

A Method Based on Hierarchical Spatiotemporal Features for Trojan Traffic Detection

Jiang Xie^{*†}, Shuhao Li^{*†}, Yongzheng Zhang^{*†‡}, Xiaochun Yun[§], Jia Li^{*†}

^{*}Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

[†] Key Laboratory of Network Assessment Technology, IIE, CAS, Beijing, China

[‡]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[§]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China

Email: {xiejiang,lishuhao,zhangyongzheng}@iie.ac.cn; {yunxiaochun,lijia}@cert.org.cn

Abstract—Trojans are one of the most threatening network attacks currently. HTTP-based Trojan, in particular, accounts for a considerable proportion of them. Moreover, as the network environment becomes more complex, HTTP-based Trojan is more concealed than others. At present, many intrusion detection systems (IDSs) are increasingly difficult to effectively detect such Trojan traffic due to the inherent shortcomings of the methods used and the backwardness of training data. Classical anomaly detection and traditional machine learning-based (TML-based) anomaly detection are highly dependent on expert knowledge to extract features artificially, which is difficult to implement in HTTP-based Trojan traffic detection. Deep learning-based (DL-based) anomaly detection has been locally applied to IDSs, but it cannot be transplanted to HTTP-based Trojan traffic detection directly. To solve this problem, in this paper, we propose a neural network detection model (HSTF-Model) based on hierarchical spatiotemporal features of traffic. Meanwhile, we combine deep learning algorithms with expert knowledge through feature encoders and statistical characteristics to improve the self-learning ability of the model. Experiments indicate that F_1 of HSTF-Model can reach 99.4% in real traffic. In addition, we present a dataset BHTT consisting of HTTP-based benign and Trojan traffic to facilitate related research in the field.

Index Terms—Trojans, CNN, LSTM, statistical characteristics, traffic detection

I. INTRODUCTION

The rapid development of the Internet has brought great convenience. However, more comprehensive service means that traffic becomes more complex, which poses greater challenges to network security. Trojans are one of the main malicious attacks in online activities. They are a class of malicious programs that attack hosts/websites to steal personal information and even remotely control devices. In 2017, a total of 19,017,282 hosts with IP addresses were implanted with Trojans. And monthly, almost 2.81 million host IP addresses in the global Internet were infected with "Flying" worm Trojan in average [1]. These Trojans spread and enforce attacks through network traffic, where HTTP traffic is one of the main carriers. Therefore, it is necessary to find an effective way to detect HTTP-based Trojan traffic.

At present, it is a highly concerned issue about ensuring the security of network devices and information. People usually

build intrusion detection systems (IDSs) to resist various network attacks. The anomaly detection, which is a primary research direction in the field of intrusion detection, can detect new and unknown attacks (0-day) by analyzing benign and malicious behavior characteristics from traffic [2]. However, most of the current anomaly detection methods, including machine learning-based methods, are highly dependent on feature engineering to achieve high detection performance. And designing a satisfactory feature engineering that can accurately extract traffic characteristics is a hard work [3]. Therefore, many anomaly detection methods are difficult to apply in practice [4].

In recent years, deep learning has attracted great attention in various fields. And many researchers have proposed anomaly detection methods based on deep learning to automatically extract features [5], [6]. After inputting low-order data features, neural networks can correct model parameters to find complex structures from data, and then abstract them into high-order data to realize automatic learning. Deep learning reduces the dependence of model on feature engineering, which allows researchers to design excellent anomaly detection models through simple data preprocessing.

In this paper, to effectively detect HTTP-based Trojan traffic and reduce the dependence on feature engineering, we present an anomaly detection method based on deep learning algorithms. Then, a statistical feature set based on experience is added to improve the self-learning ability of the model.

In general, the contributions of this paper are as follows.

- We combine raw traffic with statistical characteristics to enrich feature representation information. Then, feature encoders are built to layer the traffic into packet-level and flow-level for preprocessing. Experiments indicate that this preprocessing method can effectively improve the performance of the model.
- We propose HSTF-Model (Model based on Hierarchical SpatioTemporal traffic Features) consisting of CNN and LSTM. HSTF-Model extracts hierarchical spatiotemporal sequence features from raw data and statistical characteristics. We can adjust the weight of the two to determine which one is more reliable.

- We generate a dataset BTHT-R (Benign and Trojan Traffic based on Http-Raw) consisting of raw HTTP traffic. BTHT-R includes benign traffic (4,044,751 flows) and HTTP-based Trojan traffic (37,847 flows). In addition, a corresponding statistical feature dataset BTHT-S (Statistical) is also built. BTHT-R and BTHT-S form dataset BTHT¹. Because of privacy, we use irreversible hash technology for data masking in BTHT.

The remainder of this paper is organized as follows. Related works is described in Section II, and the introduction of dataset BTHT in Section III. Section IV introduces the methodology of HSTF-Model. We conduct experiments in Section V. Subsequently, we discuss the model and experiments in Section VI. Section VII draws our conclusion and gives future challenges.

II. RELATED WORK

The research in this paper belongs to malicious traffic detection, which is one of the cores of intrusion detection. In general, there are two types of approaches for IDSs based on traffic detection, signature detection and anomaly detection [7]. The former, also called misuse detection, mainly analyzes the characteristics and behavior patterns of known attacks. The greater the similarity of network behavior in actual detection, the more likely that it is to be judged malicious. Signature detection is effective in detecting known attacks with low errors. But it cannot detect new and unknown attacks (0-day). The latter is also known as behavior detection. Anomaly detection can detect unknown attacks. This makes anomaly detection a major research direction in intrusion detection [8]. Current mainstream anomaly detection methods include classical anomaly detection, TML-based anomaly detection, DL-based anomaly detection and so on.

