applied sciences

MDPI

Article

An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level

Ren-Hung Hwang *, Min-Chun Peng, Van-Linh Nguyen and Yu-Lun Chang

Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 62102, Taiwan

* Correspondence: rhhwang@cs.ccu.edu.tw

Received: 12 July 2019; Accepted: 14 August 2019; Published: 19 August 2019

Abstract: Recently, deep learning has been successfully applied to network security assessments and intrusion detection systems (IDSs) with various breakthroughs such as using Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) to classify malicious traffic. However, these state-of-the-art systems also face tremendous challenges to satisfy real-time analysis requirements due to the major delay of the flow-based data preprocessing, i.e., requiring time for accumulating the packets into particular flows and then extracting features. If detecting malicious traffic can be done at the packet level, detecting time will be significantly reduced, which makes the online real-time malicious traffic detection based on deep learning technologies become very promising. With the goal of accelerating the whole detection process by considering a packet level classification, which has not been studied in the literature, in this research, we propose a novel approach in building the malicious classification system with the primary support of word embedding and the LSTM model. Specifically, we propose a novel word embedding mechanism to extract packet semantic meanings and adopt LSTM to learn the temporal relation among fields in the packet header and for further classifying whether an incoming packet is normal or a part of malicious traffic. The evaluation results on ISCX2012, USTC-TFC2016, IoT dataset from Robert Gordon University and IoT dataset collected on our Mirai Botnet show that our approach is competitive to the prior literature which detects malicious traffic at the flow level. While the network traffic is booming year by year, our first attempt can inspire the research community to exploit the advantages of deep learning to build effective IDSs without suffering significant detection delay.

Keywords: deep learning for network security; long short-term memory; malicious traffic classification

1. Introduction

In a gigantic connected world like the Internet-of-Things (IoT), protecting the network from network attacks may require a comprehensive approach to defeat while under the limitation of the existing resources, e.g., the capacity of the gateway or edge server. The task is getting much harder with the rapid increase of the volume of attacks such as the distributed denial of service (DDoS) from IoT devices. For example, a Mirai-based botnet launches one of the biggest DDoS attacks in history directed at Dyn in October of 2016. The attack surpassed 1 Tbps at peak and created disruption for many major sites, including AirBnB, Netflix, PayPal, Visa, Amazon, the New York Times and so on. Currently, the variants of Mirai malware such as Satori and Miori are still storming to the network with the new records of traffic volume towards the victim, including the systems of the enterprise companies. In practice, the attacker often handles a large number of the botnets of injected IoT devices to launch such an attack. As a result, detecting the malware traffic in the early period of distributing the malicious code can significantly help to prevent the malware from becoming widespread, and mitigate the attack magnitude.

Over the decades, the malicious traffic detection system is the top topic and considered the most in the intrusion detection system (IDS) field. Thus far, the most popular approach of such IDSs is to build a classification mechanism to distinguish the incoming traffic into benign and malicious classes. An IDS can use the signature-based or rule-based approach to identify abnormal traffic. Apparently, the major drawbacks of these traditional IDSs include: (1) requiring many efforts, e.g., labor, to maintain the signature database and updates; (2) becoming ineffective for maintaining such database if the new traffic types or attack variances increasingly spike in a short time. In addition, many adversaries still have certain advantages because exploiting vulnerabilities in the systems of millions of codes, including the zero-day ones available on the darknet, may not require much effort. When the ecosystem of Internet-connected systems expands and the number of IoT devices inside increased rapidly, the attack surface also increases. Consequently, that increase can cause a greater risk of attack and easily overwhelm the update requests. Note that the traffic promises to explode quickly when massive quantities of IoT applications are predicted to flood the market in the next few years, particularly with the readiness of 5G networks. In this case, deep learning promises to be a game changer to ignite new detection approaches. The most benefits of the deep learning approaches are to build a thorough pattern that can highly represent the characteristics of a specific object through auto-learning a large volume of data and species.

Although applying deep learning (DL) in network security has just emerged recently, the topic has received a lot of attention from the research community [1–7] due to the robust auto-learning ability of DL. In addition, the evolution and the increased availability of GPU-processors significantly help to accelerate matrix computations and massive mathematical calculations, thus directly supporting the feasibility of the DL-based approaches. Deep learning for malicious traffic detection is generally categorized into many classes that are primarily based on the built-in network model, e.g., Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) or unsupervised learning such as an auto-encoder. However, existing literature still suffers several critical drawbacks. First, such systems are built on the flow-based approach (i.e., collecting packets of the same flow for a certain period of time and then classifying packets into flows) and evaluated offline. Since the system needs to allocate significant time for accumulating the traffic flow, it is not applicable to an online real-time system. This packet collection time can be extremely high if the flow is likely the part of a long session data. Furthermore, extracting the features for the flow-based detection system also requires a certain period of time. Apparently, this whole traffic profiling process naturally increases the overall time of the detection. Secondly, the flow-based classification can require a lot of resources, e.g., memory and storage, to store and process the accumulated traffic, including the deep checking on the assembled data. Some recent works, as well as our previous work [8], have shown that it is not necessary to examine all packets in a flow to determine whether it is a malicious flow; instead, a few first packets of the flow are enough for the detection purposes.

Based the above observation, in this research, we propose a novel approach to build a packet-based malicious traffic classification framework. Instead of wasting time on waiting for a complete flow of packets and then launching the classification, we use word embedding to learn the similarity in semantics and syntax of header fields a packet and leverage LSTM to learn temporal features of the fields. The goal of the method is to identify the attack only by directly monitoring a packet.

