A Semi-supervised Learning Method for Malware Traffic Classification with Raw Bitmaps

Jingrun Ma[1,2], Xiaolin Xu[3], Tianning Zang[1,2], Xi Wang[1(✉)], Beibei Feng[1,2], and Xiang Li[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{majingrun,zangtianning,wangxi,fengbeibei,lixiang1}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China

Abstract. The rapid growth of malware and its variants has a significant detrimental effect on the security of the Internet infrastructure. In recent years, deep learning-based methods have demonstrated significant success in malware detection. Nonetheless, there are concerns regarding the requirement for substantial labeled data and the feature selection methods used in present approaches. In this paper, we propose a semi-supervised learning-based method for malware traffic classification, which exploits the raw bitmap representation of malware traffic. We employ stacked bi-LSTM to learn the feature representation of malware traffic and adopt semi-supervised learning (SSL) to enhance the model performance by leveraging unlabeled traffic. Pseudo-labeling and consistency regularization are used to produce pseudo-labels, which can compute unsupervised loss. The loss function consists of two terms: a supervised loss applied to labeled data and an unsupervised loss, which are combined together for model training. Experiments indicate that our method is capable of classifying malware traffic with satisfactory accuracy.

Keywords: Malware traffic classification · Semi-Supervised Learning

1 Introduction

With the proliferation of Internet technology, cyber attacks are becoming more frequent and sophisticated. Malware plays a notorious role in cyber attacks. Adversary among cyber attacks often utilizes malware to achieve their ulterior purpose, which results in extensive damage to individual users, corporations, and governments. According to a report of Statista, the worldwide number of malware attacks has reached 5.5 billion in 2022 [2]. It is important for the cyber security expert to accurately identify malware in the network, while it is not a trivial thing. An automatic identification method is able to facilitate the detection of malware, which would minimize the threat of cyber attacks on enterprise networks to some extent.

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 341–356, 2024.
https://doi.org/10.1007/978-3-031-54528-3_19

In order to cope with increasingly complex cyber attacks, there is a variety of works on malware classification based on network traffic analysis using machine learning techniques. They establish machine learning frameworks by leveraging various network traffic features to identify malware on a network. Yan et al. [21] propose a two-tier method that utilizes statistical features of network traffic to detect malware. Furthermore, they enhance it with incremental learning, which can update models without retraining from scratch. Chen et al. [6] propose an imbalanced data gravitation-based classification algorithm to classify mobile malware and build a machine learning based model, which leverages 6 statistical features extracted from network traffic. Gezer et al. [7] propose a machine learning based method to detect TrickBot malware by utilizing 37 statistical features. Wang et al. [18] propose an Android malware detection method that utilizes text semantic features of HTTP traffic.

However, these machine learning-based methods conduct the analysis relying on the statistical features of network traffic, which need expert knowledge to carry out feature engineering. Moreover, some methods need to inspect the payload of packets to extract features, which may bring privacy risks and will be useless when encrypted traffic is introduced. Additionally, these supervised learning based methods often require a large labeled dataset to achieve competitive performance, which needs lots of expert manual labeling work.

In this paper, we address the above problem by developing a semi-supervised learning-based method that exploits raw bit features of network traffic to classify malware. Our method can identify malware efficiently as it relies only on the raw binary data of a few packets without any statistical computation or inspecting payloads. A traffic flow refers to a packet sequence that consists of multiple packets. We extract bitmap representation from each packet. Therefore, we represent a flow as a collection of bitmap representation features, which can characterize each malware from the original bits angle. We utilize bi-LSTM to extract temporal features between packets for learning the timing relationship. Moreover, by leveraging unlabeled data, we use semi-supervised learning (SSL) to improve performance and mitigate the reliance on labeled data to some extent. Pseudo-labeling and consistency regularization are used to produce pseudo-labels. Specifically, the pseudo-label is generated based on an original unlabeled flow. When the model is fed an augmented version of the same flow, the pseudo-label is used as a target to compute cross-entropy loss. We leverage mask technology for augmentation, which can produce disturbed versions of a given flow. Additionally, we retain an pseudo-label only if the model assigns a probability exceeding the threshold.

Contributions of this paper are as follows:

- We conduct an analysis of the bitmap representation of various types of malware traffic, which facilitates the characterization of traffic associated with malware.
- We propose a deep learning-based method that exploits the raw bitmap representation features to classify malware traffic. The method employs semi-supervised learning to derive effective features from unlabeled data, which

can mitigate the reliance on labeled data. And stacked bi-LSTM is incorporated to learn temporal features between packets. Our method characterizes the traffic patterns of malware without inspecting packet payloads, which can avoid privacy leakage risks.
– We conduct adequate experimentation to validate our methodology. According to experiment results, our method achieves an average accuracy of 96.41% and an average F1-score of 96.47% across three distinct datasets.