A. Classical Anomaly Detection

The classical anomaly detection profiles benign traffic patterns [7]. It is based on the hypothesis that an attacker behavior differs to that of a benign user. Benign operations of the members are profiled and a certain amount of deviation from the benign behavior is flagged as an anomaly [9].

Classical anomaly detection can be useful for new attack patterns, but it is not as effective as signature detection when detecting known attacks [10]. In addition, its performance is highly dependent on feature engineering. If the feature analysis for benign traffic is incomplete, this approach will have a high false-positive rate [7]. Therefore, more research is attempting to combine classical anomaly detection and signature detection currently, that is, analyzing both benign and malicious traffic to improve the performance of model. Because this hybrid intrusion detection can detect unknown attacks, it is also commonly considered as anomaly detection [8].

B. TML-based Anomaly Detection

There are many traditional machine learning-based (TML-based) anomaly detection methods applied to IDSs [11], for instance, Markov-based anomaly detection [12]. These TML-based intrusion detection methods simultaneously learn the characteristics of benign and malicious traffic.

Lin *et al.* [13] presented CANN, a feature representation approach that combines cluster centers and nearest neighbors for intrusion detection. The experimental results based on the dataset KDD-Cup'99 [14] can reach 99.9% in accuracy. Aljawarneh *et al.* [15] proposed a hybrid model based on the optimal characteristics of network traffic. The hybrid algorithm is composed of J48, Meta Paggging, and other classical machine learning algorithms. The accuracy of the model can be 99.81% (binary-class) and 98.56% (multiple-class) in dataset NSL-KDD [16], respectively. Chen *et al.* [17] proposed S-IDGC based on machine learning to classify imbalanced traffic data for mobile malware detection. Gezer *et al.* [18] utilize machine learning techniques to detect TrickBot malware infections.

In general, most TML-based anomaly detection methods also need to design feature engineering. Some traffic features are designed first. Then, a model is built based on those features using supervised or unsupervised learning algorithms. But designing a feature set that can accurately characterize network traffic is still an ongoing research issue [3].

C. DL-based Anomaly Detection

At present, deep learning (DL) have received great attention in various fields, including cyber security [19]. Multiple processing layers of the neural network can learn the abstract representation of data. Neural network extracts low-order features directly from the raw data, and then, combines and transforms them into high-order features for automatic learning and analysis. Therefore, DL-based anomaly detection reduces the need for researchers to spend more time designing complex feature engineering [20].

Shone *et al.* [6] presented a novel DL technique using stacked NDAEs for intrusion detection, which is a nonsymmetric deep autoencoder for unsupervised feature learning. Accuracy can reach 97.85% in KDD-Cup'99 and 89.22% in NSL-KDD. CNN and RNN are two widely used neural network models. CNN has attracted much attention for its spatial feature extraction ability. In cyber security, Vinayakumar *et al.* [21] used CNNs for network intrusion detection. Li *et al.* [22] used CNN for representation learning for intrusion detection. Yin *et al.* [5] proposed RNN-IDS for intrusion detection, and compared it with classical machine learning algorithms. LSTM, another well-known model as one of a variant of RNN, has natural advantages in processing data with sequences and dependencies. Kim *et al.* [23] applied LSTM for intrusion detection, which can achieve accuracy 96.93%.

On the basis of existing research, there are the following main problems that can be improved: 1) The classic datasets for benchmarking, such as KDD-Cup'99, are outdated. The actual network environment is more complex nowadays. 2) There is no specific research for the detection of the HTTP-based

¹The dataset can be found at https://drive.google.com/open?id=1d_SVIOzzgw2kYPIC5dKjgOI51YXTDZUi. Researchers who are going to use the dataset should indicate the original source of data by citing this paper.

Trojan traffic attack scenario. 3) Classical and TML-based anomaly detection methods are mostly dependent on well-designed feature engineering. DL-based anomaly detection methods can extract features automatically, but most of those methods currently learn directly from the original data, without making use of artificial accumulated feature experience. Our method belongs to DL-based anomaly detection. Meanwhile, we add statistical characteristics based on experience and data analysis to get a more efficient neural network model.

III. FEATURE SET CONSTRUCTION

In this paper, we generate dataset BTHT (Benign and Trojan traffic based on Http) including BTHT-R (Raw) and BTHT-S (Statistical). BTHT-R consists of benign and malicious traffic from the laboratory gateway and traffic interfaces of CNCERT/CC². For each flow in BTHT-R, statistical characteristics are made to construct dataset BTHT-S.

All the data used in experiments is from BTHT. We are also set up multiple proportions of benign:malicious data to evaluate the performance of HSTF-Model comprehensively.

A. Data Acquisition

Dataset BTHT-R is generated by capturing and filtering traffic from the real network. There are about 4 million flows in BTHT-R, 99% of which come from benign behavior and 1% come from malicious behavior (various Trojans).

For most benign traffic, after being authorized, we collected about 300GB traffic from the laboratory gateway under the premise of protecting privacy. After cleaning out irrelevant/redundant information in traffic, we obtained 4,044,751 million benign network flows eventually, including news browsing, social chatting, web browsing and so on.

