Journal of Computers Vol. 32 No. 1, 2021, pp. 65-82
doi:10.3966/199115992021023201006

Behaviour Classification of Cyber Attacks Using Convolutional Neural Networks

Wen-Hui Lin, Ping Wang*, Hsiao-Chung Lin, Bao-Hua Wu, Jeng-Ying Tsai

Department of Information Management, Kun Shan University, Tainan 710, Taiwan, ROC
linwh@mail.ksu.edu.tw, pingwang@mail.ksu.edu.tw, fordlin@mail.ksu.edu.tw, weq498aa@gmail.com, Jamestsay1207@gmail.com

Received 9 November 2019; Revised 17 May 2020; Accepted 24 June 2020

Abstract. Most existing proposals are invariably based on the assumption that defence mechanisms can filter malicious connections. This assumption cannot be guaranteed in practical applications. Remote connections generally bypass the firewall and virus detection engines by using legal network protocols, such as http, ICMP, and SSL; once connected, clients can upload malicious applications to the host. Defenders require an efficient network detection approach that can quickly learn new network behavioural features for detecting network intrusions. Deep learning (DL) can utilise enhanced features based on behaviour patterns extracted from intrusion detection datasets. Accordingly, this study focuses on network intrusion detection by using LeNet-5 model with back propagation incorporating ID3 decision tree scheme for feature reduction. In the study, behavioural feature selection, image matrix transformation, and weight comparison were used to classify network threats. The experimental results indicated that the prediction accuracy of threat classification increased with an increase in the size of the data sample (N). The prediction accuracy of intrusion detection increased up to 96.02% for six subcategories with N ≥ 10,000 and 93.75% for 39 subcategories with $N \geq 500$. The overall accuracy rate was 94.89%.

Keywords: intrusion detection, deep learning, convolutional neural networks, LeNet-5, behaviour features

1 Introduction

Most existing approaches to detect cyber-attacks involve using cyber threat analyses for matching potential attack profiles and filtering malicious connections so that defenders can effectively address network threats. Generally, malicious attempts to compromise network security involve phishing websites or operating system updates using legal network protocols, such as http, ICMP, and SSL, to bypass firewalls and virus detection engines. This enables remote intruders to upload malicious applications to hosts. In practice, discriminating between legitimate and malicious connections by using protocol analysis is difficult. The problem of detecting the traffic anomaly of possible threats on attack vectors by collecting and analyzing forensic data while being configured to monitor for, identify, and manage security threats over the Internet is referred to as the threat detection and response (TDR) problem [1].

Automatically detecting suspicious flow connections for preventing malicious threats is a challenging task for online cyber security systems. Numerous machine learning (ML) schemes have been proposed to assist defenders detect traffic anomalies in information flows and distinguishing by accumulating information on recognised attacks for the TDR problem, especially associated with the information gain (IG) approach. [2-3] The ML schemes provide a means of identifying the recognised attacks and countermeasure them in real cyber-attack cases. However, the difficulties in intrusion detection mainly arise from two aspects: (i) notable variations provide opportunities in network behaviour for new

* Corresponding Author

malicious cyber threats. (ii) discriminating between legitimate and malicious connections by using network protocol analysis is difficult. Defenders must improve the threat classification accuracy and decrease the false positive rate (FPR) for threat detection in the network intrusion detection system (NIDS). In other words, defenders cannot only rely on the MA-based scheme to classify cyber threats without incorporating new features that might lead to a high false positive rate (FPR) for detecting malicious connections.

Notably, deep learning (DL) with multilayer neural networks exhibited superior results in some intrusion detection applications [4-8]. Practically, a convolutional neural network (CNN) is used to cluster data into categories according to the classification weights. CNNs can minimise the classification error and maximise the generalisation ability of learning by using convolutional feature extraction, pooling, activation, calculations, operations, and optimisation algorithms.

In developing the proposed model, there are three important aspects of focusing on our work: (i) features were reduced according to the ID3 decision tree theory to speed up the learning of the normal and intrusive patterns, (ii) minimise the classification error and maximise the generalisation ability of learning by using convolutional feature extraction, pooling, activation, calculations, operations, and optimisation algorithms, and (iii) both forward propagation and back propagation learning processes were optimised to obtain accuracy values (%) associated with optimal weights of hidden layers with error correction between the estimated value and the actual output.

Finally, we summarized the technical achievements of this work as follows.

(i) In this study, a novel and accurate model for threat classification based on behaviour learning incorporating both the IG and the LeNet-5 [9] was developed for network anomaly detection to reduce the FPR in solving the TDR problem. (ii) Information flows of network intrusion were obtained from two sources: data sources: the NSL-KDD archive dataset [10] and suspicious local network flows captured by the National Center for High-performance Computing (NCHC), and these flows were compared and categorised using cluster analysis to determine the behavioural features of different IPs. (iii) The feature vectors were transformed into feature matrices in binary format, which formed CNN input images for accurately categorising cyber threats according to the collected data. (iii) The proposed scheme can enable the defence system to respond promptly to high-risk security concerns.

The remainder of this paper is organised as follows. Section 2 reviews the previous studies in this field and LeNet-5 model. The proposed network intrusion detection model with LeNet-5 architecture is introduced in Section 3. Section 4 presents a performance analysis of the results. Section 5 draws the conclusions.

2 Background and Related Work

This section presents methods of behaviour classification in the NIDS with enhanced features using DNNs and LeNet-5 model.

2.1 Behaviour Classification Schemes for Intrusion Detection Using DNNs

CNNs are used for multiclass classification, where an image is classified into one of the N identity classes of behaviour classification. The advantages of CNNs are: (1) They can deal with large quantities of training data. (2) CNNs can automatically learn features to capture complex visual variations by leveraging large quantities of training data. (3) In the testing phase, CNNs can be easily parallelised on GPU cores to obtain results within a short time. (4) Classifying the training images into numerous identities can enable the FCLs of DNNs to form rich identity-related features [9].

The user authentication problem is the most frequently encountered problem for feature extraction and training with ML techniques. Many classification approaches incorporate layered CNNs with classification algorithms for behaviour clustering and classification [11-23]. Behaviour classification involves classifying the behaviour into one of N identity classes by using a test set of distinct network flows, which include both normal and abnormal flows, in the learning process to measure the predictive accuracy of network intrusion detection.

Automatically detecting suspicious flow connections for preventing malicious threats is a challenging task for online cyber security systems. Numerous behaviour classification techniques comprising CNNs with optimised algorithms have been used for achieving high-precision network intrusion detection.