Due to considering the traffic at the packet level, our system has significant advantages to speed up the detection process, e.g., ignore checking a large number of packets in a session. The experimental results show that our method is competitive and prominent better in the term of accuracy, precision, recall rate, and F1-measure as compared to the prior work.

In summary, the main contributions of this study are as follows:

(1) We propose an LSTM-based deep learning approach for the packet-level classification in IDSs. Specifically, our approach can distinguish semantic meanings of malicious traffic by packet, and are thus highly competitive with the flow-based DL approaches while reducing significant time on flow processing.

(2) We propose a novel word-embedding and LSTM model to classify the incoming packet into the normal and abnormal state. The evaluation results of the proposed model on ISCX2012, USTC-TFC2016, IoT data set from Robert Gordon University and IoT data set collected on our Mirai bonet show that the proposed framework can achieve near 100% accuracy and precision in detecting malicious packets.

(3) Our discussion on packet-based deep learning classification and detection promises to provide valuable information and inspire the research community to overcome the remaining challenges, particularly for the target of speeding up the detection process in DL-based IDSs.

The remainder of this paper is organized as follows. Section 2 lists several relevant fundamental concepts and state-of-the-art work of traffic classification and deep learning detection approaches. Section 3 presents our methodology and the detail architecture of the packet classification module in our classification system, along with the data set for evaluation. We show the experimental design and results in Section 4. The conclusions and future work of this paper are summarized in Section 5.

2. Related Work

Deep learning for malicious traffic detection has gained several notable achievements with various network models. For example, the authors in [2] proposed a novel network-based anomaly detection method which extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic emanating from compromised IoT devices. The method is evaluated on commercial IoT devices infected by authentic botnets such as Mirai and BASHLITE. However, the performance of the work primarily relies on several self-generated synthetic data sets, which may lack the diversity of data exchange. In another research work [3], the authors proposed a malware traffic classification method using CNN by considering traffic data as images. The work is one of the first attempts to apply a representation learning approach for malware traffic classification from raw traffic. Unfortunately, the lack of detecting the unknown attacks and high detection time limits its prospect to deploy in practice. In addition, Li et al. [7] and Yin et al. [6] proposed using an RNN model or a combination with Restricted Boltzmann Machines (RBM) for extracting micro-flow features based on a small set of packets and then directly training with the raw packets. However, they mostly target the detection accuracy and ignore the evaluation of the detection time. Thus far, there are several other state-of-the-art relevant studies such as [9,10]. Generally, the key points of such work are that they all rely on the flow or session-based approach, i.e., indexing the traffic by the flow. Thus far, we have found no attempt to detect the attack traffic at the packet level. Our statistics on the relevant research and our research position are summarized in Table 1.

Table 1. Summary of several state-of-the-art relevant research works and our research location. (Notations: GRU: Gated Recurrent Unit; TSDNN: Shaped Deep Neural Network; QDBP: Quantity Dependent Backpropagation; WEDL: Word Embedding-based Deep Learning; NIDS: Network Intrusion Detection System).

Research	Year	Method	Feature	Shortcomings
M. S. Kim [11]	2004	Formulization	flow-based	For offline detection
R. Fu [12]	2016	LSTM, GRU	flow-based	For offline detection
B J. Radford [9]	2018	LSTM	flow-based	For offline detection
C. Li [7]	2018	RNN, RBM	process packet, flow-based	For offline detection
Y. Chen [4]	2018	TSDNN, QDBP	flow-based, N class	For offline classification
X. Yuan [5]	2017	LSTM, GRU	process packet, flow-based	For offline detection
J. Cui [10]	2018	WEDL-NIDS	process packet, flow-based	For offline detection
Ours	2019	LSTM, Word-embedding	raw data, packet-based	Target for online detection

Recently, word2vector [13,14] and LSTM-based learning models [10] provide a very powerful tool to give packet semantic meanings. This means we now can find a way to provide a thorough pattern for the malicious traffic by deep learning of a large number of raw packets. This particularly

helps to reinforce our approach to consider the malicious traffic classification at the packet level. Compared to the prior studies, there are two major differences in our work: (1) the system can identify the malicious traffic by classifying the individual packet, instead of checking the whole packets of a traffic flow; (2) our system can work directly with raw packets, i.e., reading and making detection decisions. To the best of our knowledge, our study is the first work on packet-based malicious traffic detection. In addition, despite the high detection accuracy achieved by prior work for the same attack types, a novel research direction on packet-based classification (i.e., ours) sheds a light toward research on online real-time deep learning based IDSs.

3. Methodology

3.1. Dataset

In deep learning, a lack of a variety of shareable traces dataset can be seen as one of the most obvious obstacles to obtain realistic progress on traffic classification. Many researchers may like creating synthetic datasets through their testbed. The shortcomings of this approach are that their generated traffic may not represent well the real traffic on the Internet, thus the performance evaluation results of the proposed solutions can face the problems of the credibility. Over the decades, the researchers have been recommended to use public famous traffic datasets such as KDD CUP1999 and NSL-KDD to test. Each dataset explicitly provides useful statistics on labeled features and the number of benign and malicious flows as well. However, these datasets do not provide information at the raw traffic level, which is required in our approach. In addition, while the credible dataset from Microsoft Malware Classification Challenge [15] can provide a metadata manifest and hexadecimal representation of the malware's binary content, the missing of packet information in the data, unfortunately, makes it hard to be used in this research.