The rest of this paper is organized as follows: Sect. 2 describes relevant prior work. In Sect. 3, we present the data preprocessing of malware traffic and analysis of bitmap representation features. We introduce the methodology detailedly in Sect. 4. Section 5 presents the experiments and results. Finally, Sect. 6 concludes this paper.

2 Related Work

Semi-supervised Learning. Lee et al. [10] proposed a simple and efficient method of semi-supervised learning by using pseudo-label. They trained the model on labeled and unlabeled data simultaneously. For unlabeled data, they picked up the class which has the maximum predicted probability as pseudo-label. Sajjadi et al. [14] proposed an unsupervised loss function that minimizes the difference of diverse views of the same sample, and these views can be generated by randomized data augmentation, dropout, and random max-pooling. They showed that using the transformation loss function can achieve significant improvements in accuracy. Sohn et al. [16] proposed FixMatch, a simpler semi-supervised learning algorithm that achieves huge improvement across many datasets by using pseudo-labeling and consistency regularization. Aouedi et al. [4] proposed a semi-supervised approach using stacked sparse autoencoder accompanied by denoising and dropout techniques, which can improve the robustness of extracted features and prevent the over-fitting problem during the training process.

Malware Traffic Classification. There are a lot of methods relying on building deep learning models using statistical features of malware traffic. These methods usually leverage supervised learning to characterize traffic patterns for each malware. He et al. [8] proposed an anomaly detection method based on CNN and autoencoder, which can utilize only a few abnormal samples to pre-train the model. Rios et al. [13] used SVM and a broad learning system to detect anomalies and intrusions. Shone et al. [15] presented a nonsymmetric deep autoencoder based method for unsupervised feature learning, which can facilitate improving classification performance. Li et al. [11] proposed a machine learning method for detecting organizations of IoT malware in APT attacks, which can better identify APT activity and protect the security of IoT. Bovenzi et al. [5] provided a fair assessment of three models (Decision Tree, Random Forest, and 1-D CNN) to classify Android malware traffic, and the evaluation indicated that the classical machine learning method is still effective in some scenarios.

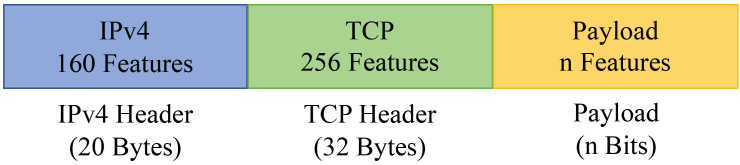


Fig. 1. Bitmap representation of a packet.

3 Data Preprocess and Feature Analysis

In this paper, a bidirectional TCP flow is considered as a sample to be classified. A flow is a collection of multiple associated packets between two computer addresses using a particular protocol on a particular pair of ports [20]. Packets with the same quintuple (*source IP, destination IP, source port, destination port, protocol*) belong to the same flow. The flow is bidirectional, including packets from client to server and server to client.

We use the raw bitmap representation of these flows, which can preserve original features and mitigate reliance on manual feature engineering. Figure 1 illustrates the bitmap representation of a packet. We use 160 bits for the IPv4 header, 256 bits for the TCP header (20 bytes fixed part and 12 bytes options), and n bits for payload respectively. TCP option orderings can have improvement on classification performance, and a binary representation of the TCP options can preserve ordering [9]. We examine three datasets (presented in Sect. 5) and find that the TCP header has 20 bytes in 73.67% packets and 21–32 bytes in 25.63% packets respectively. Overall, 99.3% TCP header of packets is less than or equal to 32 bytes. Therefore, we set 256 bits (32 bytes) for the TCP header. The employment of raw bitmap representation leads to a consistent, pre-normalized representation of each packet.

3.1 Data Preprocess

We extract raw binary data above the Ethernet layer from PCAP files of malware and reconstruct flows by leveraging 5-tuple information. Each flow consists of multiple packets, from which our sequence model will learn representative features.

Moreover, the number of packets in traffic flows is not always the same, which conflicts with the requirement of our classification model which needs a uniform size of input data. Hence, the flows that we reconstructed before need to be processed into the same format. The following unified preprocessing measures including padding and segmentation are applied to make input flows have a uniform size:

- Initially, if the length of the TCP header exceeds 256 bits, we truncate it. Conversely, we pad the TCP header with zeros.
- If the number of bits in a packet payload exceeds a certain threshold, we select the first n bits. Conversely, we pad the feature sequence with zeros.

– If the number of packets in a flow exceeds a certain threshold, we select the first M packets to represent it as a whole. Conversely, we pad the packet sequence with zeros.

A traffic flow is preprocessed according to the above rules, which allows for reducing the amount of data and unifying the data size.

