%, 0.38%; improves the F1-measure by 1.15%, 1.14%, 1.40%, 1.21%, 3.45%, 0.39%, 0.98%; and improves the AUC-ROC by 1.61%, 2.12%, 1.72%, 1.67%, 3%, 2.02%, and 1.16%, respectively. Similar to that on dataset Stratosphere, the TLS-MHSA model also has the highest precision, recall, F1-measure, and AUC-ROC among the eight detection models on datasets CICIDS-2018, CICIDS2017-CTU13, and CIC-DoHBrw-2020.

As we extract a large number of features related to TLS protocols, certificates, and cipher suites in the feature extraction process, we can have a decent result in detection accuracy, even using ordinary machine learning detection models. However, due to manual feature engineering, there are still some redundant features, while the simple machine learning models cannot avoid the interference of redundant features, thus, the RF and DT still have the lower detection effect among the eight detection models. PEAN employs raw bytes, and length sequences as its input, and leverages a self-attention mechanism to learn the intricate dependencies among network packets in bidirectional flows. Although PEAN yields superior results compared to other approaches, it falls short in scenarios when the sample size is limited. In contrast, the TLS- MHSA outperforms PEAN in such contexts, owing to its ability to handle small size of network traffic effectively. The E-minBatch GraphSAGE model has demonstrated promising detection results by incorporating edge features such as traffic duration and packet size, as well as utilizing pre-sampling techniques on graph-structured data. However, this algorithm faces challenges when applied to small sample detection, resulting in inferior detection performance compared to TLS-MHSA. Combined with the structural features of the traffic, the GCN-ETA model transforms the features into a graph structure, which leads to that it can obtain the best detection performance in all datasets except for the proposed TLS-MHSA model. The DEV-ETA model has the third best detection performance on all datasets as it uses an interpretable analytical framework that focuses more on internal details and can optimize the detection effect through interpretation and validation. The detection performance of IARF model is weaker compared to our method because it relies on online learning of network traffic features. This makes the IARF model more susceptible to errors in weight updates, resulting in the model moving in the wrong direction and residual errors. As a result, the final detection results are not satisfactory.

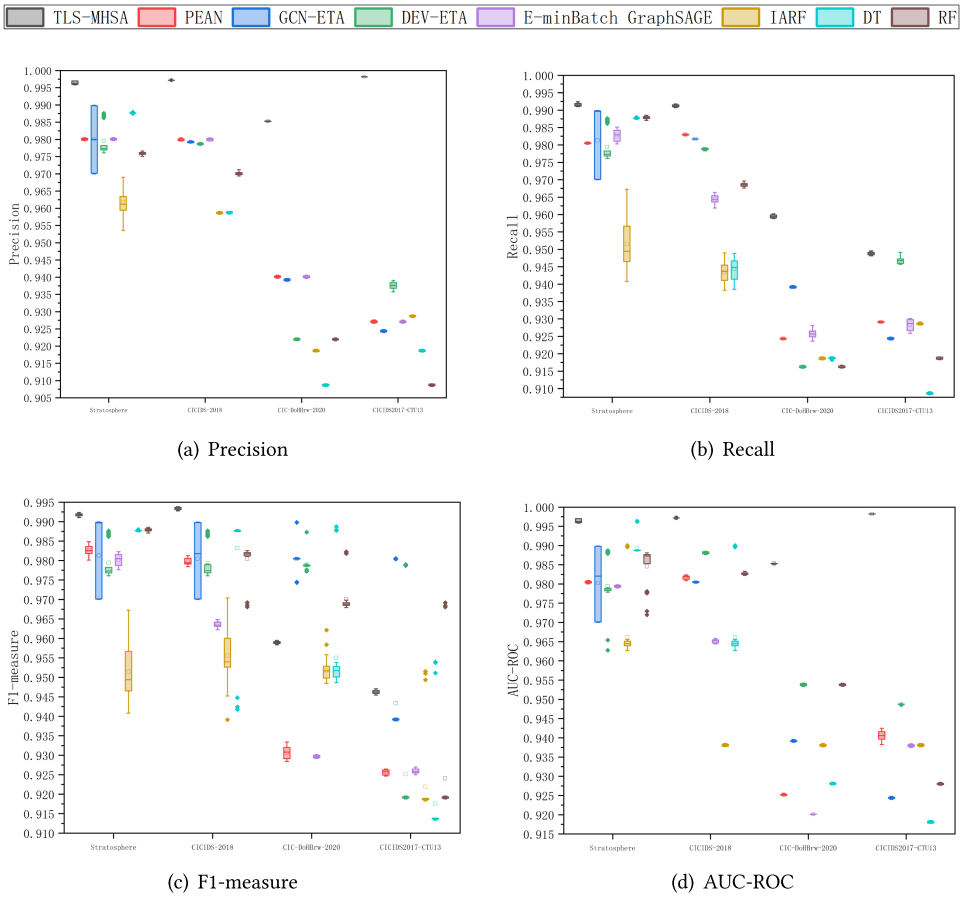
Compared with these seven methods, the proposed TLS-MHSA model achieves the best detection results on all datasets and significantly outperforms the other seven detection models on all four evaluation metrics. It is attributed to the following three reasons:

Table 3. Comparison of Eight Malicious Attack Encrypted Traffic Detection Methods

Models	Datasets	Precision (%)	Recall (%)	F1-measure (%)	AUC-ROC (%)
TLS-MHSA	Stratosphere	**99.16 ± 0.05**	**99.16 ± 0.04**	**99.16 ± 0.04**	**99.61 ± 0.10**
	CICIDS-2018	**99.56 ± 0.05**	**99.14 ± 0.03**	**99.35 ± 0.03**	**99.74 ± 0.01**
	CIC-DoHBrw-2020	**95.83 ± 0.05**	**94.85 ± 0.04**	93.00 ± 0.02	**96.07 ± 0.01**
	CICIDS2017-CTU13	**94.40 ± 0.13**	**94.88 ± 0.23**	**94.64 ± 0.31**	**99.82 ± 0.01**
PEAN	Stratosphere	97.99 ± 0.10	98.03 ± 0.19	98.01 ± 0.14	98.00 ± 0.14
	CICIDS-2018	97.42 ± 0.02	98.27 ± 0.01	97.84 ± 0.01	98.08 ± 0.16
	CIC-DoHBrw-2020	93.31 ± 0.03	92.41 ± 0.18	92.84 ± 0.10	92.50 ± 0.02
	CICIDS2017-CTU13	92.04 ± 0.02	92.89 ± 0.02	92.46 ± 0.02	93.75 ± 0.05
GCN-ETA	Stratosphere	98.00 ± 1.00	98.13 ± 0.99	98.02 ± 0.97	97.49 ± 0.24
	CICIDS-2018	97.92 ± 0.02	98.17 ± 0.01	98.05 ± 0.01	97.88 ± 0.29
	CIC-DoHBrw-2020	93.92 ± 0.03	92.92 ± 0.18	**93.42 ± 0.03**	92.20 ± 0.02
	CICIDS2017-CTU13	92.44 ± 0.02	92.57 ± 0.02	92.50 ± 0.02	93.75 ± 0.10
E-minBatch	Stratosphere	97.50 ± 0.94	98.03 ± 0.96	97.76 ± 0.95	97.89 ± 0.20
-GraphSAGE	CICIDS-2018	96.22 ± 0.08	96.17 ± 0.06	96.19 ± 0.07	96.43 ± 0.16
	CIC-DoHBrw-2020	93.51 ± 0.03	92.34 ± 0.18	92.92 ± 0.03	92.20 ± 0.02
	CICIDS2017-CTU13	92.44 ± 0.02	92.57 ± 0.02	92.50 ± 0.02	93.75 ± 0.10
DEV-ETA	Stratosphere	97.95 ± 0.42	97.94 ± 0.42	97.95 ± 0.42	97.94 ± 0.58
	CICIDS-2018	97.87 ± 0.02	97.88 ± 0.03	97.88 ± 0.02	98.81 ± 0.02
	CIC-DoHBrw-2020	92. 20 ± 0.02	91.63 ± 0.03	91.91 ± 0.02	95.58 ± 0.03
	CICIDS2017-CTU13	93.75 ± 0.10	94.68 ± 0.08	94.21 ± 0.06	94.87 ± 0.01
IARF	Stratosphere	96.18 ± 0.39	95.15 ± 0.70	95.71 ± 0.55	96.61 ± 0.65
	CICIDS-2018	95.87 ± 0.02	94.34 ± 0.32	95.10 ± 0.17	93.81 ± 0.03
	CIC-DoHBrw-2020	91.87 ± 0.03	91.86 ± 0.03	91.87 ± 0.03	93.80 ± 0.02
	CICIDS2017-CTU13	92.87 ± 0.02	92.86 ± 0.03	92.87 ± 0.03	93.75 ± 0.03
DT	Stratosphere	98.77 ± 0.01	98.76 ± 0.01	98.77 ± 0.02	97.59 ± 0.04
	CICIDS-2018	95.88 ± 0.02	94.44 ± 0.32	95.15 ± 0.16	96.61 ± 0.65
	CIC-DoHBrw-2020	90.87 ± 0.02	91.87 ± 0.02	91.37 ± 0.02	92.81 ± 0.02
	CICIDS2017-CTU13	91.87 ± 0.02	90.87 ± 0.03	91.36 ± 0.02	91.81 ± 0.02
RF	Stratosphere	97.59 ± 0.04	98.78 ± 0.04	98.18 ± 0.04	98.45 ± 0.49
	CICIDS-2018	97.01 ± 0.04	96.85 ± 0.05	96.87 ± 0.05	98.27 ± 0.03
	CIC-DoHBrw-2020	92.20 ± 0.02	91.62 ± 0.03	91.91 ± 0.02	95.38 ± 0.02
	CICIDS2017-CTU13	90.87 ± 0.03	91.87 ± 0.03	91.37 ± 0.02	93.80 ± 0.02