For the ones matching the requirements, USTC-TFC2016 [3] is one of the most prominent datasets. Table 2 summarizes the statistics of the benign and malware traffic in the dataset. As their statement, there are a total of ten types of malware traffic from public websites which were collected from a real network environment from 2011 to 2015. Along with such malicious traffic, the benign part contains ten types of normal traffic which were collected using IXIA BPS, a professional network traffic simulation equipment. The size of USTC-TFC2016 dataset is 3.71 GB, and the format is pcap.

Table 2. Summary of benign and malware traffic in USTC-TFC2016 dataset. (Notations: SMB: Server Message Block; IM: Instant Message; P2P: Peer-to-Peer).

Benign			Malware	
App Type	**Size (MB)**	**Class**	**Malware Type**	**Size (MB)**
Facetime	2.4	Voice/Video	Tinba	2.55
Skype	4.22	Chat/IM	Zeus	13.4
Bittorent	7.33	P2P	Shifu	57.9
Gmail	9.05	Email/Webmail	Neris	90.1
Outlook	11.1	Email/Webmail	Cridex	94.7
WorldOfWarcraft	14.9	Game	Nsisay	281
MySQL	22.3	Database	Geodo	28.8
FTP	60.2	Data transfer	Miuref	16.3
SMB	1206	Data transfer	Virut	109
Weibo	1618	Social Network	Htbot	83.6

For the Mirai-based DDoS traffic, we use the dataset from Robert Gordon University [16], denoted by Mirai-RGU. This data set contains Mirai botnet traffic such as Scan, Infect, Control, Attack traffic and normal IoT IP Camera traffic. It contains ten types of malicious traffic, include HTTP flood, UDP flood, DNS flood, Mirai traffic, VSE flood, GREIP flood, GREETH flood, TCP ACK flood, TCP SYN flood, and UDPPLAIN flood. The dataset includes features such as Time, Source, Destination, Protocol,

Length, and overall payload information. In addition, we also collected a dataset from the Mirai botnet we have built in the campus of National Chung Cheng University(CCU) (denoted by Mirai-CCU), as shown in Figure 1, to generate four types of attack traffic with much bigger attack magnitude: TCP SYN (41 GB), TCP ACK (2.4 GB), HTTP POST (103 GB), UDP (127.06 GB) in the total of 667 GB attack data.

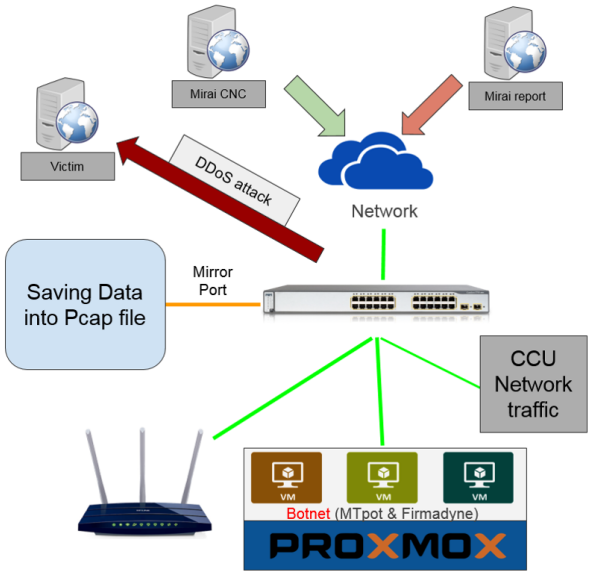

Figure 1. The testbed of our Mirai-based DDoS dataset in the campus of National Chung Cheng University (CCU).

To enrich the dataset for our experiments, we select ISCX2012 [17], which also contains both malicious and benign traffic. ISCX2012 consists of packets collected for seven days. Packets collected in the first and sixth day are normal traffic. In the second and third day, both normal packets and attack packets are collected. In the fourth, fifth, and seventh days, besides the normal traffic, HTTP DoS, DDoS and IRC Botnet, and Brute Force SSH packets are collected, respectively.

3.2. Word Embedding and Data Preprocessing

Our goal is to classify incoming packets into benign or malicious classes without pre-processing the packets into a specific flow. To achieve this target, instead of considering the whole flow (e.g., as a document), we consider each packet (e.g., as a paragraph) and construct the key sentence from each packet, in which each word is a field in the packet header. After that, we apply word embedding [18] to extract semantics and syntax features from this sentence. We choose to consider meanings on the sentence rather than the whole paragraph since the meaning of a paragraph usually can be captured by the key sentence. Here, the order of the fields in each packet (fixed for each packet type) plays the role of resembling some grammar rules which are decisive in building sentence patterns for malicious traffic (signature-based detection) or benign traffic (anomaly detection). Notably, this packet-to-sentence-based model can significantly accelerate the traffic classification, since the behavior and characteristics of one or several first packets can entirely reveal whether their flow is a malicious one.