Table 1. The Breakdown of MTA

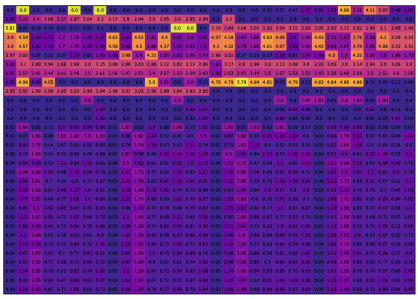
ID	Malware	Flow (#)	ID	Malware	Flow (#)
0	Benign	1,000	17	Goon	458
1	Hancitor	1,000	19	Flashpack	430
2	Qakbot	1,000	18	Styx	430
3	Fiesta	1,000	20	Bazaloader	421
4	Nuclear	1,000	21	Zloader	401
5	Trickbot	1,000	22	Grandsoft	381
6	Angler	1,000	23	Dridex	356
7	Magnitude	1,000	24	Mirrorblast	305
8	Icedid	1,000	25	Gandcrab	273
9	Rig	971	26	Formbook	238
10	Sweet-orange	958	27	Zuponcic	162
11	Ursnif	936	28	Mydoom	117
12	Bumblebee	882	29	Lokibot	113
13	Neutrino	591	30	Blackhole	108
14	Svcready	549	31	Kaixin	105
15	Astaroth	532	32	Sundown	104
16	Emotet	495	33	Infinity	104

3.2 Bitmap Representation of Packet Analysis

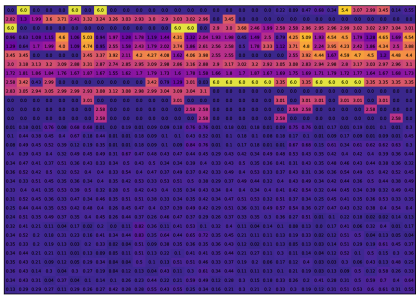
In this paper, we select raw bits sequence to identify the malware traffic correctly, which is unnecessary to compute statistical features of flows. The premise is that bitmap patterns of different malware are distinct, which can be used to characterize the malware [9]. We analyze these bitmaps of different malware on part data from MTA [1] that is also used in the evaluation. The list of malware is shown in Table 1.

We select four types of malware including Sweet-orange, Zloader, Grandsoft, and Zuponcic from Table 1. To conduct an analysis of their bitmap representation, we select M packets of each flow and parse N bits of each packet. By setting $M = 6$, and $N = 1,024$, therefore, a flow can be arranged into a binary matrix of $6 \times 1,024$. Then we reshape it into a matrix of $6 \times 32 \times 32$ to facilitate further

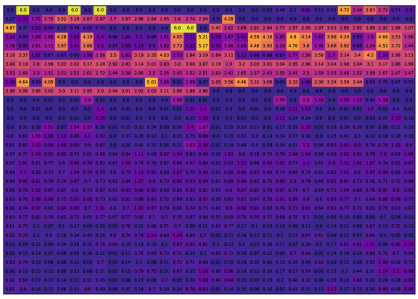
analysis. For the convenience of observation, we sum the corresponding bits of each packet, and a visual heatmap (32×32) of a flow can be obtained. We group flows by malware class and compute the mean heatmap for each malware. After that, each malware has a corresponding heatmap. An advantage of visual representation is that we can efficiently observe the overall situation corresponding to network traffic. Four types of malware heatmap are shown in Fig. 2, and each heatmap exhibits distinct color distribution patterns. It is observed that some malware displays higher heat in specific regions, while others exhibit higher heat in different areas. By conducting a visual comparison, we observe that the four types of malware have obviously different patterns of bitmaps. These disparities serve as a foundation for purposes of classification.



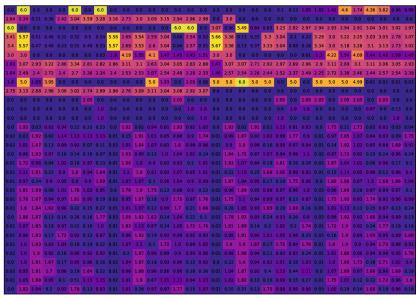
(a) Heatmap of Sweet-orange.



(b) Heatmap of Zloader.



(c) Heatmap of Grandsoft.



(d) Heatmap of Zuponcic.

Fig. 2. Heatmaps of different malware.

4 Methodology

In this section, we build a semi-supervised learning based neural network model. Afterwards, we provide a comprehensive overview of the model's overall structure and detail each individual component.

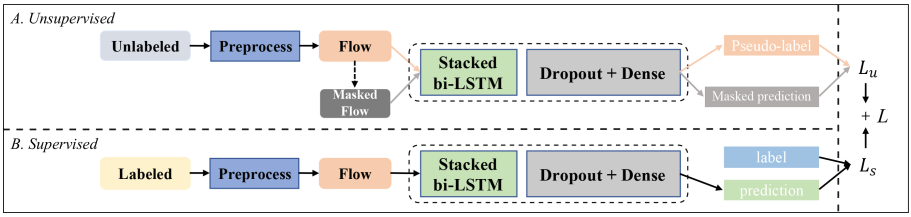


Fig. 3. Workflow of the method.

