Computer Systems Science & Engineering
DOI:10.32604/csse.2021.015074
Article

Tech Science Press

TLSmell: Direct Identification on Malicious HTTPs Encryption Traffic with Simple Connection-Specific Indicators

Zhengqiu Weng[1,2], Timing Chen[1,*], Tiantian Zhu[1], Hang Dong[1], Dan Zhou[1] and Osama Alfarraj[3]

[1]School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, 310023, China
[2]Department of Information Technology, Wenzhou Polytechnic, Wenzhou, 325035, China
[3]Computer Science Department, Community College, King Saud University, Riyadh, 11437, Saudi Arabia
[*]Corresponding Author: Timing Chen. Email: tmchen@zjut.edu.cn
Received: 05 November 2020; Accepted: 13 December 2020

Abstract: Internet traffic encryption is a very common traffic protection method. Most internet traffic is protected by the encryption protocol called transport layer security (TLS). Although traffic encryption can ensure the security of communication, it also enables malware to hide its information and avoid being detected. At present, most of the malicious traffic detection methods are aimed at the unencrypted ones. There are some problems in the detection of encrypted traffic, such as high false positive rate, difficulty in feature extraction, and insufficient practicability. The accuracy and effectiveness of existing methods need to be improved. In this paper, we present TLSmell, a framework that conducts malicious encrypted HTTPs traffic detection with simple connection-specific indicators by using different classifiers based online training. We perform deep packet analysis of encrypted traffic through data pre-processing to extract effective features, and then the online training algorithm is used for training and prediction. Without decrypting the original traffic, high-precision malicious traffic detection and analysis are realized, which can guarantee user privacy and communication security. At the same time, since there is no need to decrypt the traffic in advance, the efficiency of detecting malicious HTTPs traffic will be greatly improved. Combined with the traditional detection and analysis methods, malicious HTTPs traffic is screened, and suspicious traffic is further analyzed by the expert through the context of suspicious behaviors, thereby improving the overall performance of malicious encrypted traffic detection.

Keywords: Cyber security; malware detection; TLS; feature engineering

1 Introduction

HyperText Transfer Protocol Secure (HTTPs) is a transmission protocol for secure communication through a computer network. The HTTPs protocol itself still complies with the HTTP protocol standard, but the content of its data packets is encrypted by SSL/TLS [1]. At present, there are more than 200 malware families that use encrypted communication, accounting for more than 40%, covering almost all common types, such as Trojan horses, ransomware, infections, and worms, downloaders, among

which Trojan horses and downloader malware families account for a relatively high proportion [2]. The encrypted traffic generated by malware can be divided into the following six categories according to the purpose: C&C direct link, detection of the host network environment, normal communication of the mother body, hidden transfer of white stations, and worm propagation communication. Therefore, intelligent analysis of HTTPs traffic generated by malware has attracted a lot of attention recently.

Currently, the detection method for identifying malicious encrypted traffic mainly involves installing an intercepting agent. This solution deploys a special certificate at the gateway and configures the computers in the LAN to trust the certificate, so that the HTTPs traffic of all computers in the LAN is decrypted at the gateway. Then use traditional malicious traffic detection methods to audit the decrypted traffic. If the audit is passed, the traffic will be re-encrypted and sent to the computers in the domain [3]. Although traditional traffic auditing methods are relatively simple in technical implementation, there are still major drawbacks. Deploying interception agents is costly and requires large hardware calculations, at the same time, a series of rules need to be configured to achieve detection, so the overall flexibility is not enough.

Malicious encrypted traffic is the pain and difficulty of current traffic security detection. How to detect malicious encrypted traffic without decryption, machine learning (ML) can provide quite an effective solution. Traditional machine learning relies on training datasets and feature engineering, and various types of malicious encrypted traffic collected are various and may contain "impurities". If these data are not distinguished and directly trained, it will affect the accuracy and false positive rate of model detection [4].

This paper proposes a technical approach to identify the malicious encrypted traffic with simple connection-specific indicators based on machine learning algorithms. The encrypted traffic is analyzed in-depth by data pre-processing to obtain three file logs, namely ssl log, connection log, and certificate log. Further correlation analysis is performed on the three logs to obtain the connection-specific indicators (4-tuple). The content of the 4-tuple includes the source IP, destination IP, destination port, and protocol. Then, these 4-tuple are used as units to extract features. Finally, through training and prediction by model, high-precision malicious traffic detection and analysis is achieved without decrypting the original traffic. It can ensure the user's privacy and communication security. The efficiency of HTTPs traffic detection will be greatly improved. This method can be used as a preliminary to screen the HTTPs malicious traffic. If suspicious traffic is found, it can be further decrypted and confirmed, and ultimately improve the efficiency of the overall detection and analysis of malicious encrypted traffic. Our contributions can be summarized as follows:

- Three kinds of log information are extracted to obtain the connected 4-tuple.By adopting three different feature selection methods, the most representative features are analyzed, which helps to achieve better classification results, and reduce the cost of model training, and improve the final accuracy of the model.
- A framework of online malicious traffic detection is proposed for detecting TLS encrypted malware using a public dataset. And we make a comparison between the performance of various classifiers, such as SVM, LSTM, and CNN classifiers.
- Further use expert knowledge for feedback to enhance the classification effect, so as to reduce the false alarm rate.
- We perform online deployment models and regular detection in the actual environment, and the experimental results on a large number of dataset shows that this method has high accuracy and practicability.

The rest of this paper is listed as follows. Section 2 introduces the related works about encrypted traffic analysis using machine learning. Section 3 describes our proposed ***TLSmell*** framework, including data pre-processing and model architecture. Section 4 presents the performance evaluation results from various experimental analysis. Finally, we conclude the paper in Section 5.