In general, a field in each packet could be *a byte of the packet header*, *a field of the packet header*, or *a block of the packet payload*. As the initial trial, we consider a field in the packet header as a word and trim a packet to a fixed length of $n = 54$ bytes. Depending on the field length in the packet, the word size can vary. The strict order of the fields in the packet structure constructs a potential grammar rule for the built sentence. Note that the extracting field stage can be done along with the packet reading/decoding (i.e., data pre-processing), thus the resource consumption is quite efficient. In addition, if the length of the packet is less than n bytes, it will be padded with zeros (as shown in

Figure 2). The rationale for examining 54 bytes is that most TCP packets have a 14-byte MAC header, a 20-byte IP header, and a 20-byte TCP header. Table 3 lists the header fields and their length of TCP and UDP packets.

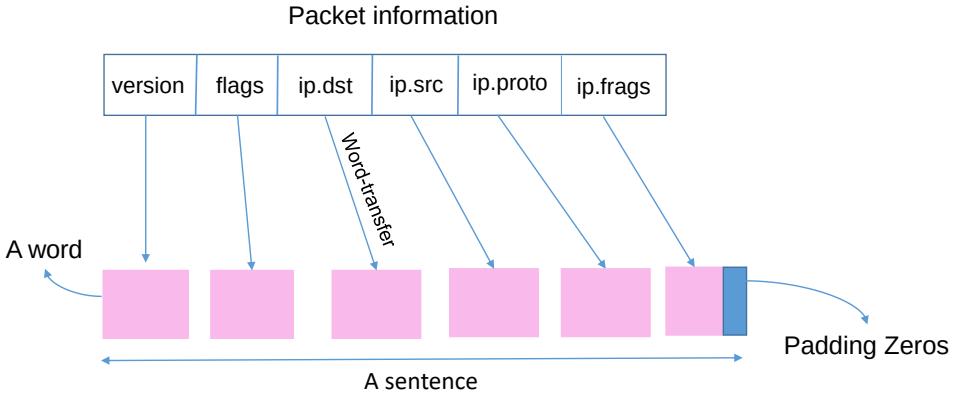

Figure 2. The illustration of the packet-word-transfer mechanism.

Table 3. Packet header(tcp:25, udp: 19) fields, extended fields (tcp: 33, udp: 33) and the length of the fields.

Header	Extended Fields	Details
Ether header (3)	extended to 7 fields	ether.dst(2 × 3) , ether.src(2 × 3), ether.type
IP header (12)	extended to 14 fields	ip.version, ip.ihl, ip.tos, ip.len, ip.id, ip.flags, ip.frag, ip.ttl, ip.proto, ip.chksum, ip.src(2 × 2), ip.dst(2 × 2)
TCP header (10)	extended to 12 fields	tcp.sport, tcp.dport, tcp.seq(2 × 2), tcp.ack(2 × 2), tcp.dataofs, tcp.reserved, tcp.flags, tcp.window, tcp.chksum, tcp.ugptr
UDP header (4)	extended to 12 fields	udp.sport, udp.dpot, udp.len, udp.chksum, 0, 0, 0, 0, 0, 0, 0, 0

After applying the word-embedding technique to fields in the packet header, each header field is embedded, (i.e., to integer number format based on their index in a dictionary of all words) and reshaped, (i.e., dimension) and put to the LSTM-based training model for performing the classification task. In order to understand each attack type and remain consistent with the order of the fields in the packet header, it is important to maintain the sequence order of fields and a consistent word dictionary. Finally, the word size (hyperparameter) selection and further the word-embedding strategy are adjustable and can be specified by the deployment environment and IoT applications under the system's protection umbrella.

3.3. Classification

The classification mechanism is the next important part of the system. As illustrated in Figures 3–5, our packet-based traffic classification framework consists of three modules: packet pre-processing module, word embedding module, and LSTM module. However, to perform training in the LSTM module, we first need to label the benign/malicious mark for the packets in the dataset, e.g., ISCX2012 [17], USTC-TFC2016 [3] and Mirai-RGU [16]. Based on the label of the flows (well-labeled in ISCX2012 and USTC-TFC2016 dataset) and the packets (well-labeled in Mirai-RGU dataset), we mark the label of the corresponding packet (malicious/benign) according to the flow's label that the packet belongs to.

For the LSTM module, we build standard LSTM cells, in which each cell consists of input gate, output gate, and forget gate. The sigmoid function is used as the activation function (α). Our proposed framework, as shown in Figure 4, consists of three LSTM layers with dropout. Total parameters of the first layer (dimension = 128, without dropout) exceed 4 million (4,194,304).

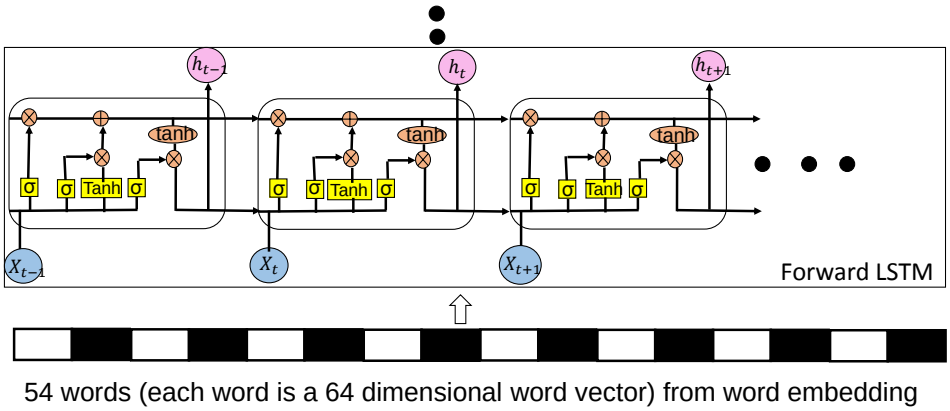

Figure 3. The illustration of the packet-word-transfer and classification module in the proposal. Packet-work-transfer module is at the input data layer.