HTTP-based Trojan traffic is provided by CNCERT/CC. There are two sources of HTTP-based Trojan traffic. One is to analyze the real-time traffic in the network through the existing monitoring system of CNCERT/CC. The other is to deploy honeypots in the network for malicious sample capture and breeding to generate HTTP-based Trojan traffic. After analysis and processing, we obtained 37,847 valid malicious flows based on HTTP by manual labeling. These malicious traffic types include malicious promotion, malicious download, Trojans implant and so on. Because of data security protection, the sensitive fields (host, etc.) of traffic we obtained were subjected to irreversible hash processing. To maintain consistency, benign traffic is handled in the same way.

```
GET /?mrow_cntrl/?id&rnd=***** HTTP/1.1
User-Agent: ***** exp
Host: www.*****.info
Cache-Control: no-cache
```

Fig. 1. Online package generated by a Trojan intrusion.

An online package generated by a Trojan intrusion behavior is shown visually in Fig. 1. The relevant sensitive data is

²CNCERT/CC is the abbreviation of “National Computer Network emergency Response technical Team/Coordination Center of China”. The web site of CNCERT/CC is <http://www.cert.org.cn/>.

replaced with ‘*’ to fully protect the privacy and data security. In addition, data size in BTHT-R is shown in Tab I.

TABLE I
STATISTICS ON PACKET SIZE (IN BYTES) AND FLOW SIZE (IN PACKETS) IN BTHT-R

Statistics	Packet	Flow
Count	14,892,047	4,082,598
Size	12,597,103,235	14,892,047
Mean	845.894	3.648
Min	12	1
Max	46,729	16,819

B. Feature Set Construction

We are committed to building a feature set based on the statistical characteristics of traffic and adding it to neural network to improve the performance. The URL length of malicious flow, for instance, is generally longer than benign. Experiments indicate that these statistical characteristics are useful for traffic identification.

A flow is represented with corresponding statistical characteristics in dataset BTHT-S. In addition, a flow consists of multiple packets, and a packet consists of different fields. This means that flow can be layered. We show the hierarchical structure in Fig. 2.

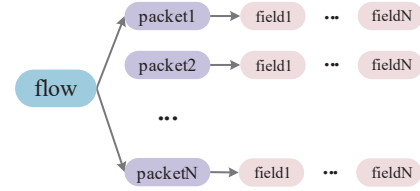


Fig. 2. Hierarchical structure in flow.

Traffic in dataset BTHT exhibits this hierarchical timing relationship (such as Fig. 1). Therefore, we extract features from two statistical levels and form two vectors, packet-level vector (PL) and flow-level vector (FL).

The first is PL . Statistical characteristics of packets are extracted according to items shown in Tab II. RFC1998 [24] proposed 47 header fields and suggested that HTTP-based web services use those header fields. Actually, most web services use only a few major header fields. We default that no more than 47 header fields in packets. A packet will generate a extensible vector $PL(1 \times 100)$. The composition and dimension can be adjusted flexibly according to different data Features.

TABLE II
COMPOSITION OF THE PACKET-LEVEL VECTOR(PL)

Item Type	Pos
request type	1
source port	2
destination port	3
URL length	4
HTTP protocol version	5
field name length	6-52
field value length	53-99
payload	100

The second is *FL*. A flow consists of multiple packets, just as a sentence consists of multiple words. Sometimes, the information provided by a single word is no-valuable and combination of multiple words can show specific content. Similarly, information based on flow level can reflect traffic behavior better. Therefore, we extract statistical characteristics of flows to form vector *FL*. By analyzing the dataset BTHT-R, we default that the number of packets in a flow will not exceed 50. It will be truncated if it is exceeding and filled with 0 if it is insufficient. Tab III details items of *FL*. A flow generates a statistical characteristics extensible vector $FL(1 \times 170)$.

TABLE III
COMPOSITION OF FLOW-LEVEL VECTOR(FL)

Item Type	Pos
number of packets	1
proportion of request packets	2
proportion of response packets	3
proportion of the same packets in the request packets	4
proportion of different packets in the request packets	5
proportion of the same packets in the response packets	6
proportion of different packets in the response packets	7
TTL value of packets	8-57
interval between acquisitions of adjacent messages	58-106
HTTP packets length	107
proportion of request length	108
proportion of response length	109
length of each packet	110-159
multiple requests corresponding to one response?	160
multiple responses corresponding to one request?	161
'get' proportion in the request	162
'post' proportion in the request	163
'head' proportion in the request	164
other proportions in the request	165
2XX proportion in response	166
4XX proportion in response	167
5XX proportion in response	168
other proportions in response	169
HTTP packets accounted for in flow	170

The *PL* and *FL* of a flow together constitute a sample in dataset BTHT-S as statistical characteristics of the corresponding raw flow. This feature set can provide more information for the neural network as experience knowledge.

IV. HSTF-MODEL METHODOLOGY

A. Overview

We propose the HSTF-Model (Model based on Hierarchical SpatioTemporal Traffic Features). With a preprocessed HTTP-based flow as input, HSTF-Model can extract features automatically, and judge whether it is malicious or benign.

Combining the layered characteristics of raw flows, HSTF-Model processes data. At packet level, after the feature encoder of raw data outputs the feature matrix, HSTF-Model uses CNN to extract spatial and character features directly. At flow level, HSTF-Model extracts temporal features between packets further by using LSTM. Then, model synthesizes all the feature information through hidden layers, and outputs the discriminant result finally.