These behaviour classification schemes are summarised in Table 1.

Table 1. Behaviour classification approaches for network intrusion detection

Scheme	Features & Achievement	Limitations
Yan, Yuan, Xu, et al. (2013) [11]	1. Developed an intrusion detection method based on traffic features, such as the average packet length, peak traffic, and peak botnet flow during different periods, to detect IRC botnet channels. 2. A clustering algorithm based on the similarity between traffic features was analysed for botnet channel detection. An extensive evaluation of CNN-based botnet channel detection systems was conducted through large-scale network flow classification.	In the feature extraction phase, it lacks of describing the selection criterion of behaviour features, it might lead to different accuracies by using various feature sets for real suspicious network flows.
Tan (2013) [12]	1. The network traffic records were observed for our proposed DoS attack detection system, which was developed according to a widely used dissimilarity measure called the earth mover's distance (EMD). 2. The system achieved a detection accuracy of 99.95% with 10-fold cross-validations for the KDD Cup 99 dataset and 90.12% for the ISCX 2012 IDS evaluation dataset with a processing capacity of approximately 59,000 traffic records per second.	If the EMD-based model cannot adequately capture the underlying best features of the data, underfitting would occur in supervised machine learning.
Abuadlla [2014] [13]	1. A two-stage neural NIDS based on flow data was proposed for detecting and classifying attacks in network traffic. 2. Both multilayer and radial basis function (RBF) neural networks were used to compare the performance, memory consumption, and time required for network training. 3. The experimental results demonstrated that the designed models were promising in terms of accuracy and computational time and had a low probability of false alarms.	1. Engineers achieved these results by directly learning all features, with manually filtering, unpacking, or categorising binary files. 2. The pre-processing of flows needs a semi-or an automatic supporting tool to assist defenders deal with huge amount of suspicious flow data.
Saxe and Berlin (2015) [14]	1. Introduced an approach that addressed these issues and described in considerable detail the DNN-based malware detection system to achieve a usable detection rate with an extremely low FPR and scale to real world training example volumes on commodity hardware. 2. The proposed system achieved a 95% detection rate with a 0.1% FPR on the basis of more than 400,000 software binaries sourced directly from customers and internal malware databases.	1. In the evaluation phase, the experimental results used a fixed amount of training sample and testing data that overfitting or underfitting occur tending to have poor predictive performance. 2. To evaluate the proposed detection system, the k-fold cross-validations might be used for performance evaluations.
Niyaz, Sun, Javaid, and Alam (2015) [15]	1. They proposed a DL-based approach for implementing an effective and extensive NIDS by using self-taught learning on the NSL-KDD dataset to better learn multilevel features. 2. Validated the performance of the proposed approach and compared the approach with a few previous studies on metrics such as accuracy, precision, recall, and F-measure values.	In the feature extraction phase, the training features were obtained from only one data source, i.e., the NSL-KDD dataset [10], these features might be different from real suspicious network flows from recent local networks.

Table 1. Behaviour classification approaches for network intrusion detection (continue)

Scheme	Features & Achievement	Limitations
Tomiyama, Yamaguchi, Shimada, Ikuse, and Yagi (2016) [16]	1. Trained a recurrent neural network (RNN) to extract features of process behaviour and then trained the CNN to classify the feature images that were generated from the extracted features in the trained RNN. 2. Experiments indicated that the satisfactory CNN-FRS performance was due to the fusion of multiple CNNs and metric learning.	The trained RNN is a state-of-the art deep learning algorithm used for modeling sequential information. However, sometimes it can not remember information for longer periods of time, i.e., long-time dependency problem in the input sequences.
Tang *et al*. (2016) [17]	1. Applied a DL approach for flow-based anomaly detection in an SDN environment. Built a DNN model for the intrusion detection system and trained the model with the NSL-KDD dataset. 2. Only six basic features (that can be easily obtained in an SDN environment) were selected from the 41 features of the NSL-KDD dataset.	In the feature extraction phase, the training features were obtained from only one data source, i.e., the NSL-KDD dataset [10], these features might be different from real suspicious network flows from recent local networks.
Ding, Xu (2017) [18]	1. Proposed a novel mixed framework aggregating time convolution network (TCN) to recurrent neural network (RNN) design based on encoder-decoder architecture to learn and memorize long-term events after the encoding process. 2. The experimental results of three common action segmentation data sets prove that the proposed model proves superior performance to that of RNNs.	1. The model accuracy relies on huge amounts of data to train the algorithms of TCN. 2. When the TCN model cannot adequately capture the underlying features of the data, underfitting can occur.
Sharafaldin, Lashkari, Ghorbani (2018)[19]	1. This study generates a publicly network intrusion system (IDS) data set, namely the CICIDS2017 dataset for testing and performance comparison. In the CICIDS2017 dataset, it contains seven common network attacks and normal connection patterns. 2. Also uses a learning algorithm to completely filter network traffic characteristics and conduct a comprehensive evaluation, exploring the best set of network behavior characteristics for detecting certain attack categories.	1. Training the deep learning algorithms on a huge dataset that might be too large to fit in memory of host. 2. A host with relatively high computational capabilities to train the algorithms.
Bai, Kolter, Koltunn (2018) [20]	Proposed a novel temporal neural network design with multiple hidden layers (convolutional layer and pooling layer) for large-scale parallel processing, namely TCN, and proved the network training and converge time were shorter than that of RNNs with the same model accuracy.	1. This method needs a complete sequential dataset to capture the underlying features of the flow data. 2. Suitable used in classifying the detection intrusion case with long temporal sequential data or a complete of dataset.
Shone, et al. (2018) [21]	1. Proposed a non-symmetric deep auto-encoder (NDAE)(deep-learning) model and evaluated using the benchmark KDD Cup ’99 and NSL-KDD datasets. 2. Classification accuracy for 5-class on NSL-KDD with 41 features is 85.42% 3. Classification accuracy for 13-class on NSL-KDD is 89.22%	1. Only 5-class (i.e., 4 categories of threats and normal) was classified in the experiment. 2. Two training dataset were out of date and some malicious behavior of traffic might be not included.