2 Related Work

At present, most of the traffic in the network is transmitted in an encrypted manner, which poses a severe challenge for the detection of malicious traffic. A large number of scholars in academia conduct research on it in different scenarios [5–9]. Wang et al. [10] proposed an N-gram model for malicious web traffic detection. However, when N is large, the N-gram model is very sensitive to the training data, which will lead to the insufficient fitting of the detection model to large training samples. Their system Anagram proposed the idea of using legitimately requested hash values to maintain the generalization ability of the detection model to a certain extent. However, the effectiveness of this method is low due to the high variability of web requests. Lokoc et al. [11] proposed a kNN-based encryption malware detection method, which focuses on metric indexing to approximate k-NN search on a few high-dimensional descriptor network traffic datasets.

Přemysl Čech et al. [12] use grid histograms and MapReduce approach in a scalable way to extract feature and compare the representation using linear and k-NN classifiers. [13] are based on typical data mining techniques, such as meta-learning, classification, and association rules. Try to train intrusion detection models using audited web traffic data. Their results show that data mining techniques can identify web attacks and show that more data mining models can be tried. Claffy et al. [14] introduced the performance and importance of data sampling methods related to network traffic. Saber et al. [15] combine oversampling and undersampling, followed by PCA, which can select the best feature subset before using SVM for effective traffic classification. Su etc. [16] adopted a hierarchical sampling method that found benign and malicious clusters from the original network traffic, and then analyzed the clusters for filtering and further malware detection.

Prasse et al. [17] used LSTM neural network model and random forest classifier to detect malware in encrypted network traffic. Anderson [18] proposed a machine learning method based on logistic regression and SVM to identify malware in encrypted network traffic, including 20 functions in TLS, DNS, and HTTP data. Anderson et al. [19] identified encrypted malicious traffic by analyzing network metadata and applied supervised machine learning algorithms.

The above methods are all effective, but for encrypted traffic, ordinary sampling methods cannot accurately obtain the corresponding results. It is still a challenge to accurately detect malicious traffic, especially encrypting malicious traffic.

In this paper, we propose an malicious HTTPs traffic detection framework using different classifiers based online training. Through the in-depth analysis of the encrypted traffic to extract the features, the 4-tuple is constructed, and then the online training algorithm of the DL model is used for training and prediction. There is no need to decrypt the traffic, which greatly improves the operating efficiency and uses expert support to enhance the model to improve accuracy.

3 Methodology

In this section, we first introduce the overall architecture of our proposed malicious HTTPs traffic detection model, TLSmell, using machine learning with online training algorithm. We then discuss several important topics of three parts, including the feature engineering, pre-processing and the classification model. System architecture of TLSmell is shown in Fig. 1

3.1 TLSmell Framework

First, we use the data collector to collect the traffic, and store it in Hive after in-depth packet analysis, read the data from Hive regularly, load the model, and perform detection. In the process of specific implementation, it is also necessary to clean and filter the training dataset. For example, the malicious traffic dataset will be mixed with some benign traffic, which is filtered according to the domain name to ensure the accuracy of the training dataset.

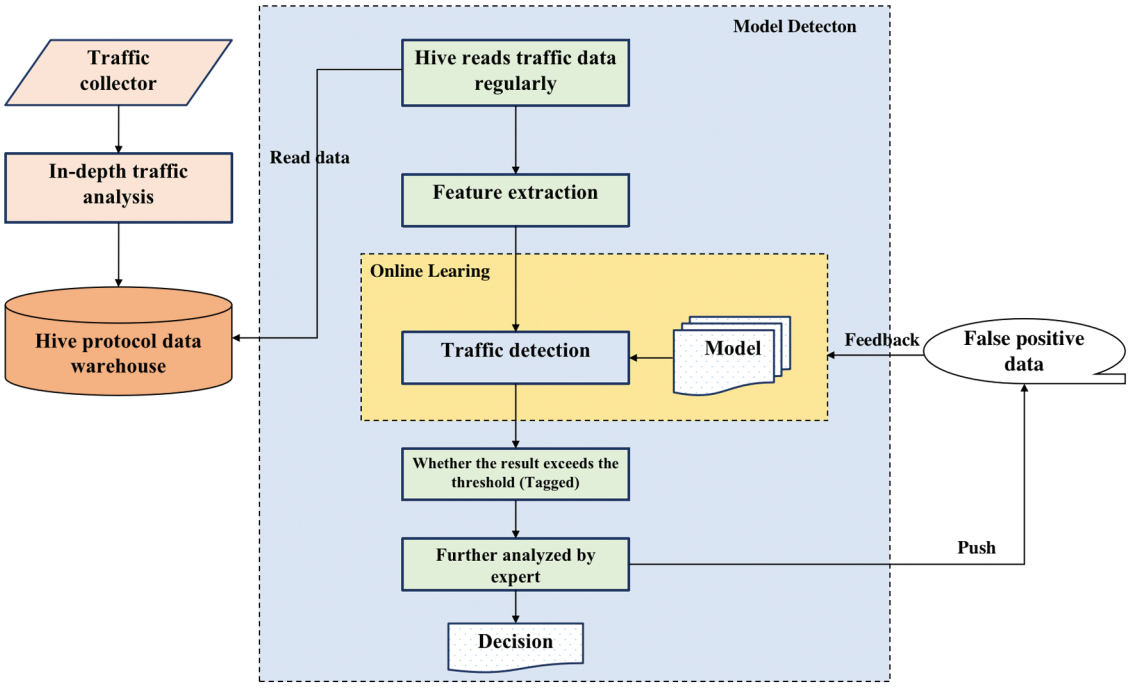

Figure 1: System architecture of TLSmell

In the actual experiments, we select a small number of labeled samples to train the HTTPs malicious traffic analysis model to obtain the first generation version. In the modeling process, we pre-load the trained first-generation model, extract features from the data read in Hive, and input the extracted feature vector into the model. The model performs detection and analysis on the feature vector of the input sample. We labeled the samples according to the setting of threshold standard ($\alpha = 0.5$). If the ones exceed the threshold, they will be labeled as negative samples, otherwise as positive samples. In order to reduce the negative impact of the model's own error on the subsequent detection, the labeled samples will be processed by security experts manually to confirm. If there are no false positives, then directly end. Otherwise, save the falsely reported data samples (including positive and negative samples), correct the label, and feed back to the online learning module. The false positive samples are added to the set for updating. After the batch processing of the samples is completed, the weight vector is updated so that the model parameters are updated in real-time to prepare for the next round of traffic detection, and then continue to repeat the above steps.