In addition, HSTF-Model is improved by adding statistical feature set. The *PL* statistics is added at the packet level and

combined with the output of CNN. Then, the *FL* statistics is add at the flow level and combine it with the output of LSTM.

The overall structure of HSTF-Model is shown in Fig. 5. Through preliminary experiments and empirical knowledge, we determined the overall structure (the number of neurons in each layer of neural network, the output size of feature encoders of *PL* and *FL*, etc.)

The various parts of HSTF-Model are detailed below.

B. Data Preprocessing

We build feature encoders to preprocess data. The first one is the raw data. As shown in Fig. 3, the feature encoder of raw data converts each field line in the packet into a 1×200 vector. As a rule of thumb, we default that no more than 47 header fields in packets. Finally, a raw packet produces a 47×200 feature matrix. It will be truncated if it is exceeding and filled with 0 if it is insufficient. Then, we use 1:1 hash technology for data privacy protection and vectorization.

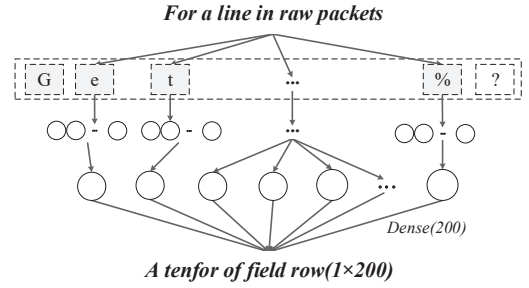


Fig. 3. Feature encoder of raw data.

The second is feature encoders of statistical data at packet-level and flow-level. As shown in Fig. 4, the feature encoder at packet-level converts *PL* to a 1×20 vector and combines it with the output of CNNs. Then, the feature encoder at flow-level converts *FL* to a 1×30 vector and combines it with the output of LSTM. Feature encoders convert statistical information into a more compact representation, which facilitates the combination with tensors in neural network.

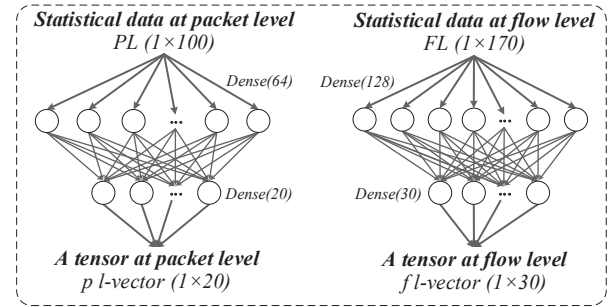


Fig. 4. Feature encoders of statistical data at packet-level/flow-level.

C. Learning of Packet Feature by CNN

CNN belongs to deep feedforward neural network. In 1980s, Fukushima *et al.* [25] realized it for the first time. CNN can extract hidden structural information. There are one or more convolution and pooling layers in CNN. The convolution layer contains multiple feature mapping neurons (convolution kernels), which separate input data into different feature regions,

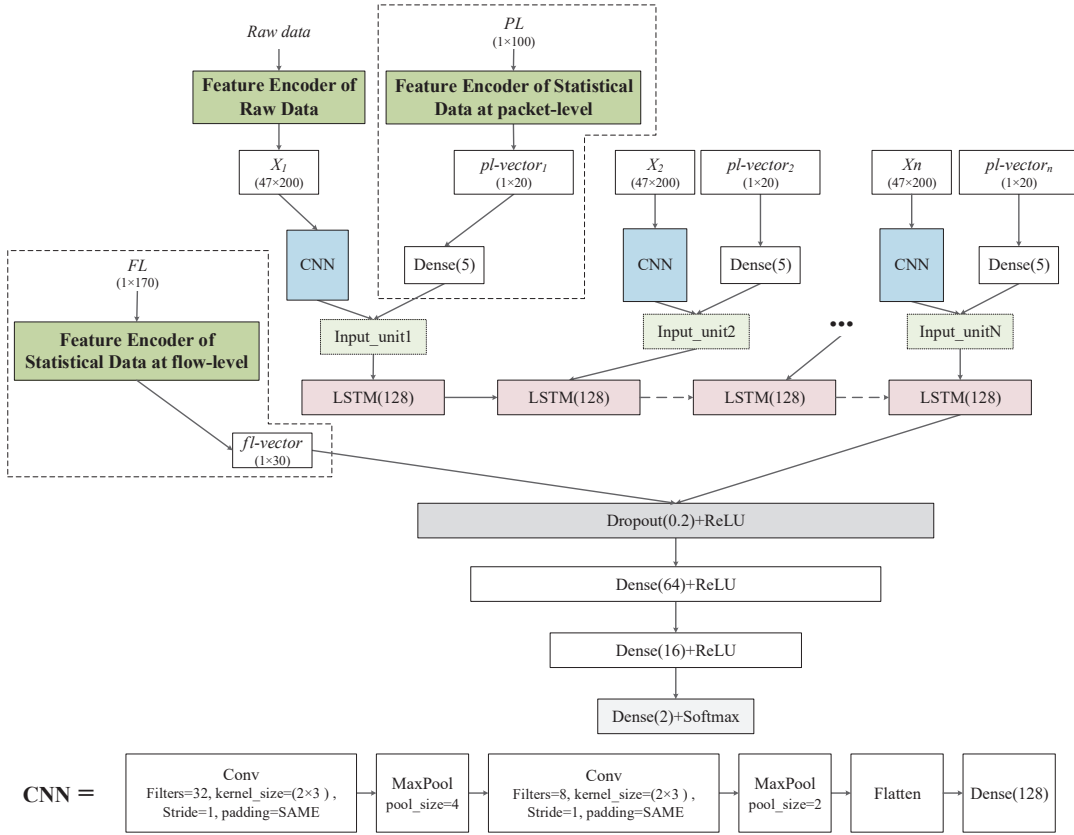


Fig. 5. Overall architecture of HSTF-Model.