Table 1. Behaviour classification approaches for network intrusion detection (continue)

Scheme	Features & Achievement	Limitations
Vinayakumar *et al*. (2019) [22]	1. Proposed a highly scalable and hybrid DNNs framework called scale-hybrid-IDS-AlertNet which can be used in real-time to effectively monitor the network traffic and host-level events to proactively alert possible cyberattacks. 2. The DNN model which performed well on KDDCup 99 is applied on other datasets, such as NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017, to conduct the benchmark.	1. The proposed system does not give detailed information on the structure and characteristics of the malware. 2. Overall, the performance can be further improved by training complex DNNs architectures on advanced hardware through distributed approach.
Lin, *et al*. (2019) [23]	1. Proposed a network intrusion detection scheme for DDoS threats using TCNs with network flow analyzer and CICFlowMeter-v4.0 to classify the network threats thru behavior feature analyses. 2. The prediction accuracy of intrusion detection goes up to 95.77% for model training with N= 50,000 for sizing (N) of samples using the IDS dataset CIC-IDS-2017.	TCN-based detector be suitable used in classifying the detection intrusion case with long temporal sequential data or a complete of dataset, e.x, NSL-KDD, UNSW-NB15, CICIDS 2017 and CICIDS 2018.

2.2 LeNet-5 model

In 1998, LeCun et al. [9] proposed a neural network architecture for handwritten character recognition denoted as LeNet-5. Typically, the LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by one or more fully-connected layers (FCLs) and finally a softmax classifier as shown in Fig. 1. In [24], the behavioural features of intrusion detection application are confined to a region of space of the size of 20 x 20 in binary format for the IRC botnet threat classification (Fig. 1). Also, malicious code variants detection is implemented using CNNs incorporating uncompressed gray-scale image in [25]. That is, by a straightforward and simple architecture such as LeNet-5 is sufficient to categorize cyber threats according to the collected feature sets.

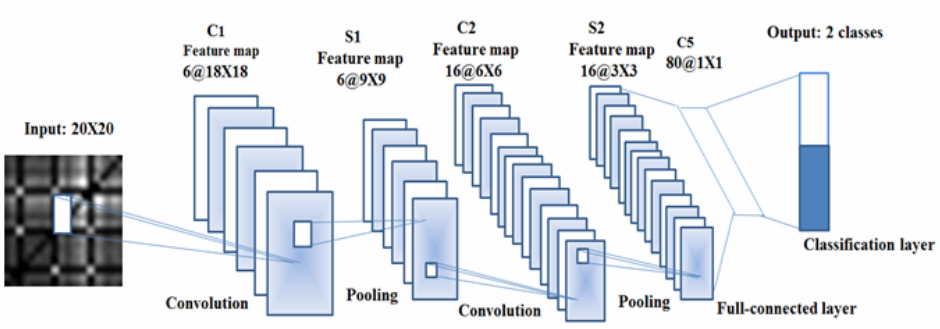

Fig. 1. Application of CNNs for the malware classification [24]

In the developed model, the feature vectors were transformed into feature matrices, which formed LeNet-5 input images for accurately classify cyber attacks. Specifically, LeNet-5 uses a cascade of numerous layers of nonlinear processing units for feature extraction and transformation and exhibited superior results in image recognition applications. Then, cluster data into categories according to the classification weights. Consequently, the proposed model can minimise the classification error and maximise the generalisation ability of learning by using convolutional features.

Classification is the most frequently encountered issue in ML techniques. Classification problems based on LeNet-5 model are illustrated as follows.

Consider a given training dataset $D(x_i, y_i)$, where x_i denotes the number of observations of a sample ($x_i \in R^N, i = 1,...,n$) and y_i indicates the class to which the point x_i belongs. y_i, $i = 1,...,n$, y_i is assigned to each observation x_i. Each feature x_i is of dimension *d*, which corresponds to the number of propositional variables.

$$f_\theta = \{(x_i, y_i) \mid x_i \in R^n, y_i \in \{0, 1, ... m\}\}_{i=1}^n, \tag{1}$$

Generally, the algorithm goal for proposed model in the CNN is to minimize the estimation error between inputs and outputs, i.e.,

$$\arg_\theta \ Min \ \ C = -\sum_{i=1}^{n}([\hat{y}^{(i)} \log y^{(i)} + (1 - \hat{y}^{(i)}) \log(1 - y^{(i)})]), \tag{2}$$

For LeNet5 model, cost function is defined by the cross-entropy loss function as

$$\arg_\theta Min \ \ C = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}\| \hat{y}^{(i)} - y^{(i)}) \|^2) + \frac{1}{2}\sum_{l=1}^{N}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ij}^l)^2. \tag{3}$$

In the following, the weights are updated using the gradient descent method as Eq.(4). It calculates the gradient of the cost function in Eq.(3) with respect to the neural network's weights, i.e, $\Delta w_{ij}(t) = -\eta \frac{\partial C}{\partial w_{ij}}$. If the update for the gradient of the cost function is less than the low limit ε, i.e, $\frac{\partial C}{\partial w} < \varepsilon$, or when the maximum execution cycle (epoch) is reached, the convolution kernel update is stopped and output the classification results. It calculates the gradient of the error function with respect to the neural network's weights. The calculation proceeds backwards through the network iteratively.

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial C}{\partial w_{ij}}, \tag{4}$$

where η is the learning rate of model learning.

Then, a mapping function $f_\theta(x_i) = W \circ x_i + b$ is defined for classification as follows:

$$\hat{y}^{(i)} \cong y^{(i)} = f_\theta(x_i) = s(W \cdot x_i + b), \tag{5}$$

where $\hat{y}^{(i)}$ is the final class of the output, $y^{(i)}$ represents the output result of the training process, *W* represents the convolution kernel of the feature matrix, *b* is the bias of the feature vector, f_θ represents a mapping function, and *s* represents the softmax function.

3 An Analysis Model for Threat Classification with LeNet-5 model

To solve the problem of detecting network intrusion after the collection of suspicious network flows, this study proposes a behaviour classification model involving CNNs with back propagation based on the LeNet-5 model. In the proposed model, behavioural feature selection, transformation, and comparison can be used to identify network intrusions with the TensorFlow toolset. The detailed flowchart for behaviour classification with the revised LeNet-5 model is illustrated in Fig. 2. Fig. 2 shows the three subphases in the behaviour classification process: (1) the feature extraction and reduction phase, (2) feature training phase, and (3) model validation phase.