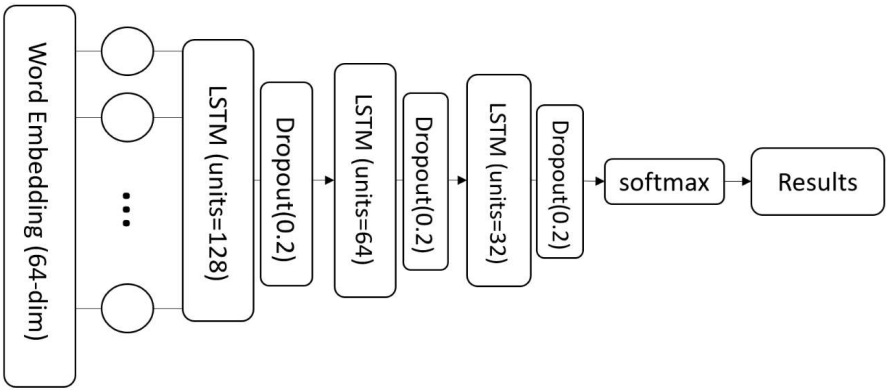

Figure 4. The illustration of the full network model.

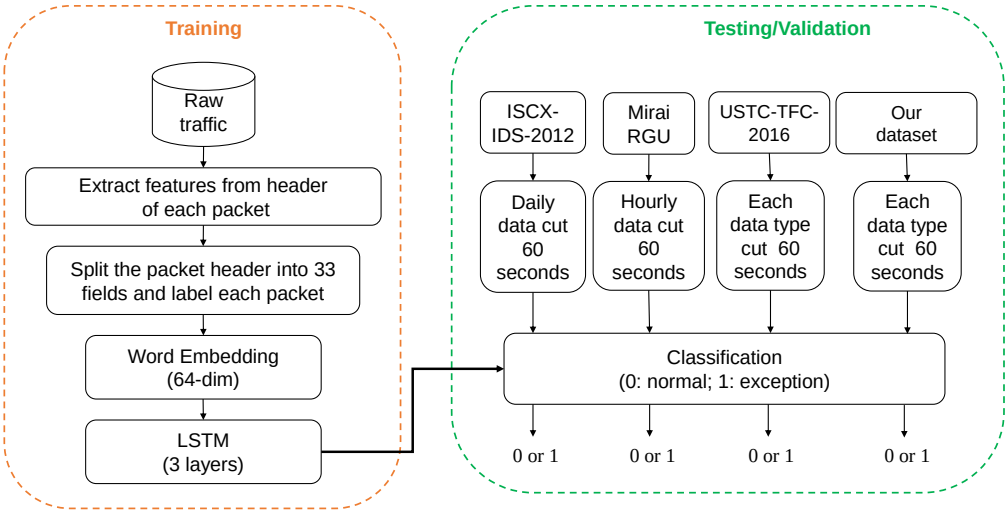

Figure 5. The illustration of the workflow of the training and testing/validation stage. Data pre-processing and packet labelling for training are done at the training stage. Training/Testing is done at the pre-processing data and with the ratio is 9:1. Validation is performed on the random samples (60 s/pcap).

The pseudo-code of our algorithm and the processing flow is illustrated in Algorithm 1. The preprocessing phase adjusts features to ensure data representation is suitable for the used algorithms, i.e., parsing the packet and converting it to the translated word. The new dataset of translated Words is in the format of the dataset of integer number. The translated dataset is then split

into two parts: Training Data and Testing Data, respectively. Training stage starts on the Training Data and running the LSTM three-layer model. Note that the dimension from input data are required to reshape, e.g., 64, before performing the training and testing. The dropout rate is flexibly set and can be tuned up to 0.5. The loss function is based on binary cross entropy and RMSProp optimizer is selected for the learning rate adjustment method. A dense output layer with softmax is added to the model. The model is then compiled with the binary cross-entropy as the loss function and the Adam optimiser over total of 200 iterations. Finally, the model is testing on the Testing Data to determine the effectiveness of the model in terms of accuracy, precision, recall, f1-score, and loss (defined later in the next section).

Algorithm 1: Algorithm for packet-based traffic classification

Data: Sequence of raw packets from network

Result: Accuracy, precision, recall, f1-score, FAR, loss

dictionary = *array*() % Create the mapping word to the index array
translatedWords = null; %variable to store the values of words to converted to integer format;
while true **do**
  Parse the packet;
  Extract 54-byte from each packet
  if packet length < 54 **then**
    Pad zeros;
  end if
  Parse each byte of the extracted data as a word; *words* defines the words extracted from the packet;
  for {$i = 1$; $i <$ count(*words*); i++} **do**
    if Word in dictionary, *dictionary*(*words*[i]) **then**
      index = *dictionary*(*words*[i]) % Pulling index of the words
    else
      dictionary[] = *words*[i]; %Update the dictionary with the new word;
      index = *dictionary*(*words*[i])
    end if
    concat(*translatedWords*, *index*)
  end for
end while% return dataset
Split Training and Test based on 90/10 (TrainingData, TestingData)
Train and Validate (TrainingData, TestingData)
Reshaping *translatedWords* to an output of 64-dimensional word vector;
Input the 64-dimensional word vector to the first layer LSTM;
Dropout;
Feedforward to the second layer LSTM;
Dropout;
Feedforward to the third layer LSTM;
Dropout;
Prepare input for mini-batch (100 packets);
Apply RMSProp optimizer;
Use softmax to output 0 or 1 to the model;
Use binary cross entropy as loss function
for *epoch* = 1; *epoch* < 200; *epoch*++ **do**
  Evaluate Loss, Validation Loss
  Evaluate Accuracy and Validation Accuracy
end for