and each convolution kernel is responsible for extracting local features. We can obtain global feature information of sample by aggregating local features. After convolution operation, the vector is pooled, which is a down-sampling method to reduce the complexity and over-fitting.

We process 47×200 feature matrix by CNN, as shown in Fig.5. ReLU is used as an activation function. After two convolutions and pooling operations, the output of CNN is processed by ReLU, and then, enters the follow-up dense layers containing 128 neurons. The output C_i is shown in Eq (1), where b is the bias term and f is the activation function. Then, multiple convolution kernels perform feature mapping and maximum pooling.

$$\begin{aligned} c_i &= f(w \cdot x_{i:i+h-1} + b) \\ \hat{c} &= \max(c = [c_1, c_2, \dots, c_{n-h+1}]) \end{aligned} \quad (1)$$

D. Learning of Flow Feature by LSTM

In 1997, LSTM was proposed by Hochreiter and Schmidhuber [26], which is a variant of RNN. RNN records the processed information previously and makes use of those information in current task. However, there are some disadvantages using RNN. One of them is the difficulty in addressing long-term dependency. It can only utilize the information that is not far from the current task. The LSTM optimizes for this problem by designing 'gate' structures to preserve and select information. Each gate consists of an activation layer and a pointwise operation.

The input gate, i_t , combines the input to determine the new information \tilde{C}_t be added. The forgotten gate, f_t , handles the previous status and determines the old information C_{t-1} be discarded. The two gates determine the proportions of new and old information in the current information C_t .

$$\begin{aligned} i_t &= \delta(W_i \cdot [h_{t-1}, x_t + b_i]) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ f_t &= \delta(W_f \cdot [h_{t-1}, x_t] + b_f) \\ C_t &= f_t \times C_{t-1} + i_t \times \tilde{C}_t \end{aligned} \quad (2)$$

The output gate, o_t , determines the output information of current neurons.

$$o_t = \delta(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3)$$

There is also a hidden layer status output h_t that is used to assist in the next task processing.

$$h_t = o_t \times \tanh(C_t) \quad (4)$$

CNNs extract different abstract features based on packets in a flow. These features are time-dependent due to the dependence of raw packets. Therefore, we use LSTM with 128 neurons per cell when processing flows in HSTF-Model.

E. Learning of Statistical Characteristics by DNN

DNN is one of the foundations of neural network. Rosenblatt proposed the perceptron model in 1958 [27], and then derived the multi-layer perceptron (MLP). MLP is also known

as simple DNN. In this paper, there is no obvious structural relationship and time dependence in statistical data. Therefore, we process the statistical characteristics by DNN.

A flow including n packets forms vector of $n \times 100 + 170$ size. Then, feature encoders convert PL and FL vector into denser representations through the full connection layers within DNN. At packet level, a dense layer containing 5 neurons further processes the output of feature encoder of PL for better stitching with the output of CNN. At flow level, the output of feature encoder of FL is combined with the output of LSTM to enter the follow-up network layer.

V. EXPERIMENT AND EVALUATION

A. Evaluation Metrics

Precision(P) and recall(R) are calculated to evaluate the performance of HSTF-Model. F_β by Eq (5) is also calculated as the comprehensive evaluation index. We can change the value of β to make the evaluation pay more attention to precision or recall. In this paper, we set $\beta = 1$, which means that precision and recall are equally important.

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R} \quad (5)$$

B. Configuration of Environment

HSTF-Model is implemented in Python3.5 based on the libraries of Keras and TensorFlow. The system environment of experiments is Ubuntu16.04 LTS. All software applications are deployed on a server machine with 64 CPU cores and 64GB memory. To further accelerate matrix computing, 8 NVIDIA GeForce GTX TITAN X are installed in the server.

For the selection of experimental data, we repeat the experiment 10 times and randomly select 70% malicious and partial benign flows from dataset BTHT for training at each time. The proportion of malicious:benign is 3:10. In subsequent experiments, the proportion would be changed if necessary. In addition, we select the remaining 30% malicious flows and 50,000 benign flows from dataset BTHT outside the training set to form the testing set for each experiment.

C. Efficiency of HSTF-Model with Statistical Characteristics

In this paper, statistical features containing empirical knowledge are added to the neural network processing as a supplement. These statistics can provide richer data representation and enhance the feature extraction capability of the model. We conduct experiments with different packet size (in bytes) and

flow size (in packets) combinations to verify the performance improvement from statistical characteristics.

We show some experimental results in Tab IV. Experiments indicated that the detection effect and robustness of HSTF-Model was generally better than the model without statistical characteristics. But the cost of detection is increased because of the need to extract additional statistical information. However, HSTF-Model is in the same order of magnitude as the complexity of the model without statistical characteristics. Therefore, the cost caused by statistical characteristics is acceptable compared to performance improvements.