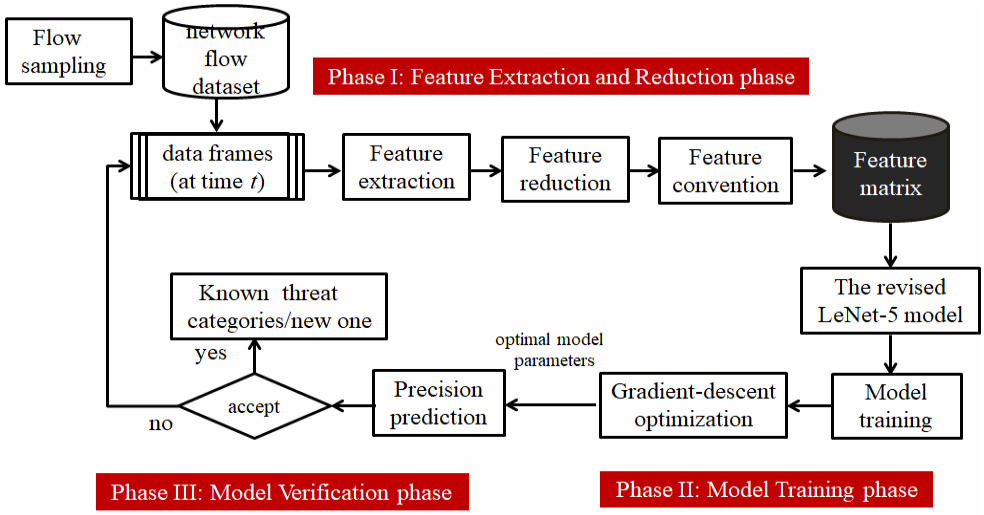

Fig. 2. Basic concept of network intrusion detection with the revised LeNet-5 model

In Fig. 2, three crucial steps were involved in developing the proposed model: (i) *Feature extraction and reduction*: Behavioural feature analyses were performed by using the flow analyzer tool with sampling network data, then features were reduced according to the ID3 decision tree theory to speed up the learning of the normal and intrusive patterns, (ii) *Feature conversion*: reduced feature set are converted to feature matrix in binary format, and (iii) *Model training and validation*: network attacks were classified and the classification accuracy of the revised LeNet-5 classifier was determined by using a behaviour-based classifier. Finally, network intrusions were detected by using behaviour classification to defend against cyber threats.

3.1 Feature extraction and reduction

In the feature extraction phase, the training sample data were obtained from two data sources: (i) well-known intrusion detection archive: the NSL-KDD dataset [10] derived from KDDCUP99 dataset [26] and (ii) real flow data: suspicious network flows captured by the NCHC for extracting new behavioural features.

Step 1.1 Experiment data sources

First, a series of experiments were performed to investigate the effectiveness of the LeNet-5 classifier by using the experiment dataset which contains 41 features, from which 34 are numeric and seven are symbolic or discrete. The experiment dataset comprises four major types of network attacks and 39 attack categories, such as probing (ipsweep, nmap, portsweep, and satan), DoS (back, land, neptune, pod, smurf, and teardrop), R2L (ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, and warezmaster), and U2L (buffer_overflow, load_moudle, perl, and rootkit).

Step 1.2 New behaviour features extracted from recent network flows

Real flow data from edge network routers were captured by the NCHC and used for discovering new behavioural features of recent cyber threats. The Tcpreplay (http://tcpreplay.appneta.com/) and Wireshark tools were used for simulating attack scenarios, which were recorded as streamline files (Pcaps). In this study, the Pcap analyser was downloaded from GitHub [27] to analyse the list of suspicious connections and distinguish crucial features of malicious network connections by comparing the features of the normal and abnormal flows. This analysis helps defenders identify behavioural features through the filtration of source and destination IPs and their ports, DNS requests, web requests, mail traffic, and query packets, as illustrated in Fig. 3.

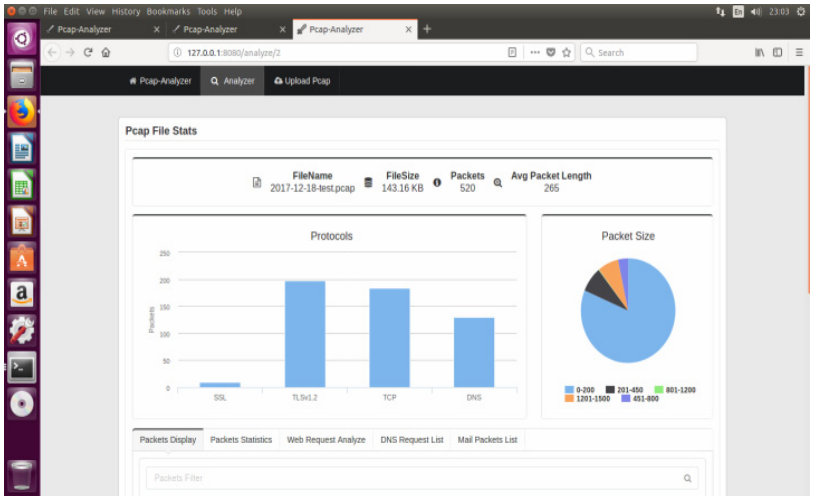

Fig. 3. Importing testing files to the Pcap analyser

Step 1.3 Feature reduction

A reduction in the behavioural features increases the threat detection speed; however, it may also increase the false alarm rate. Finding a reduced set of features for creating a predictive model requires experimentation and knowledge regarding the problem that you wish to solve. For building the model, the subset of selected features in our dataset was determined using the ID3 decision tree theory algorithm. In this algorithm, the values of attributes are represented by branches, and the attributes are arranged as nodes according to their ability to classify data. The ID3 algorithm begins with the original set S as the root node. In each iteration of the solution procedure, the algorithm repeats every unused attribute of the set S and calculates the IG of that attribute. The algorithm enables the defenders to select the attribute with the largest IG value. Finally, the set S is split on the selected attribute to produce subsets of the data.

Let the IG of attribute A be represented as IG(A), which is the measure of the difference in entropy values before and after the set S is split on attribute A. Thus, the reduction in the uncertainty after the splitting of set S on attribute A is given as follows [25]:

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t), \qquad H(S) = -\sum_{t \in T} p(t)\log_2 p(t). \tag{6}$$

where $H(S)$ is a measure of the degree of uncertainty in the dataset S, represents the subsets created by splitting set S on attribute A, $p(t)$ is the ratio of the number of elements in t to the number of elements in S, and $H(t)$ is the entropy of subset t. Equation 6 illustrates the acceleration in the learning of the normal and intrusive patterns when IG(A) is used to screen out the additional features for training the CNN classifier. ID3 does not guarantee an optimal solution and can get stuck in the local optima. It uses a greedy approach to select the best attribute for splitting the dataset in each iteration. Backtracking can be used during the search for the optimal decision tree to improve the ID3 algorithm.