3.2 Pre-Processing

To conduct our experiments, we leveraged a public dataset (MCFP), consisting of several malicious, benign, and hybrid networks traffic packet capture files, which comes from Stratosphere Malware Capture Facility Project of the Czech Technical University [20]. A set of malicious TLS network traffic was chosen for detection in our work. In addition, since the MCFP dataset mainly contains HTTPs traffic of various malwares, and lacks benign TLS network traffic, we simulated and captured more benign traffic through visiting a series of mainstream websites, and using Wireshark to capture and filter them.

MCFP contains nearly 400 malicious sub-dataset captured by botnets. Some of these datasets provide log files that have been parsed by Zeek, including conn.log, ssl.log, x509.log, etc.. Zeek IDS is an open source network traffic analyzer, which itself is used for security monitoring, and it also supports a wide range of traffic analysis tasks. The traffic data collected by the above log dataset are all included in the

pcap file. If the log information is not included in the sub-dataset, we will use the powerful traffic analysis function of Zeek IDS to perform deep packet analysis through the pcap file given in the dataset to generate the corresponding network activity log file.

After Zeek IDS processing, three important log files are obtained. These three important files need to be further aggregated to form a "4-tuple-connected". The quadruple includes source IP, destination IP, destination port and protocol. And then the three log files are classified and aggregated according to the connection information. The details are shown in Fig. 2.

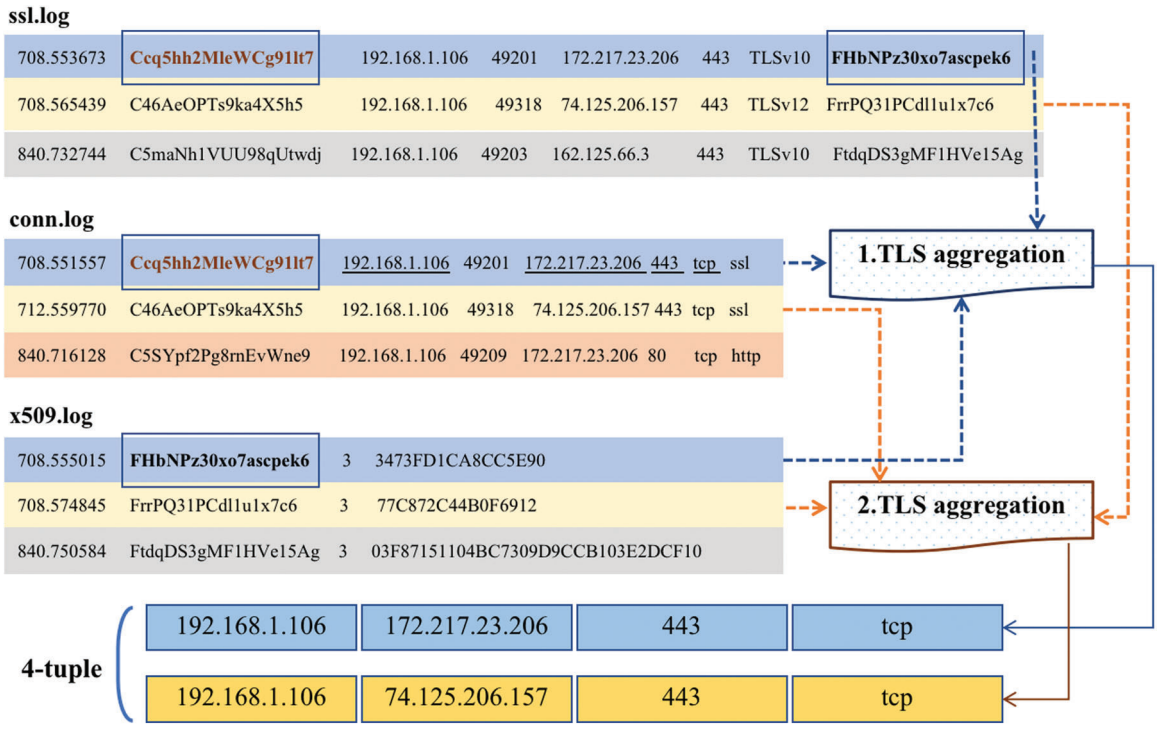

Figure 2: Get the 4-tuple through the logs connection

The steps to connect tuples are as follows:

Step 1. Read an TLS record from the ssl.log file, obtain its unique key, use the key to find the unique connection record in the conn.log file, and obtain the content of the 4-tuple and the label of the connection (benign or malicious). If the found connection record has no corresponding label or no connection record is found, skip to the next TLS record.

Step 2. If the 4-tuple is successfully found, search the first certificate record matching the certificate path recorded by TLS in the x509.log file.

After the above 2 steps, if three records are successfully found, then determine whether there is such TLS aggregation information in the TLS aggregation pool, and if not, add it to the TLS aggregation pool.

Step 3. After TLS is aggregated, the feature information can be further extracted based on these HTTPs records. The final model training set is shown in Tab. 1.

Since the original data is non-quantified data, it needs to be quantified by feature extraction and then classified. In the next sections, we will introduce how to extract the effective features.

Table 1: Model training set with 4-tuple

4-tuple	Feature1	Feature2	…	label
192.168.1.1, 102.35.45.6, 443, tcp	f1	f2	…	benign
192.168.3.1, 142.36.15.6, 443, tcp	f1	f2	…	malicious

3.3 Feature Engineering

After getting the dataset, the next step is to extract its features. From our analysis and feature creation process we extracted 33 important indicators from each 4-tuple-connected. Most of them were created based on professional knowledge in the field and thorough analysis of malware data. For these features, they are divided into 3 groups: connection indicators, TSL indicators, and certificate indicators. Connection characteristics are characteristics from connection records that describe common behaviors of communication flows that are not related to certificates and encryption. The TSL feature is the feature from the TSL record, which describes the information of the TSL handshake and encrypted communication, and the certificate feature is the feature from the certificate record, which describes the information of the certificate provided to the project by the web service personnel during the TSL handshake. We pre-processed the 33 extracted features, such as normalization and missing value replacement. For example, if the feature cannot be calculated due to lack of information, assign it -1. Standardize all features, such as (x-x.mean)/x.std. In Tab. 2, the features are explained in detail.