Training and validation often cost significant time regarding the system evaluation. In fact, the dataset size after converting (transferring to integer format) can be smaller in storage than that of the original text/string as in the packets, since the integer format may cost few spaces than the text data type. However, the dimension unlikely reduces. As a result, selecting word size may dramatically affect the training time. Smaller word size reduces the size of dictionary, but also causes more training data and running time. In this research, we set the maximum word size to 2 bytes based on the consideration of the length of most fields in the packet while limiting the number of words in the dictionary to be less than 65,536.

4. Evaluation Results

Similar to most existing deep learning research, our proposed classification model has been implemented using TensorFlow/Keras (version 2.2.4, Google, Mountain View, CA, USA). All of our evaluations have performed on the GPU-enabled TensorFlow which is running on a 64-bit Ubuntu 18.04.2 LTS server with an Intel(R) Xeon(R) E5-2650 2.2 GHz processor, 32 GB RAM, and an NVIDIA RTX Tesla K80.

To perform our evaluations, we have used the ISCX2012, USTC-TFC2016, Mirai-RGU and Mirai-CCU datasets. During the training and testing stages, we try to include hundreds to thousands of thousands of packets while balancing the benign and malicious traffic. At the validation stage, we run the trained model on the packets which is extracted randomly per 60 s from the selected datasets, i.e., packets in the consecutive 60 s in the original dataset are extracted while maintaining their temporal order. At this stage, the original (real) traffic is presented to the trained model without any manipulation or balancing of two types of traffic. Since our approach aims to gain a significant reduction in processing time, we have many interests in classifying the incoming packet whether it is malicious or not, instead of considering the attack type in detail. In practice, if a malicious packet is detected by the proposed system, it can raise an alarm to the network administrator and direct the packet for some offline computational intensive traffic classification systems. Therefore, in the following section, we define several measurements to evaluate the performance of the proposed solution in the term of a binary classifier. All of these metrics are all derived from the four values found in the confusion matrix in Table 4. The traffic proportion of training/testing/validation configuration of four datasets is listed in Tables 5–8.

Table 4. Confusion matrix.

		Predicted Classs	
		Malicious	**Benign**
Ground truth	Malicious	True Positive (TP)	False Negative (FN)
	Benign	False Position (FP)	True Negative (TN)

where:

- True Positive (TP)—Attack packet that is correctly classified as an attack.
- False Positive (FP)—Benign packet that is incorrectly classified as an attack.
- True Negative (TN)—Benign packet that is correctly classified as normal.
- False Negative (FN)—Attack packet that is incorrectly classified as normal.

The accuracy (Equation (1)) measures the proportion of the total number of correct classifications:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1)$$

The precision implies the number of correct classifications that is influenced by the incorrect classification (Equation (2)):

$$Precision = \frac{TP}{TP + FP}. \quad (2)$$

The recall (Equation (3)) measures the number of correct classifications that is considered with the number of missed entries:

$$Recall = \frac{TP}{TP + FN}. \tag{3}$$

The false alarm or fair (Equation (4)) measures the proportion of benign packets incorrectly classified as malicious:

$$Falsealarmrate(FAR) = \frac{FP}{FP + TN}. \tag{4}$$

The F_1-score (Equation (5)) measures the harmonic mean of precision and recall, which serves as a derived effectiveness measurement:

$$F_1 - Score = 2 \times \frac{Precision * Recall}{Precision + Recall}. \tag{5}$$

In our experiments, based on the flow statistics in the datasets and number of labeled packets in data pre-processing, we can calculate the number of packets (including benign packets) for training and number of real attack packets in testing, as summarized in Tables 5–8 below. Due to a large amount of input data, we set the mini-batch size value at 100 and the data training with less than 200 epochs.

The performance results of our proposal for each kind of malicious traffic with 10-fold testing and 10-fold validation are shown in Tables 9 and 10, respectively. From these two tables, we can observe that our method can perform the classification with nearly 100% accuracy and precision in major cases of the security attacks, which outperforms compared to the prior works, e.g., [6,19,20]. However, since the characteristics of traffic types and attacks are unique, the performance measurements are also different in each scenario and heavily relied on the training dataset. From our experimental experience, we notice that, besides tuning the learning model and its parameters (e.g., learning rate), word-embedding and attack representation samples in datasets play a critical role in the improvement of the performance.

Table 5. Train/Test/Validate traffic proportion in the ISCX-IDS-2012 dataset.

		All	Train/Test	Validation
6/12	Benign Attack	5,947,337 26,374	26,374 26,643	average: 4148/60 s
6/13	Benign Attack	3,925,130 1,838,019	100,000 100,000	average: 1209 /60 s
6/14	Benign Attack	8,687,942 960,711	100,000 100,000	average: 5947 /60 s
6/15	Benign Attack	17,551,503 17,431,539	100,000 100,000	average: 12,746 /60 s
6/16	Benign Attack	17,260,920 49,764	50,000 49,764	average: 7580 /60 s

Table 6. Train/Test/Validate traffic proportion in the Mirai-RGU dataset.