4.1 Overview

The proposed semi-supervised learning based neural network model consists of an unsupervised module and a supervised module, as shown in Fig. 3. The raw binary data are extracted from unlabeled and labeled flows during data preprocessing for the unsupervised module and supervised module respectively. Then the unsupervised module computes an pseudo-label for each unlabeled flow that is then used in a standard cross-entropy loss. To obtain an pseudo-label, we first compute the model's predicted class distribution of a given unlabeled flow. If the predicted probability exceeds a given threshold, the predicted class is used as a pseudo-label. Subsequently, we conduct data augmentation within the unsupervised module by randomly masking certain fields of raw binary data. And we leverage the pseudo-label and the model output for the augmented flow to compute loss L_u. Then the supervised module computes loss L_s for labeled flows. Finally, the sum of L_u and L_s is used as the overall cross-entropy loss L to train the model.

4.2 Semi-supervised

Semi-supervised methods typically generate an pseudo-label for unlabeled data and train the model with a lot of unlabeled data and a small amount of labeled data.

Unsupervised. We leverage pseudo-labeling and consistency regularization to complete the unsupervised module, like [16].

Pseudo-labeling leverages the idea of using the model itself to obtain pseudo-labels for unlabeled data. Given a batch of K unlabeled flows U, $U = \{u_1, u_2, u_3, ..., u_K\}$, where u_i is an unlabeled flow, the pseudo-label can be generated by:

$$y_u^i = argmax(F(u_i)) \tag{1}$$

where $F(\cdot)$ is the neural network that can extract representation for a given input x. In order to make the model have a better predictive effect, we retain the pseudo-labels only when $max(F(u_i)) \geq \theta$, where θ is a manually set threshold.

Consistency regularization aims to produce consistent predictions for input from different views, which means its prediction should be consistent even if

the input is slightly disturbed. Data augmentation provides different views of a single input. For an unlabeled flow u_i, we apply an augmentation $Aug(\cdot)$ by masking certain fields of u_i. The u_i consists of M packets, and each packet has N bits. We divide these N bits into 24 parts according to the IP and TCP protocol specifications. The details are shown in Table 2. Within this table, Start and End respectively denote the initial and terminal indices of each partition. Furthermore, the interval is inclusively bounded by the left endpoint, while exclusively bounded by the right endpoint $[start, end)$. We randomly select m packets, and for each packet, we randomly select n parts ($n < 24$), and mask them with -1.

Then we compute the cross-entropy loss between pseudo-label y_u^i and the output for an augmented version of u_i as follows:

$$L_u = \frac{1}{K} \sum_{i=0}^{K} \mathbb{1}(max(F(u_i)) \geq \theta) H(y_u^i, F(Aug(u_i))) \tag{2}$$

where $H(\cdot)$ denotes cross-entropy loss function.

Table 2. IP and TCP Specification Partition

Field Name	*start*	*end*	Field Name	*start*	*end*
IP version	0	4	TCP sport	160	176
IP header len	4	8	TCP dport	176	192
IP service	8	16	TCP seq	192	224
IP total len	16	32	TCP ack	224	256
IP identification	32	48	TCP header len	256	260
IP flags	48	51	TCP keep	260	266
IP frag offset	51	64	TCP flags	266	272
IP ttl	64	72	TCP window size	272	288
IP protocol	72	80	TCP header check	288	304
IP hdr check	80	96	TCP urg ptr	304	320
IP src	96	128	TCP options	320	416
IP dst	128	160	TCP payload	416	N

Supervised. For labeled flows, L_s is the standard cross-entropy loss. Given a batch of B labeled flows X, $X = \{x_1, x_2, x_3, ..., x_B\}$, where x_i is labeled flow, the L_s can be computed by:

$$L_s = \frac{1}{B} \sum_{i=0}^{B} H(y^i, F(x_i)) \tag{3}$$

After acquiring the loss of unlabeled flows and labeled flows respectively, the overall loss can be computed as follows:

$$L = L_s + \lambda_u \cdot L_u \tag{4}$$

where λ_u is a hyperparameter denoting the weight of the unlabeled loss.

4.3 Model Architecture

Network traffic is the conversation between the client and the server, which means that the packet sequence is naturally temporal. LSTM is a special kind of Recurrent Neural Network (RNN) that can process a sequence of inputs. Traditional RNNs have the disadvantage of long-term dependency problem which will result in gradient disappearance or explosion. LSTM uses gate structure to remember or forget information to avoid the long-term dependency problem. Therefore, we use stacked bi-LSTM to extract feature representation and followed by a dropout layer and a fully-connected layer to classify malware traffic.