Table 2: Features created

Feature	description	Category
num	Number of SSL aggregation and connection records	connection
mean_duration	Mean of continuous time	connection
std_duration	Standard deviation of continuous time	connection
range_count	Standard deviation of the duration range	connection
orig_bytes	Number of payload bytes from originator	connection
resp_bytes	Number of payload bytes from responder	connection
orig_pkts	Number of packets from originator	connection
resp_pkts	Number of packets from responder	connection
mean_orig_bytes	Average number of originator bytes	connection
mean_resp_bytes	Average number of responder bytes	connection
mean_orig_pkts	Average number of originator packets	connection
mean_resp_pkts	Average number of responder packets	connection
std_orig_bytes	Number of Standard deviation originator bytes	connection
std_resp_bytes	Number of standard deviation responder bytes	connection
std_orig_pkts	Number of Standard deviation originator packet	connection
std_resp_pkts	Number of standard deviation responder packets	connection
ssl_orig_pkts/ orig_pkts	Proportion of originator encrypted data packet statistics to the total data packet	connection

Table 2 (continued).

Feature	description	Category
ssl_resp_pkts/ resp_pkts	Proportion of responder encrypted data packet statistics to total data packets	connection
orig_bytes_li	Maximum number of originator bytes	connection
resp_bytes_li	Maximum number of responder bytes	connection
orig_pkts_li	Maximum number of originator packets	connection
resp_pkts_li	Maximum number of responder data packets	connection
ssl_duration/ sum_duration	Ratio of ssl/tls connection duration to the duration of the data stream	connection
avg_time_diff	Average secondary time difference	connection
fn/fs	Ratio of Connection record and SSL aggregation	TSL
cert/x509_data	Ratio of Self-signed certificate	TSL
tls_num/ssl_num	Ratio of TLS to SSL	TSL
sni_num	SNI ratio	TSL
sni_ip_num	SNI is IP	TSL
valid_count	Percentage of valid certificates	certificate
valid_cert	Validity of certificate period during capture	certificate
mean_cert	Average age of the certificate	certificate
mean_squre_cert	Age standard deviation of the certificate	certificate
num_san_dns	Number of DNS in the certificate	certificate

3.4 Classification Models

After data pre-processing, feature extraction and selection, finally a corresponding model needs to be constructed for malicious detection. For the classification target, combining the actual data characteristics to select the appropriate classifier is helpful to improve the discernment of the model. Considering the multi-dimensional characteristics of the feature and the dependency relationship between the features of each dimension, we propose a traditional machine learning (SVM) and two deep learning architectures (CNN, LSTM) for the detection of malicious traffic.

SVM (Support Vector Machine) is a supervised learning model for classification and regression analysis [21]. The traditional machine learning model proposed for malware TLS encrypted network traffic classification is a nonlinear SVM using radial basis function (RBF) as Kernel function. It can effectively use nuclear techniques for nonlinear classification and map the input to a high-dimensional feature space. We use RBF as Kernel function in this paper.

CNN (Convolutional Neural Networks) is a type of Feedforward Neural Networks (FNN) that includes convolution calculations and a deep structure. It is one of the representative algorithms of deep learning [22]. CNN has the ability of representation learning, and can perform shift-invariant classification of input information according to its hierarchical structure. The parameters of each layer of the CNN architecture is displayed in Tab. 5 and Fig. 3.

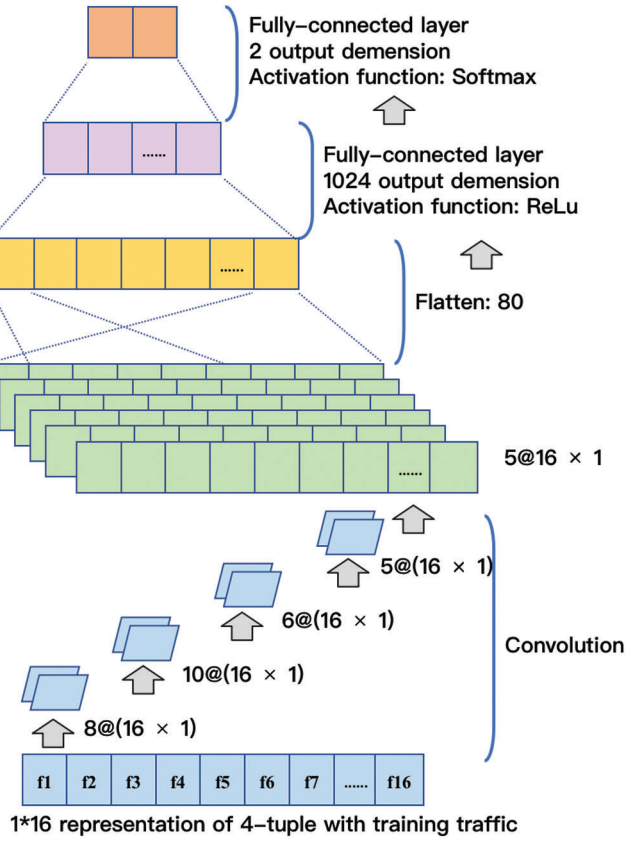

Figure 3: CNN architecture

LSTM (Long short-term memory) overcomes the vanishing gradient problem of RNN (Recurrent neural network) by adding storage units [23]. These storage units include forget gates, input gates, and output gates, adding filtering to the past state, and it help to choose which states have more influence on the current state instead of simply selecting the most recent state. The parameters of each layer of the LSTM architecture is displayed in Tab. 6 and Fig. 4.