Step 1.4 Feature conversion

In the following, the feature vectors were transformed into feature image (i.e., image matrix transformation) which formed input images of LeNet-5 model for accurately categorising cyber threats. To recognize the feature differences between malicious connection and normal application in binary format [6, 23], we use the feature similarity matrix (S) to represent the similarity degree of behaviour features using the Mahalanobis Distance formula to reveal the correlation between packet payload features. The feature similarity matrix (S) is expressed as an m x m square matrix where the diagonal vector is 1 and the symmetric vectors of S (sjk) is shown as follows.

$$S(x_i)=[s_{j,k}]_{mxm}=\begin{cases} s_{jk} & j\neq k \quad j,k=1,...,m \\ 1 & j=k \quad j,k=1,...,m \end{cases}=\begin{bmatrix} 1 & s_{12} & ... & s_{1m} \\ s_{21} & 1 & ... & s_{2m} \\ ... & ... & ... & ... \\ s_{m1} & s_{m2} & ... & 1 \end{bmatrix}_{mxm} . \quad (7)$$

Where $s_{jk}=1-[(E_j-E_k)^p]^{1/p}$. Notably, Eq.(7) is equivalent to the Euler distance formula when p=2 and $s_{jk}=1-\sqrt{(E_j-E_k)^T(E_j-E_k)}$. Then the B2M (binary mapping to image) algorithm in [24] was implemented to form the feature map for model training. As shown in Fig. 4, the feature map generated by S from normal application and malware is significantly different. Thus, it can be used for image classification later.

(a) Visualization of feature matrix of bot (b) Visualization of feature of normal APP

Fig. 4. Feature map generated by feature similarity matrix

3.2 Model training and classification

For the model training phase, the LeNet-5 model designed by LeCun et al. in 1998 was revised, as illustrated in Fig. 5 and Table 2. The proposed model incorporated the gradient-descent optimisation algorithm (i.e., adaptive delta algorithm) to fine-tune the model parameters for reinforced learning. The algorithm involves using the back propagation error derivatives and learning rate of all the layers. Thus, the classification was used to determine the learning error of multilayer neural nets. The weights of the neural nets were adjusted to minimise the learning error.

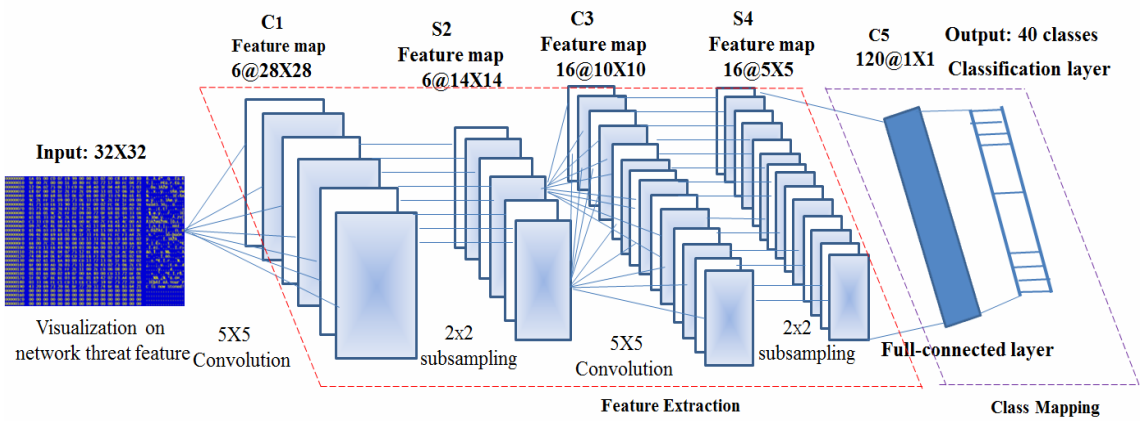

Fig. 5. Revised LeNet-5 model for network threat classification

Table 2. Architecture of the revised LeNet-5 model

Layer	Function description
Convolution layer C_1	A 5×5 pixel filter was applied to three 32× 32 pixel input images derived from the image matrix of the behavioural features and six 28 × 28 pixel feature maps were generated.
Pooling layer S_2	The pooling layer S2 on the six 28×28 pixel feature maps generated six 14 × 14 pixel pooled feature maps (1/2 the size).
Convolution layer C_3	A 5×5 pixel filter was applied to six 14×14 pixel input images. This generated 16 feature maps of size 10×10 pixels.
Pooling layer S_4	The subsampling layer S4 used input feature maps of size 10×10 pixels and generated 16 feature maps of size 5×5 pixels.
FCL C_5	1920 filters of size 5×5 pixels were applied to convert 16 feature maps of size 5×5 pixels into an FCL of size 120 × 1, that is, the FCL contained 120 neurons.
Classification layer	Case I output: The number of classes was 6 (1-5 for the different threat types and 0 for normal). Case II output: The number of classes was 40 (1-39 for the different threat types and 0 for normal).

The learning results were used as a basis for model parameters for model validation such as the weight matrix, batch_size, batches_per_epoch, epochs, and classification accuracy.

3.3 Model validation phase

A cross-validation scheme was adopted to evaluate the intrusion detection accuracy. The overtraining problem was overcome by using different fold values for the cross-validation scheme. For example, a fold value (k) of 10 indicates that 90% of the collected data are used in the training experiment, and the remaining 10% of the data are repeated 10 times for testing. In the model validation phase, the system had a quick response for threat classification by using weights of neural nets in the trained LeNet-5 model.

The detailed algorithm for behavioural classification with LeNet-5 model described by PDL in Fig. 6.