Stacked Bi-LSTM. An LSTM has three gates to control the cell state including forget gate, input gate, and output gate. Before adding new information, The forget gate decides what information to be discarded from the previous sequence. f_t outputs a value between 0 and 1 according to the hidden information h_{t-1} and the input x_t. The current cell keeps more information if f_t is close to 1 otherwise less.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

After discarding the information, the input gate decides what new information to be stored in the current cell. A *tanh* layer creates new information according to the hidden information h_{t-1} and the input x_t.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6}$$

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

Based on f_t, i_t and $\tilde{C_t}$, cell updates C_{t-1} into the new state C_t.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C_t} \tag{8}$$

After the cell is updated, we need to determine the output h_t of the cell, which is controlled by the output gate o_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t \times tanh(C_t) \tag{10}$$

To learn sequential features from both forward and backward directions, we use bidirectional LSTM (bi-LSTM) to incorporate the contextual features of the packet feature sequence. To improve the representation of the model, we adopt multi-layer bi-LSTM to learn the low-level and high-level features at the same time.

Dense Layer. The last part of the model includes a dropout layer and a fully-connected layer followed by a softmax function to obtain a prediction vector.

5 Evaluation

In prior sections, we analyze the traffic of malware and preliminarily introduce the architecture of the model. This section sequentially introduces the experiment-used datasets, the evaluation metrics, and the performance comparison between the prior work and ours.

5.1 Datasets

We conduct experiments on three public datasets to verify the effectiveness of the method. Next, we give a brief description of these datasets. Table 3 shows the breakdown of these datasets.

Table 3. The Breakdown of Datasets

Dataset	Malware (#)	Flows (#)	Year
MTA [1]	34	19,420	2013-2023
Stratosphere [17]	68	13,065	2013-2021
USTC-TFC2016 [19]	9	4,236	2016
Unlabeled	-	108,752	-

MTA. Malware-traffic-analysis.net [1] is a website that provides sources for packet capture (PCAP) files and malware samples. We discard both the PCAPs with multi-class flows and the malware that owns only a few flows here. We also set max flows per malware. Finally, 34 types of malware are left to use in this experiment.

Stratosphere. Stratosphere ISP [17] provides a Malware Capture Facility Project that is responsible for making long-term malware captures and continually obtaining malware and normal data. It includes over 400GB PCAPs, and to obtain a balanced dataset, we use a part of them to evaluate our method. Specifically, after discarding extremely large PCAP files, 68 types of malware are used in our evaluation.

USTC-TFC2016. Wang et al. [19] publish a dataset consisting of ten types of malware traffic and ten types of normal traffic respectively. After excluding a malware that solely contained UDP flows, we use 9 types of malware traffic in the evaluation.

Unlabeled. In order to conduct semi-supervised learning, we selectively utilize a portion of the traffic present in the first three datasets, along with our own laboratory-derived traffic, as the unlabeled dataset. Our own laboratory-derived traffic is incorporated as benign traffic to enhance the diversity. It is pertinent to note that only a designated fraction of the unlabeled traffic was incorporated into the training process, namely, in a 5:1 ratio with labeled traffic.

5.2 Metrics

Here, we use precision (P), recall (R), F1-score (F1), and accuracy (ACC) to evaluate the effectiveness of our approach. In our setting, for each malware, a correctly identified flow is treated as a true positive (TP); a flow identified as belonging to this malware but actually not is treated as a false positive (FP); a flow of this malware identified to others is treated as a false negative (FN). Based on these three definitions, above metrics are defined for each device as follows.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1\text{–}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{13}$$

We use macro averages of these three metrics and accuracy in evaluation. Accuracy is defined as follows.

$$Accuracy = \frac{\#\ of\ correct\ identification}{\#\ of\ total\ identification} \tag{14}$$

5.3 Experimental Setting

We take the raw bits as the input of the model. The packet number is set to 6 and bits number is set to 2,048 (256 bytes). The λ_u is set to 1 and θ is set to 0.95. Besides, we set the dimension of hidden states of each bi-LSTM to 256 and take the 2-layer bi-LSTM in each sequence. The dense layer size is also set to 256. Moreover, we take dropout with a 0.3 ratio to avoid over-fitting, and use 0.001 as the learning rate of the *AdamW* optimizer. We implement our approach with Pytorch 1.13 and deploy it on a server with 20-cores CPU and 64GB memory. The server uses an NVIDIA 2080 for accelerating computing.

5.4 Experimental Results

We use three well-known malware traffic datasets to evaluate the performance of our method in terms of precision, recall, F1-score, and accuracy. After extracting features from malware traffic to generate training and testing samples, we have

randomly split these instances into three groups, one containing 80% of the instances for training, one containing 10% of the instances for validating, and the other containing 10% of the instances for testing.