In this paper, we use a nonlinear SVM using RBF as the traditional machine learning model and the one-dimensional CNN architecture and LSTM as deep learning architectures, and then compare the performance of different classifiers through experiments.

4 Experimental Results

4.1 Dataset Description

The important basic part of this paper is data collection. The authenticity and reliability of the collected data directly determine the effectiveness of the model. We extracted malicious traffic from MCFP dataset. Additionally we use Wireshark to capture benign traffic. After pre-processing and clustering, 25397 valid data are finally obtained. Our dataset has a total of 11136 benign samples, 14261 malicious samples, which are shown in Tab. 3.

In this paper, the dataset is divided into training set, test set and validation set according to 6:2:2, that is, 60% of the malicious and benign samples are used for training the model, 20% for testing, and 20% for validation. For the sake of generality, this paper randomly selects 60% of all types of malicious samples, and mixes them with the 60% randomly selected from benign samples as a labeled training set. The

advantage of it is to ensure that each type of features can be learned by the model. Additionally, a stratified split of training, test and validation set across 5 folds is performed on the dataset to maintain class balance.

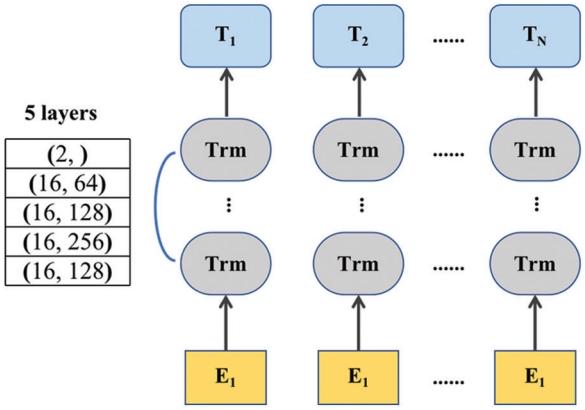

Figure 4: LSTM architecture

Table 3: Experimental dataset

Label	Training set	Test set	Validation set
Benign (y = 1)	6486	2325	2325
Malicious (y = 0)	10751	1755	1755

4.2 Feature Abstraction

In this paper, three different evaluation methods, including Fisher Score [24], Select K Best [25], and Random Forest [26], are used for feature selection. We first use these evaluation methods to score the features and sort them from largest to smallest. Then sort the features according to different numbers. Finally use the machine learning model to predict the evaluation results. The results are shown in Figs. 5a–5c.

According to the Fig. 5, the accuracy of the random forest began to converge on 16 features, and the other two methods began to converge on 20 features. Finally, the features intersection of the three methods are selected and 16 features are determined. The final selected features are as shown in Tab. 4.

4.3 Experimental Design

Finally, the data model is a value matrix, in which each row is identified by ID in the connected 4-tuple, and the column is the feature value. Each feature can range from 0 to 1, or the value is −1. In the data model, 60% is used as training data, 20% is used as test data, and the remaining 20% is used as verification data. The accuracy of the model is calculated according to the following formula.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

where TP, True Positive, means that the detection result is accurate and all are malicious; FP, False Positive, means that the detection result is wrong, and the model training is labeled as malicious but is actually benign HTTPs traffic; TN, True Negative, means that the detection result is correct, and the actual and predicted values are benign HTTPs traffic; FN, False Negative, means that the predicted value is benign, but it is actually malicious.