In addition, we need to determine the appropriate packet size and flow size, which are not fixed in different data. The range of packet size and flow size is very wide. We cannot experimentally validate all size combinations. Therefore, the control variable method is used by follow-up experiments.

D. Influence of Packet Size and Flow Size

The appropriate packet size is determined in experiments. According to the statistical analysis of the dataset BTHT and previous experimental experience, we set *flow size* = 4. Fig. 6 details the results.

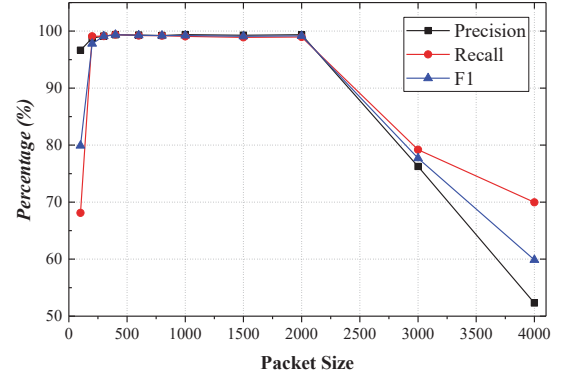


Fig. 6. Effect of packet size on HSTF-Model (flow size = 4).

We change the packet size from 100 to 4000 (size = 100, 200, 300, 400, 600, 800, 1000, 1500, 2000, 3000, 4000). HSTF-Model can achieve the best result when *packet size* = 400 according Fig. 6. When the size exceeds 2000, in the experiment, the detection effect begins to decrease. The reason is that most malicious samples in dataset BTHT come from an online package of Trojans, causing practical packet size is not large and the malicious features are basically hidden in preceding bytes. When the size enlarged, it is equivalent to

TABLE IV
THE PERFORMANCE COMPARISON OF HSTF-MODEL WITH AND WITHOUT STATISTICAL CHARACTERISTICS

Proportion malicious:benign	Packet size	Flow size	HSTF-Model without statistical characteristics				HSTF-Model			
			Precision	Recall	F_1	Time	Precision	Recall	F_1	Time
3:10	400	4	99.08	98.26	98.67	11.4s	99.35	99.4	99.37	11.6s
3:10	400	8	99.21	99.18	99.19	13.5	99.23	99.19	99.21	15.5s
1:4	400	4	99.26	96.74	97.99	11.5s	99.42	97.8	98.6	11.8s
1:4	400	8	99.3	97.15	98.21	13.6s	99.38	98.96	99.17	16s
1:8	800	8	99.65	95.43	97.49	18.4s	99.94	96.64	98.26	20.8s

adding noise, which causes the drop-in effect. Therefore, we choose 400 as the packet size for subsequent experiments.

After CNNs output abstract feature vectors, these vectors form time-series units, which are processed by LSTM. The experimental results of selecting flow size are shown in Fig. 7.

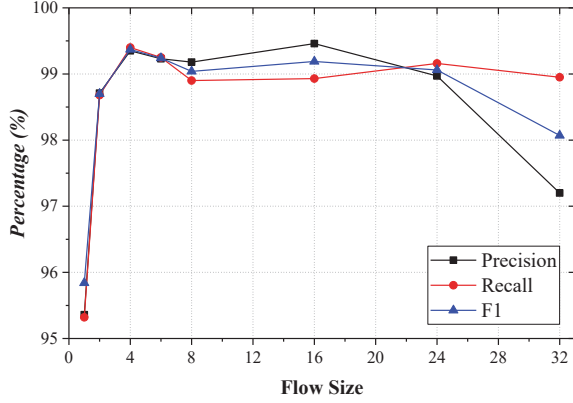


Fig. 7. Effect of flow size on HSTF-Model (packet size = 400).

We change the flow size from 1 to 32 (size = 1, 2, 4, 6, 8, 16, 24, 32). HSTF-Model can achieve the best results when $flow\ size = 4$. Trojans usually contact the attacker by sending some online packets, which are usually concentrated in the first few packets. Therefore, we only need the first few packets in a flow to determine whether it originated from malicious behavior. In this paper, 4 is chosen as the best flow size.

E. Efficiency of HSTF-Model in Imbalanced Data

To evaluate the performance of HSTF-Model in imbalanced data, we use different proportions of data in experiments. The results are shown in Tab V. With the increase in the proportion of benign samples in training, it is easier to extract features of benign data and pays more attention to it. In that case, the training model improves the discriminant threshold for malicious traffic and the precision of the model close to 100%. For instance, the precision exceeds 99.99% at proportions of 1:24 and 1:100. However, with the decrease of malicious data, it becomes more difficult to extract the characteristics of malicious samples, and the ability of HSTF-Model to identify malicious samples also declines. At 3:10, the recall reaches 99.4%, while at 1:100, the recall drops to 78.96%.

TABLE V
EFFICIENCY OF HSTF-MODEL AT DIFFERENT TRAINING PROPORTIONS

Proportion malicious:benign	Evaluation Index(%)		
	Precision	Recall	F_1
3:10	99.35	99.4	99.37
1:4	99.42	97.8	98.6
1:8	99.95	96.72	98.31
1:24	99.99	90.83	95.19
1:100	99.99	78.96	88.24

We show the recall rate convergence process of HSTF-Model at different proportions in Fig. 8. The performance is getting worse with the decrease of malicious data. But HSTF-Model can still be considered that has good robustness and can handle imbalanced data well. When malicious data only

accounts for only 4.17% in training, it can still converge and has the $F_1 = 95.19\%$.