(1) The multi-head self-attention mechanism has the ability to automatically learn the key features, which can provide more effective features for the training of the model.
(2) The focus on key features can reduce the false positive caused by overfitting.
(3) In the pre-processing stage of our TLS-MHSA model, we extracted comprehensive features that are closely related to encrypted malicious traffic, enabling the TLS-MHSA model to select highly relevant features from them. Additionally, compared to seven other methods, the standard deviation of performance metrics for the TLS-MHSA model is the smallest, indicating that the TLS-MHSA model has the highest reliability.

It can be seen from Figure 6 that compared with the PEAN, GCN-ETA, DEV-ETA, E-minBatch GraphSAGE, IARF, DT, and RF models, most of the precision, recall, F1-measure , and AUC-ROC of the TLS-MHSA model are higher, and there are no outliers in the experimental data of the TLS-MHSA model. In addition, the interquartile ranges (the box length) of the TLS-MHSA model are



(a) Precision (b) Recall

(c) F1-measure (d) AUC-ROC

Fig. 6. The stability of eight compared methods.

apparently the least, which means that the detection efficiency of the TLS-MHSA model is more stable. It can be seen in Figure 6(a) that the precision of the box-line plot demonstrates that the stability of the precision of the TLS-MHSA model is the highest among all compared models, and the TLS-MHSA model has no outlier in all datasets. In contrast, in the Stratosphere dataset, GCN-ETA and IARF models exhibit significant fluctuations, while PEAN, DT, E-minBatch GraphSAGE, and RF exhibit stability, but with lower accuracy values compared to TLS-MHSA. Finally, the DEV-ETA model shows some outliers.

Based on the graph in Figure 6(b), the recall of IARF on the Stratosphere dataset is unstable. This is because the results are obtained using an online learning model, which is susceptible to errors in weight updates. If a weight update error occurs at a certain point, subsequent weight updates may continue to be incorrect, which leads the model gradually close to the wrong direction and introducing residuals, thereby exhibiting high volatility. This is the reason why IARF exhibits lower stability compared to TLS-MHSA. On the other hand, PEAN, DT, and RF are relatively stable, but their average values are lower than that of the TLS-MHSA model, while DEV-ETA still exhibits a considerable number of outliers. The same issue is also present in the other three datasets. Figure 6(c) and Figure 6(d) also show that the TLS-MHSA is the most stable model in these eight compared models and has the highest AUC-ROC and F1-measure without any outliers. PEAN is the most stable model in terms of F1-measure and AUC-ROC, second only to the

TLS-MHSA model, as it utilizes the raw byte or length sequence, which contain a considerable amount of information and are less prone to fluctuations. The large fluctuations in the values of the detection metrics of GCN-ETA on all datasets are because this model uses only five stream features of TCP streams as features of graph structure edges, which are susceptible to individual extreme values. E-minBatch GraphSAGE uses the AGG aggregation function to aggregate edge information of the graph structure, making it more stable than GCN-ETA. The IARF model also has large fluctuations in the values of the detection metrics on all datasets because this model is an online learning algorithm and easily fluctuates during the learning process. Compared with these models, using a multi-head self-attention mechanism in the TLS-MHSA model can learn the key features of encrypted traffic and provide more effective features for the detection process. Thus, our proposed TLS-MHSA model can accurately identify and analyze multiple types of encryption attack traffic, thus achieving a huge improvement in detection efficiency.