Table 4. Result on Three Datasets

	MTA [1]				Stratosphere [17]				USTC-TFC2016 [19]			
Method	ACC (%)	P (%)	R (%)	F1 (%)	ACC (%)	P (%)	R (%)	F1 (%)	ACC (%)	P (%)	R (%)	F1 (%)
DT	91.92	90.36	91.15	90.64	92.29	92.03	92.12	91.92	92.33	92.92	92.73	92.8
RF	94.44	95.65	93.38	94.23	95.96	96.01	95.7	95.74	92.1	92.37	92.59	92.36
CNN [19]	76.93	80.48	76.61	77.49	81.76	82.84	81.59	81.47	90.33	91.0	91.38	91.14
CNN+LSTM [12]	81.31	79.46	79.7	79.31	84.41	84.21	84.37	84.06	86.67	88.2	87.95	88.0
Ours (bytes)	90.58	91.08	89.8	90.29	88.32	88.5	88.03	87.93	87.5	88.28	89.0	88.47
Ours (bits)	**97.48**	**97.91**	**97.27**	**97.51**	**97.65**	**97.51**	**97.48**	**97.46**	**94.1**	**94.49**	**94.45**	**94.45**

Table 4 presents the experiment results on three datasets. Our method reaches the accuracy of 97.48%, 97.65%, and 94.1% on MTA, Stratosphere, and USTC-TFC2016 datasets, respectively. Precision, recall, and F1-score all exceed 94.45%. In order to better analyze the classification result of each type of malware traffic, we generate a confusion matrix for the result on MTA dataset. Figure 4 shows the confusion matrix on MTA dataset, in which rows represent actual labels and columns represent predictions. The diagonal values show the correct classifications. Figure 4 illustrates that our method performs well on most malware and only a few types are not correctly identified. Our method has reached an identifying recall of 100% on 21 types of malware. The Blackhole malware has the lowest recall rate, with only 81.8% of instances, and 18.2% of the Blackhole samples are found to be misidentified as Styx. Figure 5 shows the heatmap of Blackhole and Styx. It is obvious that there is a high level of similarity between their heatmaps, extending from the protocol header to the payload. BlackHole and Styx are two exploit kits that work in the same way: each includes a variety of exploits plus an administrator panel. We speculate that they are based on what is essentially the same implementation, which results in a challenge to differentiate between them. Overall, the experimental results clearly show that our approach has the ability to conduct accurate malware traffic classification.

Comparison with Other Methods. To demonstrate the effectiveness of our method, we conduct a comparative analysis with four other methods:

- Decision Tree (DT) and Random Forest (RF), are traditional machine learning methods. We use scikit-learn to perform Decision Tree (DT) and Random Forest (RF) [3]. In order to prove the effectiveness of our model, they also take bitmap representation as input.
- CNN [19], is a deep learning method. Wang et al. [19] propose a malware traffic classification method using a convolutional neural network by taking traffic data as images. The method uses the first 784 bytes of each bidirectional flow, and the bytes can be transformed into images with shape 28×28.

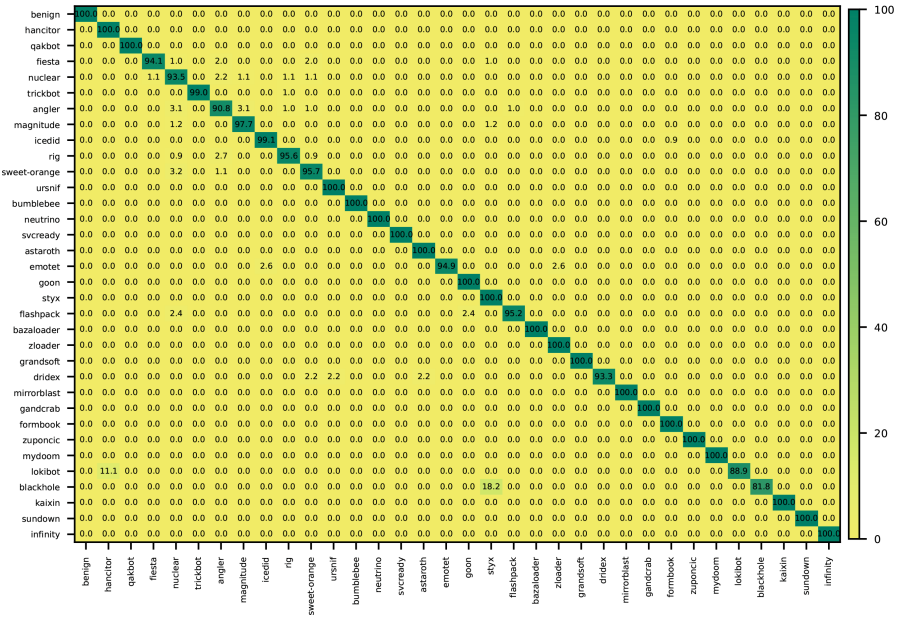
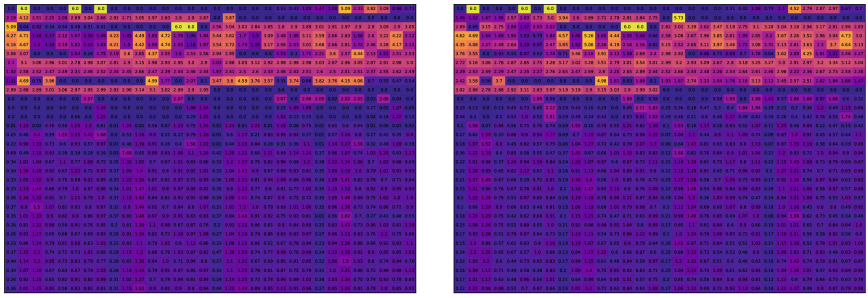