Input: model parameters of the revised LeNet-5 model (RLN5) including the batch set, feature matrix W, learning rate l, and test data

Output: weight matrix and predicted accuracy of classification

RLN5 Training Algorithm

1. Initialize the model parameters of the model
2. Set the initial value of the parameter batch_size to 32, batch_per_epoch to 64, and epochs to 10
3. Set the initial values of parameter learning_rate to 0.1
4. Assign the stop condition value (ε) as 0.0001
5. Training loop
6. While (the number of epoch iterations) do
7. While (the number of batches) do
8. Perform the feature training, as given in Eqs. 1-4
9. Perform the data classification, as given in Eq. 5
10. Return (model_file)
11. The training results of the model_file include: (1) the number of iterations, (2) final cost of the object function (C), (3) feature matrix (W), (4) output result of the training process ($y^{(i)}$), and (5) final class of the output ($\hat{y}^{(i)}$)
12. Return train (output_file)
13. End loop
14. End loop
15. Test phase
16. Accuracy prediction by using specific parameters from the model_file
17. return predict (accuracy)
18. End

Fig. 6. Algorithm RLN5

4 Experimental Results

In this section, the applicability of the proposed CNN-based security analysis model is demonstrated by using two cyber security examples. The experiments were conducted using the Python programming language and TensorFlow, which is an open source software library for numerical computations involving data flow graphs. Moreover, TensorFlow incorporates numerical libraries such as Keras, Openpyxl, NumPy, and OpenCV for DL computation. The parallelisation of the multicore architecture increased the computation speed of the CNN model. The multicore architecture included an Intel i7-6700 processor (3.4 GHz * 8) with 32 GB RAM, a 64-bit Ubuntu 14.04 operating system, an Nvidia GeForce GTX 1080 graphics card (GPU), graphics core computing, and the MongoDB 2.2.6 database. The experimental environment is described in Table 3.

Table 3. Experimental environment for CNN-based security monitoring

IP	Programming language	Numerical library
192.168.0.12-14	Python 3.5	Tensorflow-gpu 1.1.3 keras v2.2.4 openpyxl numpy 1.1.8 python-opencv

4.1 Two Experiments

Two experiments were conducted to test the proposed scheme by using simulated cyber-attack conditions to classify various threats with CNNs.

Step 1. Feature extraction phase

For a quick classification response to the cyber threats, the first example incorporated a revised LeNet-5 model with the experiment dataset. The LeNet-5 model comprised one or more convolutional layers followed by FCLs, similar to typical artificial neural networks where models attempt to classify entire images into 40 classes.

Step 1.1 Feature reduction

Equation 6 was applied to the experiment dataset, and the features were ranked according to their IG score. As shown in Fig. 7, the appropriate feature subset could be determined by referring to the number of selected features. From Fig. 7, the threshold values for the CNN classifier training were selected at the turning point, where the IG approached a stable condition. Thus, the number of features selected (32 at IG ≥ 0.119) depended on the statistical characteristics of the behavioural features. Therefore, most network attacks could be identified by selecting a set of reduced features and comparing their threshold values.

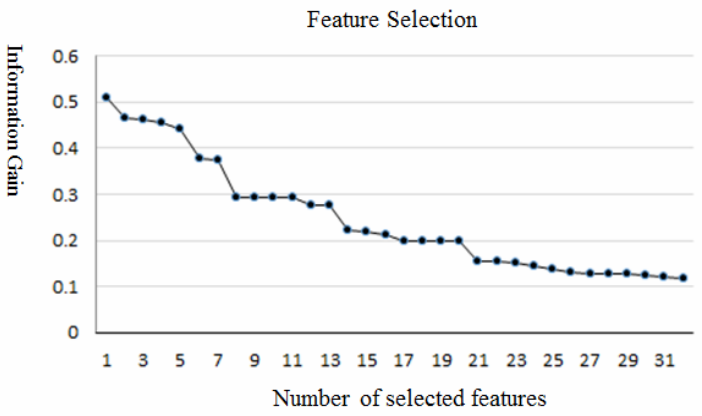

Fig. 7. Relation between the IG (y-axis) and number of selected features (x-axis)

The top 32 reduced features ranked according to their IG score are listed in Table 3. Generally, fewer features for comparison allow for faster attack detection; however, this also increases false alarms. In this study, the behavioural patterns (Table 4) of 32 intrusion features were identified using behavioural analysis for conducting the training experiment.

Table 4. Top-32 features ranked according to their IG score

NO.	Feature	IG	Rank	NO.	Feature	IG	Rank
26	srv_serror_rate	0.508	1	3	Service	0.221	11
25	serror_rate	0.464	2	11	num_failed_login	0.220	12
4	Flag	0.462	3	22	is_guest_login	0.212	13
12	logged_in	0.456	4	37	dst_host_srv_diff_host_rate	0.20	14
39	dst_host_srv_serror_rate	0.442	5	35	dst_host_diff_srv_rate	0.197	15
30	diff_srv_rate	0.377	6	8	wrong fragment	0.197	16
38	dst_host_serror_rate	0.374	7	14	root_shell	0.1974	17
6	dst_bytes	0.292	8	10	Hot	0.292	18
5	src_bytes	0.292	9	34	dst_host_same_srv_rate	0.292	19
29	same_srv_rate	0.276	10	31	srv_diff_host_rate	0.276	20

NO.	Feature	IG	Rank
33	dst_host_srv_count	0.156	21
27	rerror_rate	0.153	22
23	count	0.150	23
9	urgent	0.145	24
2	protocol_type	0.139	25
41	dst_host_rerror_rate	0.131	26
36	dst_host_same_src_port_rate	0.129	27
32	dst_host_count	0.129	28
17	num_file_creation	0.128	29
18	num_shells	0.123	30
13	num_compromised	0.120	31
10	num_root	0.119	32

Step 1.2 Converting features to an image matrix

This step involved the pre-processing of the experimental data and included the following: (1) the symbol conversion of the network packets, (2) normalisation of the numeric data, and (3) conversion of features to an image matrix. As shown in Fig. 8, the B2M (Binary mapping to image) algorithm for the experiment was implemented. Finally, these packets were normalised to a size of 32 × 32 pixels and were used in the experiments.

```
def standard_function(data):
    min_data = np.min(data, axis=1).reshape(len(data), 1)
    max_data = np.max(data, axis=1).reshape(len(data), 1)

    data = (data - min_data) / (max_data - min_data)
    return data


def euclidean_distance(data_flag, data):
    # Create diagonal array and img_data size.

    diag_data = np.empty((len(data), len(data[0]), len(data[0])))
    img_data = np.zeros_like(diag_data)
    for i in range(len(data)):
        diag_data[i] = np.diag(data[i])

    # Euclidean function.

    for img_flag in range(len(data)):
        if not img_flag % 10000:
            print(data_flag, img_flag)
        for i in range(len(diag_data[img_flag, 0])):
            minus_data = diag_data[img_flag] - diag_data[img_flag, :, i]
            for j in range(len(minus_data)):
                img_data[img_flag, j, i] = 1 - \
                    np.sqrt(np.sum(minus_data[j] ** 2))
    return img_data
```

Fig. 8. Program of B2M algorithm using Euler distance formula

In this study, 80% of the data was randomly selected as the training dataset and the remaining 20% of the data was used for testing. The datasets together contained 39 threat subcategories to avoid a classification bias. The threat classification according to a statistical analysis of the training dataset is displayed in Fig. 9.