Attack Type	All	Train /Test	Validation
syn	1,526,926	728,000	1,526,926 /10 s
ack	5,390,837	728,000	5,390,770 /50 s
http	744,991	728,000	3152 /60 s
udp	4,567,726	728,000	4,567,659 /58 s

Table 7. Train/Test/Validate traffic proportion in the USTC-TFC-2016 dataset.

Benign				Malware			
Type	**All**	**Train/Test**	**Validate (avg/60 s)**	**Type**	**All**	**Train/Test**	**Validate (avg./60 s)**
Facetime	6000	6000	6000	Tinba	22,000	22,000	729
Skype	12,000	12,000	12,000	Zeus	93,141	93,141	105
BitTorrent	15,000	15,000	15,000	Shifu	500,000	100,000	3
Gmail	25,000	25,000	25,000	Neris	499,218	100,000	896
Outlook	15,000	15,000	15,000	Cridex	461,548	100,000	34
World Of Warcraft	140,000	100,000	140,000	Nsis-ay	352,266	100,000	5617
MySQL	200,000	100,000	200,000	Geodo	250,000	100,000	12
FTP	360,000	100,000	360,000	Miuref	88,560	88,560	7
SMB	925,453	200,000	925,453	Virut	440,625	100,000	858
Weibo	1,210,060	100,000	1,210,060	Total	2,707,358	803,701	-
Total	2,908,513	1,058,000	-			-	

Table 8. Train/Test/Validate traffic proportion in our dataset.

File	No# of Pcap	All (Benign, Attack)	Train/Test	Validate(avg./60 s)
Capture_2	3 (N)	(129,178, 0)	100,000	14,100
Capture_3	1 (M)	(54,641, 393,325)	100,000	9795
Capture_4	6 (N, M)	(268,461, 518,105)	100,000	10,010
Capture_5	1 (N, M)	(67,239, 519,376)	100,000	9780
Capture_6	1 (N, M)	(66,989, 519,609)	100,000	9781
Capture_7	1 (N, M)	(67,061, 519,400)	100,000	9783
Capture_8	1 (N, M, S)	(66,801, 981,651)	100,000	9996
Capture_9	12 (N, M, G)	(72,204, 1,373,042)	100,000	10,004
Capture_10	7 (N, M, GT)	(70,457, 969,937)	100,000	9784

Table 9. Performance evaluation results of the proposed solution on the four datasets with 10-fold testing.

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)	FPR (%)
ISCX-IDS-2012	99.99	99.98	99.99	99.99	7.46×10^{-7}
USTC-TFC-2016	99.99	100	99.99	99.99	1.1×10^{-7}
Mirai-RGU	100	100	100	100	0
Mirai-CCU	99.46	99.63	99.38	99.51	0.026

Table 10. Performance evaluation results of the proposed solution on the four datasets with 10-fold validation.

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)	FPR (%)
ISCX-IDS-2012	99.97	100	99.97	99.98	0
USTC-TFC-2016	99.88	99.99	99.86	99.93	0.02
Mirai-RGU	99.98	99.99	99.95	99.97	0
Mirai-CCU	99.36	99.49	99.27	99.38	0.031

One might question the aforementioned results of our proposed DL-based framework. That is, intuitively, the proposed framework can classify malicious packets because it obtains the features of malicious packets from the training dataset. Thus, we perform a validation test in which the Mirai-RGU dataset is used for training and packets from Mirai-CCU dataset are used for validation.

Our experimental results show that, even though the training and validation stages use different datasets, the proposed framework is still able to detect malicious packets with very high accuracy. The performance results are shown in Table 11.

Table 11. Performance evaluation results of the Mirai-CCU dataset validation on the Mirai-RGU training model.

	Mirai-RGU				
Dataset	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-Score (%)**	**FPR (%)**
Mirai-CCU	97.22	96.25	98.73	97.5	0.36

4.1. Time Efficiency

Time efficiency is also an important metric measurement in our experiment. The time efficiency involves two cases: training time and detection time. The training time is defined by the total time to complete the training on the selected dataset. This time depends on the data-preprocessing, the training model (number of layers and dims) and server capacity. Therefore, it would not be fair to draw comparisons within this respect, due to differences in the hardware and software used. Through our evaluation, due to considering at the deeper level (the packet level), the amount of data (after transferring from packet information) used for training in our method is expected to be higher than the flow-based approaches. As a result, the time required for training will be higher. The time for offline training, i.e., ours, could be up to 17 h with USTC-TFC2016 at 200 epochs, in order to obtain the detection performance results as mentioned above. The bigger dataset it is, the more training time it costs. However, our method has advantages of detection time. The detection time in this study is defined as the total time (elapsed time) to perform the classification on a given testing sample, i.e., the testing dataset. Given a testing file 108 MB and our server capacity (mentioned above), our system only takes fewer than 2 s to complete the classification, no matter what type of traffic is. This achievement emboldens our approach for online monitoring since we believe that this speed can satisfy most on-demand applications and in medium networks. For the core networks, we may need more efforts to integrate this system for potential deployment, since the network speed at such nodes can be up to hundreds of gigabits per second. Note that the system at the core networks can be equipped with the much more powerful servers.