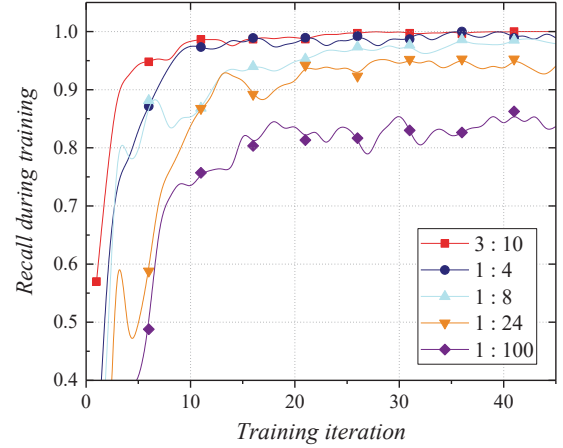


Fig. 8. The recall rate convergence curves of HSTF-Model at different training proportions.

F. Comparison with Other Methods

We implement several of the latest malicious traffic detection methods in combination with our own data analysis. And some classical machine learning algorithms (Bayes, SVM, Decision Tree) are also implemented, which are widely used in the field of malicious traffic detection. We compare HSTF-Model with these methods.

The experimental results are shown in Tab VI. HSTF-Model has the best comprehensive performance in detection effect and time cost. Rbf-SVM get the recall of only 74.49% and C4.5 get the precision of only 74.29%. Although Naive Bayesian costs the least time, its performance is worse than others because it gives up the association between packets. LSTM-R is fast but not excellent because there is no deep consideration of the structural relationship inside the packet. Then, the detection result of S-IDGC and Proposed-Hybrid-Model are excellent but slower than HSTF-Model. These two models are built using traditional machine learning methods, and it is difficult to improve performance with data iterations. HSTF-Model can be continuously updated on the basis of the original. Moreover, HSTF-Model is more robust than other methods when the data is imbalanced in dataset BTHT.

TABLE VI
HSTF-MODEL COMPARED TO OTHER METHODS IN DATASET BTHT

Method	Precision	Recall	F_1	Test Time
Rbf-SVM	100	74.49	85.38	34m12s
C4.5	74.29	98.66	84.76	1m28s
GaussianNB	99.36	52.25	68.49	5.7s
LSTM-R [28]	97.64	96.27	96.95	8.8s
S-IDGC [17]	99.14	99.28	99.21	3m20s
Proposed-Hybrid-Model [15]	99.45	97.67	98.55	8m17s
HSTF-Model	99.35	99.4	99.37	11.6s

In a word, HSTF-Model has the best comprehensive performance because neural networks have excellent self-learning ability. Meanwhile, statistical characteristics can improve the

richness of data, thus enhancing the feature extraction ability of the model and accelerating model convergence.

VI. DISCUSSION

In this paper, we designed the appropriate granularity and range of values to cover the most optimal solutions, and then, determined the optimal values of flow size and packet size by controlling variables. HSTF-Model relies mainly on training data to improve detection capabilities. When the HTTP-based Trojan traffic in the training data is sufficient, the model can still maintain accurate recognition of such Trojan traffic in more complicated traffic environments.

Although HSTF-Model performs well in dataset BTHT, there are still some shortcomings. The first is about false-positives. When a flow is small, there are few features that can be extracted. HSTF-Model is difficult to detect it effectively. The second is that generalization is not fully verified, and the performance of the model is slightly dithered in different situations. For instance, the precision in the real-time Internet will result in a decrease of 2%-5% due to the diversity of traffic. Our next work is to alleviate this problem.

VII. CONCLUSION

In this paper, we build features encoders and an effective prototype detection method HSTF-Model. Deep learning is combined with statistical characteristics. It is used for HTTP-based Trojan traffic detection and can reach to 99.4% in recall (99.35% in precision) in experiments. In addition, we provide a dataset BTHT consisting of benign traffic and malicious traffic for experiments and other research in the field.