In summary, a large number of experimental results demonstrate that after extracting comprehensive features of encrypted attack network traffic using our pre-processing method, the TLS-MHSA model is able to use these features to further construct a high-quality feature combination through multi-head self-attention mechanism, which makes the model have higher precision and relatively stable identification performance compared with the other seven detection models. Therefore, the comparison of experimental performance evaluation can prove that the use of a multi-head self-attention mechanism in the TLS-MHSA model has more substantial effectiveness.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented a detailed introduction to our encrypted malicious traffic detection model, which is based on a multi-head self-attention mechanism and is called TLS-MHSA. First, we use the Joy tool to extract the features of a complete TLS encrypted traffic and the Python scripts to filter the features that are not relevant to the detection of encrypted malicious traffic, thereby improving the detection efficiency. Then we obtain the multiple feature combinations by stacking multiple self-attention layers and enhance the detection efficiency by judging the effectiveness of the feature combinations through the attention mechanism, thereby obtaining the most important new feature combinations. Meanwhile, we optimize the parameters and the structure of the detection model by continuously pre-training it with training and test datasets to obtain the best detection model. We also conduct extensive experiments to verify the efficiency of the proposed TLS-MHSA model for malicious encrypted traffic detection in terms of precision, recall, F1-measure, and AUC-ROC. The experimental results show that the proposed TLS-MHSA model can detect encrypted malicious network traffic with high stability and more accurately.

Although the proposed TLS-MHSA model achieves good detection results, it also has some shortcomings too that need to be overcome in the future:

(1) The encrypted traffic dataset used in this study is based on TLS v1.2 protocol, but the detection efficiency of the TLS-MHSA model on the datasets based on TLS v1.3 protocol may decline. Therefore, it is necessary to develop a new encrypted malicious traffic detection model for the network traffic based on TLS v1.3 protocol.
(2) There is also some room for further optimizing the feature weight learning and feature interaction capabilities of the multi-head self-attention-based model, thereby making it can adapt to the increasingly complex and changing encrypted malicious traffic.

REFERENCES

[1] Payam Vahdani Amoli, Timo Hamalainen, Gil David, Mikhail Zolotukhin, and Mahsa Mirzamohammad. 2016. Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets. *JDCTA (International Journal of Digital Content Technology and its Applications* 10, 2 (2016), 1–13.

[2] Blake Anderson and David McGrew. 2016. Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. Association for Computing Machinery, New York, NY, USA, 35–46. https://doi.org/10.1145/2996758.2996768

[3] Blake Anderson, Subharthi Paul, and David McGrew. 2018. Deciphering malware's use of TLS (without decryption). *Journal of Computer Virology and Hacking Techniques* 14, 3 (2018), 195–211.

[4] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. 2006. Early application identification. In *Proceedings of the 2006 ACM CoNEXT Conference*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/1368436.1368445

[5] Zigang Cao, Gang Xiong, Yong Zhao, Zhenzhen Li, and Li Guo. 2014. A survey on encrypted traffic classification. In *Applications and Techniques in Information Security*, Lynn Batten, Gang Li, Wenjia Niu, and Matthew Warren (Eds.). Springer Berlin, Berlin, 73–81.

[6] Jinfu Chen, Tianxiang Lv, Saihua Cai, Luo Song, and Shang Yin. 2023. A novel detection model for abnormal network traffic based on bidirectional temporal convolutional network. *Information and Software Technology* 157 (2023), 107166. https://doi.org/10.1016/j.infsof.2023.107166

[7] Cisco. 2019. Joy. https://github.com/cisco/joy, Retrieved on 2019-11-8.

[8] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. *Computers & Security* 45 (2014), 100–123.