Fig. 4. Confusion matrix on MTA dataset.



(a) Heatmap of Styx. (b) Heatmap of Blackhole.

Fig. 5. Heatmaps of Styx and Blackhole.

– CNN+LSTM [12], is a deep learning method. Marín et al. [12] propose a deep learning based method that also uses raw bytes as the input. The method uses CNN to learn spatial features from raw bytes data and uses LSTM to learn temporal features from packet sequences.

Our method achieves a notable margin of improvement compared with other methods on MTA and Stratosphere. On USTC-TFC2016, our method exhibits a less substantial but still noteworthy edge in performance.

For DT and RF methods, RF outperforms the DT on MTA and Stratosphere, while slightly lags behind the DT method on USTC-TFC2016. Our method

demonstrates superior performance compared to RF in all metrics for MTA and Stratosphere datasets. Specifically, we achieve a 2.37% higher average accuracy and a 2.5% higher average F1-score than RF on these datasets. Moreover, it exhibits better performance than DT in all metrics on USTC-TFC2016. We obtain a 1.77% higher accuracy and a 1.65% higher F1-score than DT on this dataset. The results compared with DT and RF demonstrate the effectiveness of our semi-supervised neural network model.

For CNN [19] and CNN+LSTM [12] methods, our method outperforms theirs on all three datasets. We reach at least 16.17%, 13.24%, and 3.77% more than theirs on accuracy across three datasets. Additionally, we achieve a superior F1-score compared to theirs, with an increase of 18.2%, 13.4%, and 3.31%, respectively. Our semi-supervised model, along with bitmap representation, has been proven effective when compared to their results.

Bits vs Bytes. Several studies utilize raw bytes sequence as input for training neural networks, including [12,19]. Therefore, we also take raw bytes sequence as the input of our method and conduct experiments on three datasets. Table 4 demonstrates that our method utilizing bits as input yields superior results to our method utilizing bytes, exhibiting an improvement in F1-score of 7.22%, 9.53%, and 5.98% respectively across three datasets. Notably, our method utilizing bytes also achieves an improvement in F1-score of no less than 10.98% and 3.87% compared to CNN [19] and CNN+LSTM [12] on MTA and Stratosphere. Although it slightly lags behind CNN [19] on USTC-TFC2016, it is worth noting that the overall superiority of our semi-supervised neural network model is evident.

Table 5. Ablation Study of SSL on Three Datasets

Dataset	Metrics	Without-SSL	With-SSL	Gap
MTA [1]	ACC (%)	95.78	97.48	1.7 ↑
	P (%)	95.68	97.91	2.23 ↑
	R (%)	94.14	97.27	3.13 ↑
	F1 (%)	94.58	97.51	2.93 ↑
Stratosphere [17]	ACC (%)	96.89	97.65	0.76 ↑
	P (%)	96.85	97.51	0.66 ↑
	R (%)	96.68	97.48	0.8 ↑
	F1 (%)	96.7	97.46	0.76 ↑
USTC-TFC2016 [19]	ACC (%)	92.33	94.1	1.77 ↑
	P (%)	93.32	94.49	1.17 ↑
	R (%)	92.91	94.45	1.54 ↑
	F1 (%)	92.82	94.45	1.63 ↑

Ablation Study. We present ablation results in Table 5. Without-SSL denotes training without the unsupervised module, while with-SSL refers to training with all modules. Table 5 indicates that incorporating the unsupervised module leads to an improvement in accuracy, precision, recall, and F1-score on the MTA dataset by 1.7%, 2.23%, 3.13%, and 2.93%, respectively, pointing towards superior results. Similarly, on the USTC-TFC2016 dataset, our proposed method achieves an increase in accuracy, precision, recall, and F1-score by 1.77%, 1.17%, 1.54%, and 1.63%, respectively, validating the effectiveness of our approach. For the Stratosphere dataset, although our method shows only a slight improvement, it still improves on all metrics.