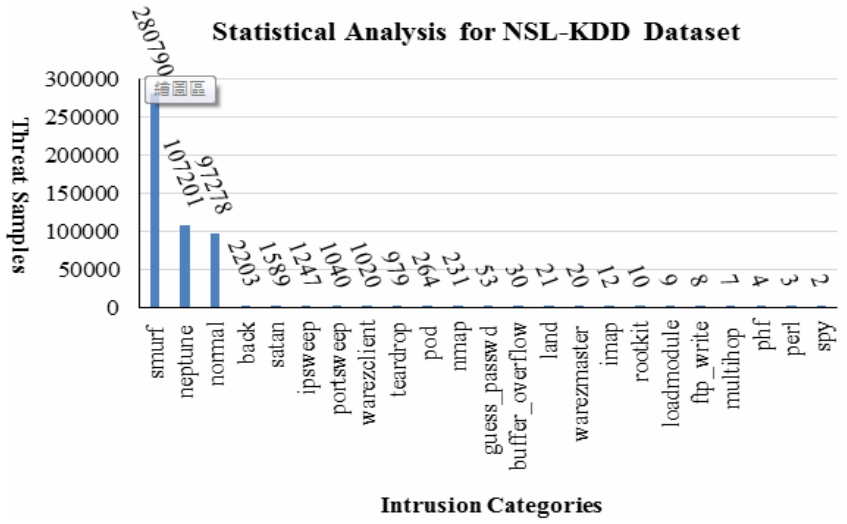

Fig. 9. Threat classification according to a statistical analysis of the NSL-KDD dataset (y-axis)

After completing the statistical analysis of the experiment dataset, experiments were conducted for model training to verify the accuracy of different training samples. First, 32 intrusion features were selected to perform the following model training experiment. In the experiment, the cost function is cross-entropy, the iteration parameters batch_size = 64, epoch = 10, as shown in Table 5.

Table 5. The selected parameter in training model

Parameter in the proposed model	Value used in the proposed model
Input image size	(32 x 32, Gray_level)
Loss in object function	Categorical cross entropy
Optimizer	Adaptive delta
Learning_rate	0.1
Batch_per_epoch	64
Epoch	10

Step 2. Model learning phase

The model parameters were obtained from two types of training samples, namely the legitimate network connection and a malicious connection. Two cases were analysed to verify the effectiveness of the LeNet-5 classifier as follows.

Case I. Classification of the major types of intrusion threats by using numerous samples

The major intrusion threats were classified by selecting attack types that had numerous samples (i.e., more than 10,000 samples of captured data) in the training process. The statistical analysis of the threat types for the experiment dataset is represented in Table 6. Only six major types of threats were screened to be classified in the experiment.

Table 6. Statistical analysis of the threat types for the experiment dataset (> 10,000 records)

Attacks	Kddcup.data.corrected	No. Of Record	Training data	Test data
Smurf	2807886	15000	12000	3000
Neptune	1072017	15000	12000	3000
Normal	972781	15000	12000	3000
Satan	15892	15892	12713	3178
Ipsweep	12481	12481	9984	2496
portsweep	10413	10413	8330	2082

In the experiment, the revised LeNet-5 was trained to detect network intrusion from the behavioural patterns of the collected samples. The softmax function was used for evaluating the model. The cross entropy function was used to adjust the learning rate for increasing the speed of the training process. The adaptive delta optimisation algorithm was employed for minimising the classification error. The prediction accuracy was 96.02% (on average) when k =2~10, as indicated in Table 7.

Table 7. Accuracy associated with different folds of the cross-validation scheme (Case I)

k-fold	Prediction accuracy (%)
k=2	93.57%
k=3	95.27%
k=4	96.52%
k=5	96.23%
k=6	96.22%
k=7	96.63%
k=8	96.82%
k=9	96.65%
k=10	96.30%
Average	96.02%

Case II. Classification for the 39 attack subcategories

In the experiment, the revised LeNet-5 model was used for classifying the data into 39 subcategories according to their behaviour. In the experiment dataset, the number of samples for a specific threat type was generally less than 500. The dataset had to be evenly and randomly replicated to 500 samples to avoid a learning bias due to a small number of samples. Thus, revising each dataset to an equal number of samples for model training can prevent the precision rate of the model from being decreased to 70% or lower when defenders use multiple unbalanced training datasets.

After the proposed model had been trained, it was used to distinguish threats by using the trained weights of neural nets. The proposed model parameters were obtained from two types of training samples, namely the legitimate network connection and a malicious connection. The prediction accuracies (%) for the cross-validation method (k = 2~10) are listed in Table 8.

Table 8. Accuracy associated with different k-hold of the cross-validation scheme (Case II)

k-fold	Prediction accuracy (%)
k=2	87.19%
k=3	89.62%
k=4	95.07%
k=5	94.96%
k=6	95.12%
k=7	95.44%
k=8	95.51%
k=9	95.47%
k=10	95.40%
Average	93.75%

For Case II, the average accuracy of the proposed classifier was 93.75%.

Step 3. Model validation phase

In the experiment, the size of the testing dataset was varied to examine the classification accuracy. The experimental results are displayed in Fig. 10. Fig. 10 reveals the classification accuracy and number of true positives and true negatives for different sizes of the testing dataset. The experimental results indicated that the classification error decreased as the size of the testing dataset increased. Therefore, the prediction accuracy of threat classification increased with an increasing N value. The prediction accuracy of intrusion detection increased up to 96.02% for six subcategories with $N \geq 10{,}000$ and 93.75% for 39 subcategories with $N \geq 500$. The overall accuracy rate was 94.89%. Thus, the prediction accuracy of threat detection was higher with 32 important features than those of the SVM-based intrusion model with all 41 features and 34 important features in [28].