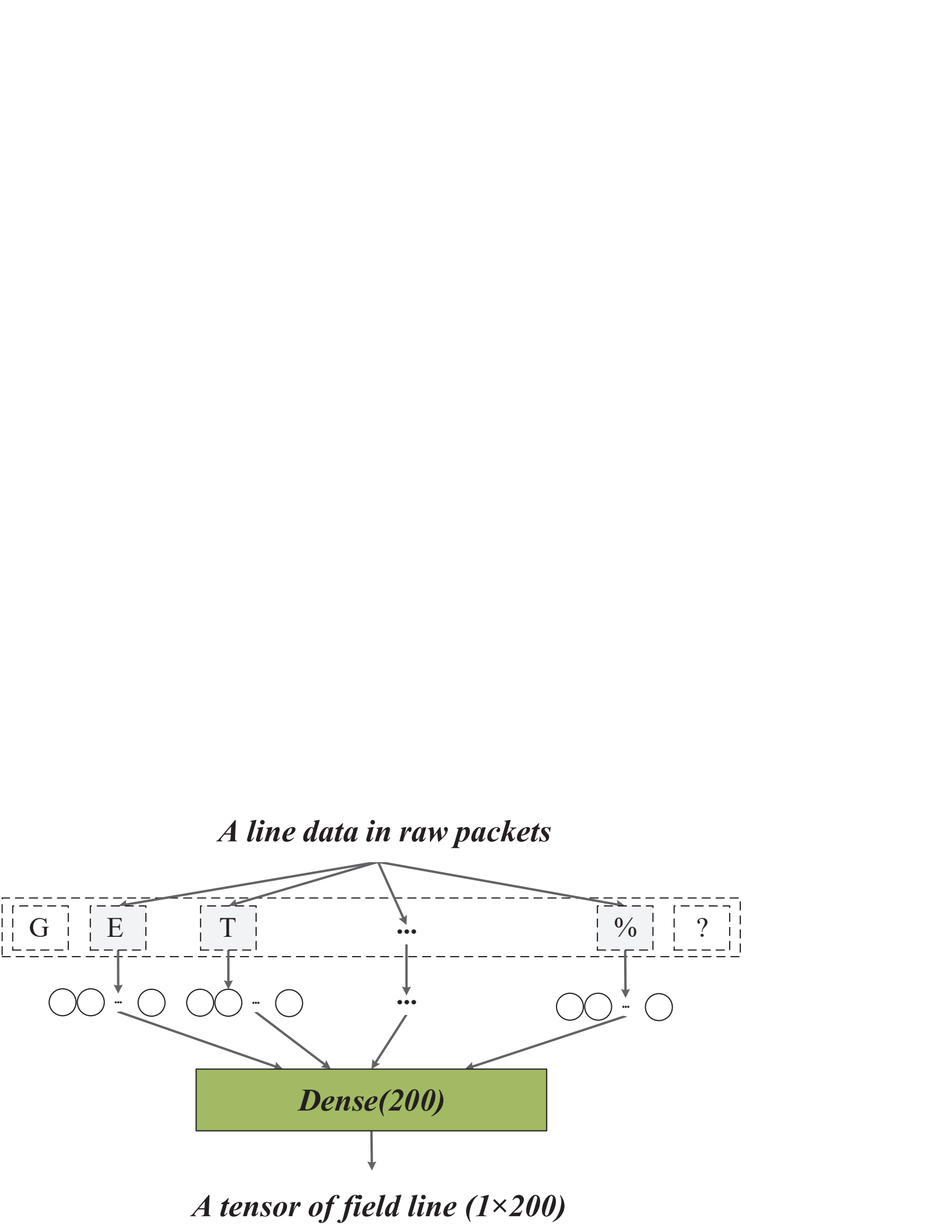
In the future, we will expand the dataset BTHT further by adding more malicious traffic, and then, enhance the generalization of the model through data iteration. In addition, we only perform a simple coarse-grained partitioning of traffic in this paper, but as the complexity of traffic increases, more fine-grained hierarchical partitioning and feature extraction must be performed. This is one of our next research work.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (No.2016YFB0801502), and the National Natural Science Foundation of China (Grant No.U1736218). The corresponding author of this paper is Shuhao Li.

REFERENCES

- [1] CNCERT/CC, *2017 China Internet Cyber Security Report*. People Post Press, 2 2018.
- [2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [3] F. Zhang and D. Wang, "An effective feature selection approach for network intrusion detection," in *2013 IEEE Eighth International Conference on Networking, Architecture and Storage*. IEEE, 2013, pp. 307–311.
- [4] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Computer Communications*, vol. 49, pp. 1–17, 2014.
- [5] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.
- [6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.
- [7] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks," *Expert systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.
- [8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2013.
- [9] O. Can and O. K. Sahingoz, "A survey of intrusion detection systems in wireless sensor networks," in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*. IEEE, 2015, pp. 1–6.
- [10] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, no. 4, pp. suppl27–suppl30, 2002.
- [11] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.
- [12] C.-M. Chen, D.-J. Guan, Y.-Z. Huang, and Y.-H. Ou, "Anomaly network intrusion detection using hidden markov model," *Int. J. Innov. Comput. Inform. Control*, vol. 12, pp. 569–580, 2016.
- [13] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13–21, 2015.
- [14] S. Rosset and A. Inger, "Kdd-cup 99: knowledge discovery in a charitable organization's donor database," *SIGKDD Explorations*, vol. 1, no. 2, pp. 85–90, 2000.
- [15] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.
- [16] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.
- [17] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433, pp. 346–364, 2018.
- [18] A. Gezer, G. Warner, C. Wilson, and P. Shrestha, "A flow-based approach for trickbot banking trojan detection," *Computers & Security*, vol. 84, pp. 179–192, 2019.
- [19] X. Yang, L. Kong, L. Zhi, Y. Chen, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, no. 99, pp. 35 365–35 381, 2018.
- [20] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2016, pp. 581–585.
- [21] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1222–1228.
- [22] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 858–866.
- [23] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *International Conference on Platform Technology and Service*, 2016, pp. 1–5.
- [24] E. Chen and T. Bates, "Rfc 1998: An application of the bgp community attribute in multi-home routing," *August 1996*, 1996.
- [25] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 826–834, 1983.
- [26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [27] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [28] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769*, 2018.



Statistical data at packet level
PL (1 × 100)



A tensor at packet level
pl-vector (1 × 20)

Statistical data at flow level
FL (1 × 170)



A tensor at flow level
fl-vector (1 × 30)

GET /mrow_cntrl/?id&rnd=148015984 HTTP/1.1
User-Agent: KUKU v3.10 exp
Host: www.kukustrustnet7.info
Cache-Control: no-cache