[9] Han Guo, Juan Cao, Yazi Zhang, Junbo Guo, and Jintao Li. 2018. Rumor detection with hierarchical social attention network. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18)*. Association for Computing Machinery, New York, NY, USA, 943–951. https://doi.org/10.1145/3269206.3271709

[10] Jin Lan, Jia Z. Lu, Guo G. Wan, Yuan Y. Wang, Chen Y. Huang, Shi B. Zhang, Yu Y. Huang, Jin N. Ma, and Robertas Damaševičius. 2022. E-MinBatch GraphSAGE: An industrial internet attack detection model. *Sec. and Commun. Netw.* 2022 (Jan. 2022), 12 pages. https://doi.org/10.1155/2022/5363764

[11] Kunda Lin, Xiaolong Xu, and Honghao Gao. 2021. TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT. *Computer Networks* 190 (2021), 107974.

[12] Lan Lin, Huan Luo, Renjie Huang, and Mao Ye. 2019. Recurrent models of visual co-attention for person re-identification. *IEEE Access* 7 (2019), 8865–8875.

[13] Peng Lin, Kejiang Ye, Yishen Hu, Yanying Lin, and Cheng-Zhong Xu. 2023. A novel multimodal deep learning framework for encrypted traffic classification. *IEEE/ACM Transactions on Networking* 31, 3 (2023), 1369–1384. https://doi.org/10.1109/TNET.2022.3215507

[14] Mohammadreza MontazeriShatoori, Logan Davidson, Gurdip Kaur, and Arash Habibi Lashkari. 2020. Detection of DoH tunnels using time-series classification of encrypted traffic. In *2020 IEEE Intl. Conf. on Dependable, Autonomic and Secure Computing, Intl. Conf. on Pervasive Intelligence and Computing, Intl. Conf. on Cloud and Big Data Computing, Intl. Conf. on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, Calgary, AB, Canada, 63–70. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026

[15] Zequn Niu, Jingfeng Xue, Dacheng Qu, Yong Wang, Jun Zheng, and Hongfei Zhu. 2022. A novel approach based on adaptive online analysis of encrypted traffic for identifying malware in IIoT. *Information Sciences* 601 (2022), 162–174.

[16] Olivier Roques, S. Maffeis, and M. Cova. 2019. *Detecting Malware in TLS Traffic*. Master's thesis. Imperial College London.

[17] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP* 1 (2018), 108–116.

[18] Anish Singh Shekhawat, Fabio Di Troia, and Mark Stamp. 2019. Feature analysis of encrypted malicious traffic. *Expert Systems with Applications* 125 (2019), 130–141.

[19] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1745–1759.

[20] George Stergiopoulos, Alexander Talavari, Evangelos Bitsikas, and Dimitris Gritzalis. 2018. Automatic detection of various malicious traffic using side channel features on TCP packets. In *Computer Security*, Javier Lopez, Jianying Zhou, and Miguel Soriano (Eds.). Springer International Publishing, Cham, 346–362.

[21] Stratosphere. 2015. Stratosphere Laboratory Datasets. https://www.stratosphereips.org/datasets-overview, Retrieved March 13, 2020.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017). https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[23] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. 2015. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management* 25, 5 (2015), 355–374.

[24] Gernot Vormayr, Joachim Fabini, and Tanja Zseby. 2020. Why are my flows different? A tutorial on flow exporters. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2064–2103.

[25] Tianwei Wang, Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen, Yaqiang Wu, Qianying Wang, and Mingxiang Cai. 2020. Decoupled attention network for text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, Palo Alto, CA, 12216–12224. https://doi.org/10.1609/aaai.v34i07.6903

[26] Luming Yang, Shaojing Fu, Yongjun Wang, Kaitai Liang, Fan Mo, and Bo Liu. 2022. DEV-ETA: An interpretable detection framework for encrypted malicious traffic. *Comput. J.* 66, 5 (03 2022), 1213–1227. https://doi.org/10.1093/comjnl/bxac008. arXiv:https://academic.oup.com/comjnl/article-pdf/66/5/1213/50397336/bxac008.pdf

[27] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention network. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. AAAI Press, Palo Alto, CA, 3926–3932.

[28] Tangda Yu, FuTai Zou, Linsen Li, and Ping Yi. 2019. An encrypted malicious traffic detection system based on neural network. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, Guilin, China, 62–70. https://doi.org/10.1109/CyberC.2019.00020

[29] Juan Zheng, Zhiyong Zeng, Tao Feng, and Leandros Maglaras. 2022. GCN-ETA: High-efficiency encrypted malicious traffic detection. *Sec. and Commun. Netw.* 2022 (Jan. 2022), 11 pages. https://doi.org/10.1155/2022/4274139

Received 22 November 2022; revised 31 May 2023; accepted 27 July 2023