4.2. Discussion

There is a trade-off between the word size and the classification performance, i.e., accuracy and detection time. Reducing the word size can potentially increase the detection time, and training time as well, due to more words involved, longer sentence created and dimension extended. For accuracy, the performance can vary and depend on the considered attack type. Apparently, we can't scale the size too small. In our evaluation, we set the word size to two bytes as aforementioned. However, there is promising research to propose an algorithm to auto-calculate this value properly on the deployed environment, even that for specific attacks.

In addition, the approach to consider the classification at the packet level can open the door to accelerating the detection since we can schedule the packets for parallel processing. However, the explicit shortcoming is to increase the training time and resource usage (e.g., memory) due to a large number of parameters and data size putting to the training model. Balancing the factor of acceptable training time but still gaining the best classification performance is a non-trivial task and a potential research direction.

Finally, the DL-based classification approach is highly susceptible to the data poisoning attack due to its dependence on the training data. Thus far, we have found few attack models targeting the evasion of the deep learning-based malicious classification systems, including ours. However,

this can be soon changed when the popularity of deep learning will attract more attackers to exploit its vulnerabilities for hacking or monetization. Generating/preventing adversary models against deep learning is thus one of the most interesting and promising security research topics.

5. Conclusions

In this paper, we have proposed a novel packet-based malicious traffic classification framework by using the word embedding and Long Short-Term Memory (LSTM) network model. Our evaluation results show that the performance of ours is competitive with the prior work on classifying flows into benign or malicious, but with much lower flow pre-processing time. In other words, the major advantage of the proposed framework is that it does not require pre-process packets into flows, thus boosting the detection acceleration. We believe that our first attempt on using a packet-based malicious classification approach can inspire the research community to consider the optimization methods and exploit the advantages of deep learning to build effective IDSs without suffering significant detection delay.

Author Contributions: Conceptualization, R.-H.H. and M.-C.P.; Methodology, R.-H.H. and Y.-L.C.; Data curation, Y.-L.C.; Formal analysis, R.-H.H. and Y.-L.C.; Investigation, R.-H.H., M.-C.P., V.-L.N. and Y.-L.C.; Resources, R.-H.H.; Software, Y.-L.C.; Validation, R.-H.H., V.-L.N. and Y.-L.C.; Visualization, V.-L.N. and Y.-L.C.; Writing—original draft preparation, Y.-L.C.; Writing—review & editing, R.-H.H., V.-L.N. and Y.-L.C.; Supervision, R.-H.H.; Project administration, R.-H.H.; Funding acquisition, R.-H.H.

Funding: This research was supported in part by the Ministry of Science and Technology of Taiwan, ROC, under Grants MOST 107-2218-E-194-014.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies, New York, NY, USA, 3–5 December 2016; pp. 21–26.
2. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 11–22. [CrossRef]
3. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware Traffic Classification Using Convolutional Neural Networks for Representation Learning. In Proceedings of the 31st International Conference on Information Networking, Da Nang, Vietnam, 11–13 January 2017; pp. 712–717. [CrossRef]
4. Chen, Y.C.; Li, Y.J.; Tseng, A.; Lin, T. Deep Learning for Malicious Flow Detection. In Proceedings of the IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, Montreal, QC, Canada, 8–13 October 2017.
5. Yuan, X.; Li, C.; Li, X. DeepDefense: Identifying DDoS Attack via Deep Learning. In Proceedings of the IEEE International Conference on Smart Computing, Hong Kong, China, 29–31 May 2017. [CrossRef]
6. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
7. Li, C.; Wang, J.; Ye, X. Using a Recurrent Neural Network and Restricted Boltzmann Machines for Malicious Traffic Detection. *NeuroQuantology* **2018**, *6*, 21954–21961. [CrossRef]
8. Hwang, R.H.; Peng, M.C.; Huang, C.W. Detecting IoT Malicious Traffic based on Autoencoder and Convolutional Neural Network. *IEEE Globecom Conf.* **2019**, submitted.
9. Radford, B.J.; Apolonio, L.M.; Trias, A.J.; Simpson, J.A. Network Traffic Anomaly Detection Using Recurrent Neural Networks. *arXiv* **2018**, arXiv:1803.10769.
10. Cui, J.; Long, J.; Min, E.; Mao, Y. WEDL-NIDS: Improving Network Intrusion Detection Using Word Embedding-Based Deep Learning Method. In Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence, Palma de Mallorce, Spain, 15–18 October 2018; pp. 283–295.

11. Kim, M.S.; Kong, H.J.; Hong, S.C.; Chung, S.H.; Hong, J.W. A flow-based method for abnormal network traffic detection. In Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium, Seoul, Korea, 23 April 2004; pp. 599–612.
12. Fu, R.; Zhang, Z.; Li, L. Using LSTM and GRU neural network methods for traffic flow prediction. In Proceedings of the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, Wuhan, China, 11–13 November 2016; pp. 324–328.
13. Goldberg, Y.; Levy, O. word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv* **2013**, arXiv:1402.3722.
14. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
15. Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E.; Ahmadi, M. Microsoft Malware Classification Challenge. 2018. Available online: https://arxiv.org/pdf/1802.10135.pdf (accessed on 29 May 2019).
16. McDermott, C.D.; Majdani, F.; Petrovski, A.V. Botnet Detection in the Internet of Things using Deep Learning Approaches. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
17. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
18. Mikolov, T.; tau Yih, W.; Zweig, G. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; pp. 746–751.
19. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
20. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* **2018**, *6*, 1792–1806. [CrossRef]