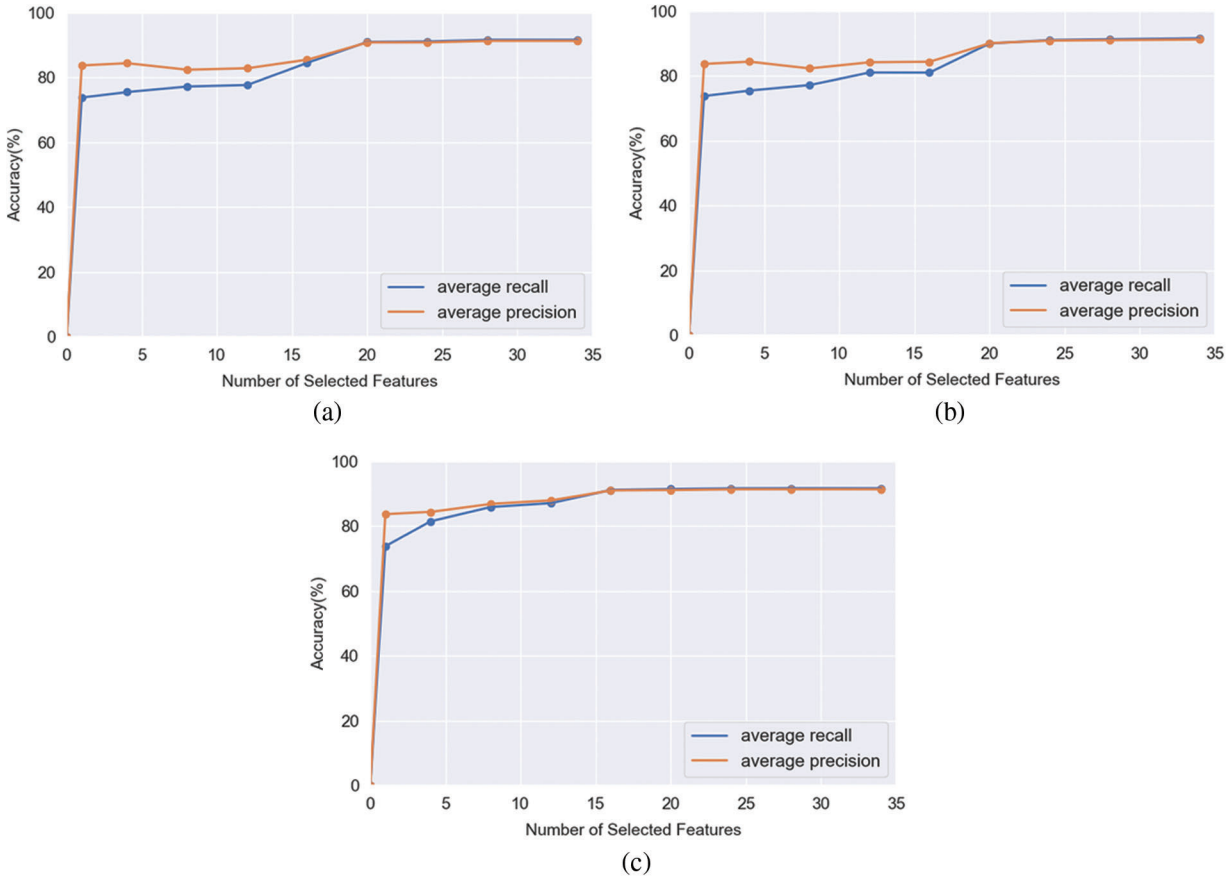

Figure 5: Three different evaluation methods used for feature selection. (a) Fisher Score. (b) Select K Best. (c) Random Forest

Table 4: The final selected features

Feature	Score	Feature	Score
Average age of the certificate	0.4144	Maximum number of originator packet	0.0183
Number of DNS in the certificate	0.1522	Age standard deviation of the certificate	0.0159
Number of Standard deviation originator bytes	0.0808	SNI is IP	0.0131
Mean of connection duration	0.0727	Standard deviation of the duration range	0.0115
Number of SSL aggregation and connection records	0.05	Maximum number of responder data packets	0.0108
Maximum number of originator bytes	0.0317	Number of payload bytes from responder	0.0097
Maximum number of responder bytes	0.0292	Average number of originator bytes	0.0089
Average number of responder bytes	0.0216	Standard deviation of connection duration	0.0088

Table 5: The parameters of each layer of the CNN architecture

Layer	Units	Output dimension	Parameter numbers
Input	1	(16,1)	0
Conv1d	10	(16, 10)	310
Conv1d	8	(16, 8)	2408
Conv1d	6	(16, 6)	1926
Conv1d	5	(16, 5)	1505
Conv1d	5	(16, 5)	1255
Flatten	80	(80,)	0
Dense	1024	(1024,)	82944
Dense	2	(2,)	2050

Table 6: The parameters of each layer of the LSTM architecture

Layer	Units	Output dimension	Parameter numbers
Input	1	(16,1)	0
LSTM	128	(16, 128)	66560
LSTM	256	(16, 256)	394240
LSTM	128	(16, 128)	197120
LSTM	64	(16, 64)	49408
LSTM	2	(2,)	130

Malicious HTTPs traffic recognition rate:

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

Malicious HTTPs traffic false positive rate:

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

4.4 Detection Performance

This experiment explores the performance comparison between traditional machine learning (SVM) and second another two deep learning architectures (CNN, LSTM) under the data pre-processing method (optimal parameters). The parameters of each layer of the one-dimensional CNN architecture and LSTM are displayed in Tabs. 5 and 6 respectively. In order to evaluate the performance of the models, we used a 5-fold cross-validation strategy. The experimental results are shown in Tab. 7.

Finally, we compared the results of the two deep neural networks to those obtained using SVM algorithm. The main difference between these algorithms is that the deep neural network has a long- and short-term storage layer, so it can use the text information contained in the certificate subject and issuer in a more effective way than traditional machine algorithms (such as SVM). In Tab. 7, the accuracy of the

deep neural network in the malicious traffic detection is 1.2% higher than that of the SVM model. In addition, improvements in results have also been observed in recall, accuracy, and F1-Score statistics.

Table 7: The performamce of different CLASSIFIER with TLSMell model

model	TPR	FPR	Recall	Precision	F1	Accuracy
SVM	89.85	9.24	89.85	88.10	89.32	90.37
CNN	88.61	5.14	88.61	92.93	90.71	92.16
LSTM	91.21	4.58	91.21	93.82	92.50	93.60