Discussion. Our model demonstrates effectiveness in the raw bitmap patterns of malware traffic. Furthermore, semi-supervised learning can effectively improve experimental outcomes by extracting useful features from unlabeled data. We evaluate our model on various datasets, and its performance demonstrates consistent accuracy. However, we have identified a few limitations in our model. For instance, it marginally enhances the Stratosphere dataset when compared to exclusively utilizing supervised learning. Therefore, we intend to enhance this aspect in future research. Moreover, the RF algorithm demonstrates favorable outcomes across various datasets. Notwithstanding the prevalence of deep learning, traditional machine learning approaches, such as random forest, continue to exhibit some competitiveness.

6 Conclusion

In this paper, we propose a semi-supervised learning-based method for the purpose of malware traffic classification, which takes raw bit sequences of flow as input and automatically infers the malware class. Our method solely relies on network packet traces and utilizes stacked bi-LSTM to learn representative features of typical malware classes from the network traffic. Additionally, we employ semi-supervised learning to effectively extract features from unlabeled traffic, thereby enhancing its classification capability. Experimental results demonstrate that our proposed method achieves high accuracy in identifying malware traffic.

Acknowledgements. We thank the anonymous reviewers for their insightful comments. The corresponding author is Xi Wang.

References

1. malware-traffic-analysis.net, https://www.malware-traffic-analysis.net/
2. Number of malware attacks per year 2022 — statista. https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/
3. scikit-learn: machine learning in python. https://scikit-learn.org/stable/
4. Aouedi, O., Piamrat, K., Bagadthey, D.: A semi-supervised stacked autoencoder approach for network traffic classification. In: 2020 IEEE 28th International Conference on Network Protocols (ICNP), pp. 1–6. IEEE (2020)

5. Bovenzi, G., Cerasuolo, F., Montieri, A., Nascita, A., Persico, V., Pescapé, A.: A comparison of machine and deep learning models for detection and classification of android malware traffic. In: 2022 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6. IEEE (2022)
6. Chen, Z., et al.: Machine learning based mobile malware detection using highly imbalanced network traffic. Inf. Sci. **433**, 346–364 (2018)
7. Gezer, A., Warner, G., Wilson, C., Shrestha, P.: A flow-based approach for trickbot banking trojan detection. Comput. Secur. **84**, 179–192 (2019)
8. He, M., Wang, X., Zhou, J., Xi, Y., Jin, L., Wang, X.: Deep-feature-based autoencoder network for few-shot malicious traffic detection. Secur. Commun. Netw. **2021**, 1–13 (2021)
9. Holland, J., Schmitt, P., Feamster, N., Mittal, P.: New directions in automated traffic analysis. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 3366–3383 (2021)
10. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML. vol. 3, p. 896 (2013)
11. Li, S., Zhang, Q., Wu, X., Han, W., Tian, Z.: Attribution classification method of apt malware in Iot using machine learning techniques. Secur. Commun. Netw. **2021**, 1–12 (2021)
12. Marín, G., Casas, P., Capdehourat, G.: Deep in the dark-deep learning-based malware traffic detection without expert knowledge. In: 2019 IEEE Security and Privacy Workshops (SPW), pp. 36–42. IEEE (2019)
13. Rios, A.L.G., Li, Z., Xu, G., Alonso, A.D., Trajković, L.: Detecting network anomalies and intrusions in communication networks. In: 2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES), pp. 000029–000034. IEEE (2019)
14. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: Advances in Neural Information Processing Systems 29 (2016)
15. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Topics Comput. Intell. **2**(1), 41–50 (2018)
16. Sohn, K.: Fixmatch: simplifying semi-supervised learning with consistency and confidence. Adv. Neural. Inf. Process. Syst. **33**, 596–608 (2020)
17. Stratosphere: Stratosphere laboratory datasets (2015), retrieved March 13, 2020. https://www.stratosphereips.org/datasets-overview
18. Wang, S., Yan, Q., Chen, Z., Yang, B., Zhao, C., Conti, M.: Detecting android malware leveraging text semantics of network flows. IEEE Trans. Inf. Forensics Secur. **13**(5), 1096–1109 (2017)
19. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference On Information Networking (ICOIN), pp. 712–717. IEEE (2017)
20. Xu, C., Shen, J., Du, X.: A method of few-shot network intrusion detection based on meta-learning framework. IEEE Trans. Inf. Forensics Secur. **15**, 3540–3552 (2020)
21. Yan, A., et al.: Network-based malware detection with a two-tier architecture for online incremental update. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2020)