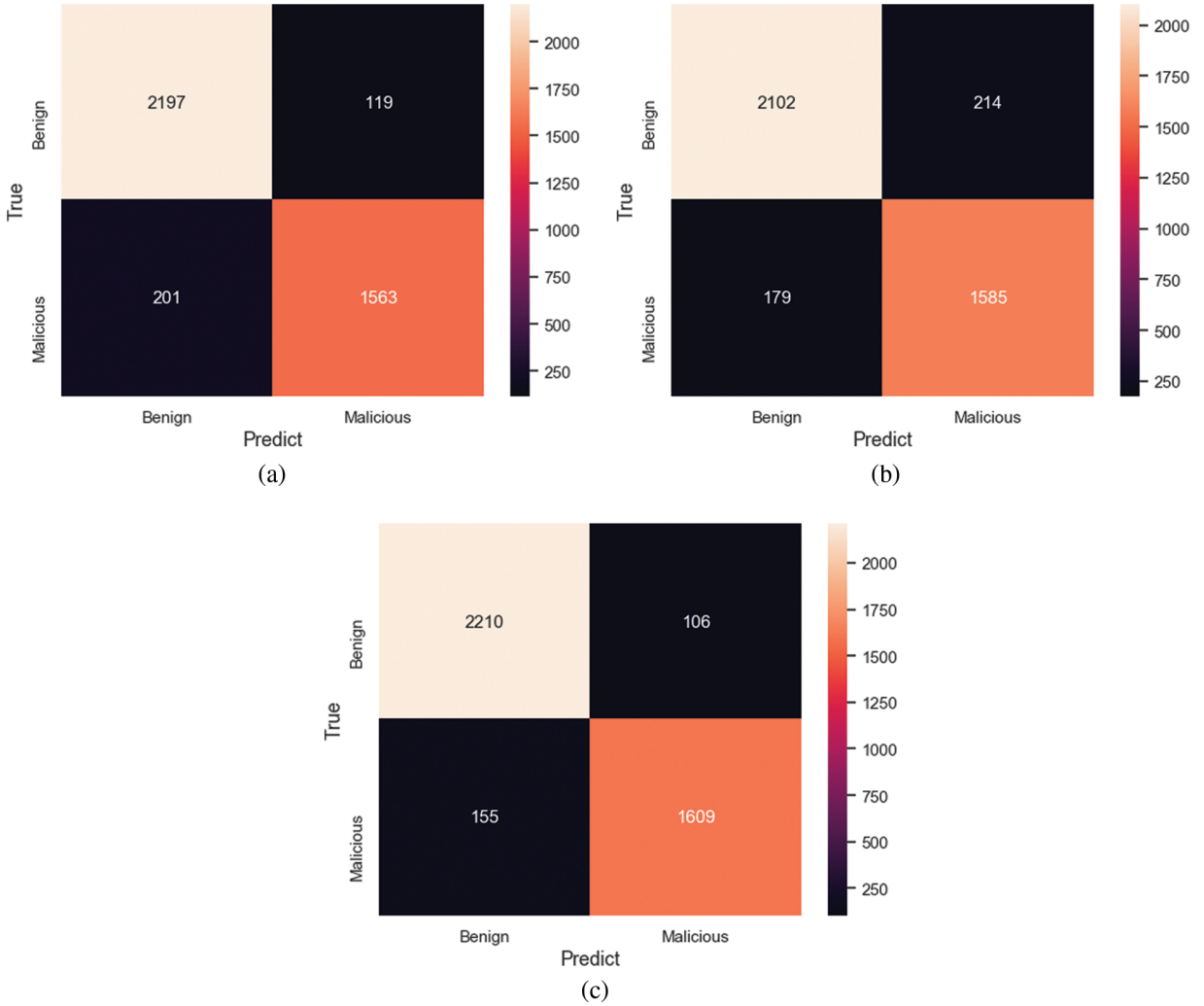

Figure 6: Confusion matrix of different types of classifiers. (a) CNN confusion matrix. (b) SVM confusion matrix. (c) LSTM confusion matrix

We not only present our results in terms of classification accuracy, but also as a confusion matrix showing the true positives and false positives broken down per-models. This was done to illustrate that

we were not simply using a naive majority-class classifier, but were in fact making useful inferences. As shown in Fig. 8, all three classifiers have high false positive rates. It can also be seen from another angle that the features we extracted can distinguish benign and malicious traffic well.

Through the comparison of the different models above (Figs. 6–8; Tab. 7), researchers can adopt different machine learning classification models according to the actual application needs. The results also verify the generalization of the feature extraction method proposed in this paper.

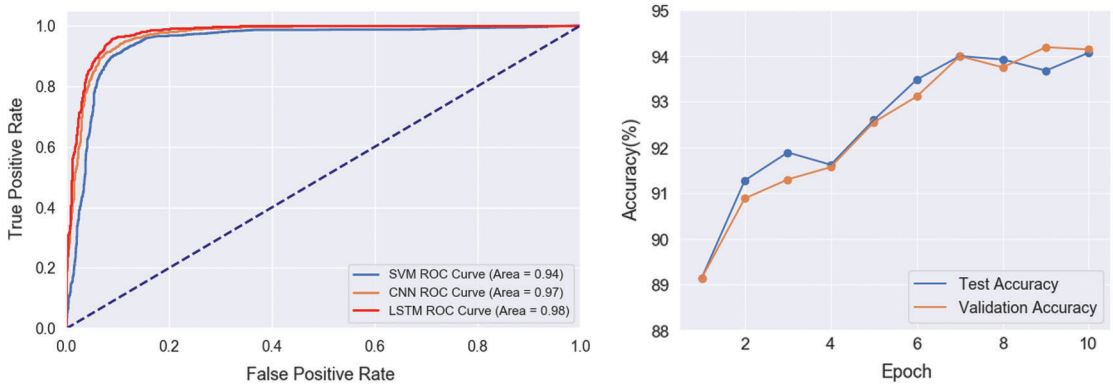

Figure 7: The comparison of the different models. (a) ROC curve of test set with different classifiers. (b) Comparison of test and validation accuracy

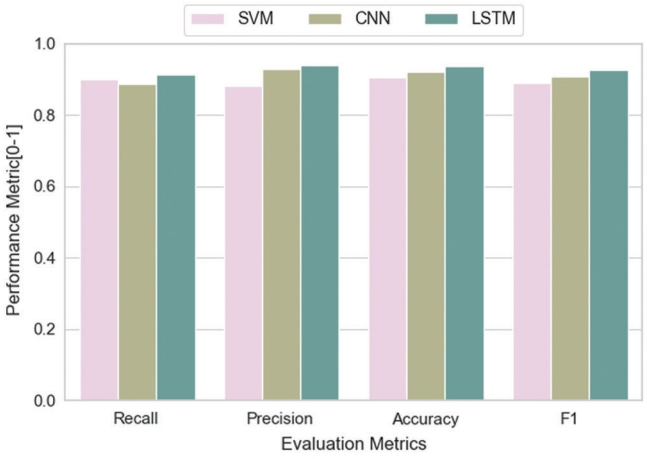

Figure 8: Performance of different malicious traffic classifiers

5 Conclusion

The method proposed in this paper does not need to decrypt HTTPs traffic, and has better support for real-time malicious traffic detection with high accuracy and efficiency. By using connection-specific indicators (4-tuple) and three different feature selection methods, the most representative features are analyzed for model training, testing and validation. And the online learning method can quickly update the model in real-time based on the online feedback data, and improve the prediction accuracy.

Through the comparison of different malicious traffic classifiers, researchers can adopt different DL classification models according to the actual application needs, and also verify the generalization of the method proposed in this paper.

Later research can further reduce the size of the data set and ensure the effectiveness of model detection. In the future, we will further optimize the connection-specific indicators and apply it to new network malicious traffic detection such as IOT and industrial Internet.

Funding Statement: This work is supported in part by the following grants: Wenzhou key scientific and technological projects (No.ZG2020031); Researchers Supporting Project of King Saud University, Riyadh, Saudi Arabia (No.RSP-2020/102); National Natural Science Foundation of China under Grant (No.U1936215 and 61772026); Ministry of Industry and Information Technology of the People's Republic of China under Grant (No.TC190H3WN); State Grid Corporation of China under Grant (No.5211XT19006B); Wenzhou Polytechnic research projects (No.WZY2020001); 2020 industrial Internet innovation and development project (TC200H01V); Wenzhou Scientific Research Projects for Underdeveloped Areas (WenRenSheFa [2020] 61(No.5)).; Zhejiang key R & D projects (No.2021C01117).

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] I. Torroledo, L. D. Camacho and A. C. Bahnsen, "Hunting malicious TLS certificates with deep neural networks," in *AISec'18*, Toronto, ON, Canada, pp. 64–73, 2018.

[2] C. Xiong, T. Zhu, W. Dong, L. Ruan and R. Yang, "CONAN: A practical real-time APT detection system with high accuracy and efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, pp. 1, 2020.

[3] T. Zhu, Z. Weng, L. Fu and L. Ruan, "A web shell detection method based on multiview feature fusion," *Applied Sciences*, vol. 10, no. 18, pp. 6274, 2020.

[4] J. Zhang, S. Zhong and J. Wang, "A storage optimization scheme for blockchain transaction databases," *Computer System Science and Engineering*, 2020.

[5] J. Wang, Y. Yang, T. Wang, R. S. Sherratt and J. Zhang, "Big data service architecture: A survey," *Journal of Internet Technology*, vol. 21, pp. 393–405, 2020.

[6] J. Wang, W. Wu, Z. Liao, R. S. Sherratt and A. Tolba, "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, no. 1, pp. 39758–39767, 2020.

[7] J. Wang, Y. Tang, S. He, C. Zhao and A. Tolba, "Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in Internet of things," *Sensors*, vol. 20, no. 9, pp. 2451, 2020.

[8] W. Li, H. Xu, H. Li, Y. Yang and J. Wang, "Complexity and algorithms for superposed data uploading problem in networks with smart devices," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5882–5891, 2020.

[9] J. Zhang, S. Zhong, T. Wang, H. Chao and J. Wang, "Blockchain-based systems and applications: A survey," *Journal of Internet Technology*, vol. 21, no. 1, pp. 1–14, 2020.

[10] K. Wang, J. J. Parekh and S. J. Stolfo, "Anagram: Acontent anomaly detector resistant to mimicry attack," in *Proc. Int. Workshop on Recent Advances in Intrusion Detection Springer*, Berlin, Heidelberg, vol. 4219, pp. 226–248, 2006.

[11] J. Lokoč, J. Kohout, P. Čech, T. Skopal and T. Pevný, "k-NN classification of malware in HTTPS traffic using the metric space approach," in *Proc. Pacific-Asia Workshop on Intelligence and Security Informatics*, Springer International Publishing, vol. 9650, pp. 131–145, 2016.

[12] Č. Přemysl, J. Kohout, J. Lokoč, T. Komárek and T. Pevn, "Feature extraction and malware detection on large HTTPS data using MapReduce," in *Proc. Int. Conf. on Similarity Search and Applications*, Tokyo, Japan, pp. 311–324, 2016.

[13] Z. Liu and Y. Lai, "A data mining framework for building intrusion detection models based on IPv6," in *Proc. the 3rd International Conference and Workshops on Advances in Information Security and Assurance (ISA '09)*, Berlin, Heidelberg: Springer-Verlag, pp. 608–618, 2009.

[14] K. C. Claffy, G. C. Polyzos and H. Braun, "Application of sampling methodologies to network traffic characterization," in *Proc. Communications Architectures, Protocols and Applications (SIGCOMM '93)*, New York, NY, USA: Association for Computing Machinery, pp. 194–203, 1993.

[15] A. Saber, B. Fergani and M. Abbas, "Encrypted traffic classification: Combining over-and under-sampling through a PCA-SVM," in *Proc. 3rd Int. Conf. on Pattern Analysis and Intelligent Systems (PAIS)*, Tebessa, pp. 1–5, 2018.

[16] L. Su, Y. Yao, N. Li, J. Liu and B. Liu, "Hierarchical clustering based network traffic data reduction for improving Suspicious Flow Detection," in *Proc. 17th IEEE Int. Conf. On Trust, Security And Privacy In Computing and Communications/ 12th IEEE Int. Conf. On Big Data Science And Engineering (TrustCom/BigDataSE)*, New York, NY, pp. 744–753, 2018.

[17] P. Prasse, L. Machlica, T. Pevny, J. Havelka and T. Sceffer, "Malware detection by analysing network traffic with neural networks," in *Proc. the 2017 IEEE Security and Privacy Workshops (SPW)*, San Jose, CA, USA, pp. 205–210, 2017.

[18] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. the 2016 ACM Workshop on Artificial Intelligence and Security (AISec '16)*, New York, NY, USA: Association for Computing Machinery, pp. 35–46, 2016.

[19] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-Stationarity," in *Proc. the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '17)*, New York, NY, USA: Association for Computing Machinery, pp. 1723–1732, 2017.

[20] L. Zhao, L. Cai, A. Yu, Z. Xu, D. Meng *et al.,* "Prototype-based malware traffic classification with novelty detection," in *Proc. ICICS 2019: Information and Communications Security*, pp. 3–17, 2020.

[21] E. Meng, S. Huang, Q. Huang, W. Fang and L. Wu, "A robust method for non-stationary streamflow prediction based on improved EMD-SVM model," *Journal of Hydrology*, vol. 568, pp. 462–478, 2019.

[22] R. Girshick, J. Donahue, T. Darrel and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. 2014 IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, pp. 580–587, 2014.

[23] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[24] T. Zhu, Z. Qu, H. Xu, J. Zhang and Z. Shao, "RiskCog: Unobtrusive real-time user authentication on mobile devices in the wild," *IEEE Transactions on Mobile Computing*, vol. 99, pp. 1, 2019.

[25] A. Cohen, W. Dahmen and R. A. DeVore, "Compressed sensing and best k-term approximation," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 211–231, 2009.

[26] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.