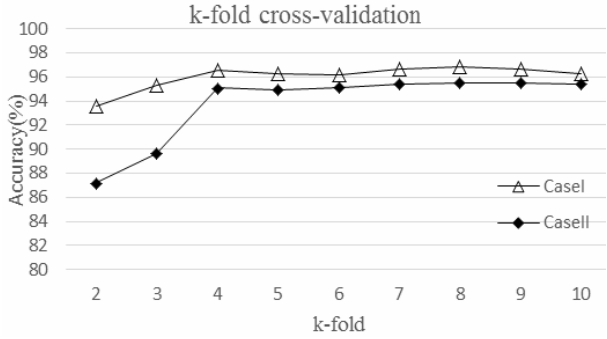

Fig. 10. Threat classification accuracy with different sample sizes

4.2 Method Comparisons

Table 9 compares the major scheme features, accuracy and limits of the method proposed in this study with those of the method proposed by [12, 15, 17, 21-22]. In the proposed scheme, behaviour patterns derived from two distinct sources of threats, i.e., global threats (NSL-KDD) and local threats in Taiwan (NCHC). In further, the detailed classification of attacks in most studies was neglected, except in [15]. In practice, defenders need to realize the detailed attack types of threats for defensive count- measure.

Table 9. Method comparison

	Scheme feature	Training Data (behaviour of traffic flows)	Classification Accuracy (%)	Limitations
Proposed scheme	Feature extraction: Wireshark Feature selection: ID3 Classification: LeNet-5	1. Global threat :NSL-KDD 2. Local threat: extract new behavioural features of suspicious network flows captured by the NCHC in Taiwan	1. Accuracy for 7-class (6 categories of threats and normal) with 32 features is 96.02% 2. Accuracy for 40-class (i.e., 39 subcategories of attacks and normal) with 32 features is 93.75%.	To filter and extract the exact features from the huge amount of suspicious flow data, manual tool such as Wireshark and Tcpreplay are used.
Tan (2013) [12]	Feature extraction: none Feature selection: EDM Classification: Dissimilarity measure.	1. Global threat: KDD Cup 99 +ISCX 2012 2. Local threat: none	1. Training accuracy for 7-class (6 categories of threats and normal) on KDD Cup 99 with 32 features is 99.95% 2. Training accuracy on ISCX 2012 is 90.12%	Lacks of the experiments for detailed classification of threats.
Niyaz, Sun, Javaid, Alam (2015) [15]	Feature extraction: none Feature selection: information gain Classification: two-stage process of ANN	1. Global threat: NSL-KDD 2. Local threat: none	1. over 98% accuracy for 2, 5, 23 classes of attacks on NSL-KDD with 23 features 2. 86% testing accuracy for 2, 5 classes of attacks.	Behaviours of traffic flows were only extracted from global threats, i.e., NSL-KDD dataset
Tang et al. (2016) [17]	Feature extraction: none Feature selection: none Classification: DNN	1.Global: NSL-KDD 2.Local: none	1. Accuracy for 5-class on NSL-KDD with 6 features is 75.75%	Only 5-class (i.e., 4 categories of threats and normal) was classified in the experiment.
Shone, et al. (2018) [21]	Feature extraction: none Feature selection: none Classification: stacked auto-encoder DNN	1. Global threat: KDD Cup 99 + NSL- KDD 2. Local threat: none	1. Accuracy for 5-class (4 categories of threats) on NSL- KDD associated with 41 features is 85.42% 2. 89.22% accuracy for 13-class in NSL- KDD	Only 5-class was classified in the experiment.

Table 9. Method comparison (continue)

	Scheme feature	Training Data (behaviour of traffic flows)	Classification Accuracy (%)	Limitations
Vinayakumar, et al. (2019) [22]	Feature extraction: Tcpdump Feature selection: information gain Classification: DNN	1. Gobal threat: NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, CICIDS 2017 2. Local threat: none	1. Accuracy for 7-class in KDD Cup 99 associated with 41 features (DNN 1 layer ~ 5 layers) is 95~99% 2. Accuracy on UNSW-NB15 is 65~75% 3. Accuracy for WSN-DS is 65~75%	1. In the experiment, only 5-class was classified for KDD Cup 99. 2. 8-class was classified for UNSW-NB15 3. 5-class was classified for WSN-DS

The overall accuracy of the scheme for 7-class classification proposed in this study was observed to have a higher precision than that of the scheme in [17, 21]. Moreover, using selected 32 features to perform the intrusion detection allows the system to provide superior execution efficiency of the system than 41 features used in [21-22]. Compared to 7-class classification accuracy in [12, 17, 21-22], the validation for 40-class (39 types of attacks and normal) was conducted in our experiment that reveals the actual attack type of network threats. Thus, the present approach provides superior flexibility for analyzing the traffic anomaly from two different data sources and is therefore more effective at performing intrusion detection in practical applications.

5 Conclusions

This paper presents an intrusion detection model that incorporates a LeNet-5 classifier with ID3 algorithm for feature selection to enhance the precision and the execution speed of the model. The proposed approach minimises the classification error by using back propagation error derivatives and the quick intrusion detection by only using 32 features compared to existing schemes in [21-22]. For a small number of samples on specific threats, this study also improved the prediction accuracy of intrusion detection by using augmentation strategy of samples by being randomly replicated to 500 samples to avoid a learning bias. Overall, the results indicate that the precision of the proposed model for classification of detailed attack types (e.x., 40-class) is higher than that of existing schemes.

A future study will include a large number of experiments to obtain insights into the accuracy with false-positive rate by using ROC curve and analyze the computational speed of the proposed system. Moreover, using Temporal Convolution Networks (TCNs) to learn the complex behavioral characteristics of malicious attacks from suspicious flows is a challenge task of intrusion detection in the following study.

Acknowledgments

This work was supported jointly by the Ministry of Science and Technology of Taiwan under Grant Nos. MOST 108-3116-F-168-001-CC2, and MOST 108-2410-H-168-003.

References

[1] Microsoft Corporation, Threat Analysis & Modeling, v2.1.2, 2007. <http://www.microsoft.com/en-us/download/details.aspx?id=14719>, 2018 (accessed 02.01.18).

[2] S. Mukherjeea, N. Sharma, Intrusion detection using Naive Bayes classifier with feature reduction, Procedia Technology 4(2012)119-128.

[3] S. Chebrolu, A. Abraham, J. P. Thomas, Feature deduction and ensemble design of intrusion detection systems, Computers & Security 24(4)(2005) 295-307.

[4] G. Kou, G. -M. Tang, S. Wang, H.-T. Song, Y. Bian, Using deep learning for detecting BotCloud, Journal on Communications 36(11)(2016) 1-7.

[5] J.-E.Yan, C.-Y. Yuan, H.-Y. Xu, et al., Method of detecting IRC botnet based on the multi- features of traffic flow, Journal on communications 34(10)(2013) 49-64.

[6] Z.-Y. Tan, Detection of denial-of-service attacks based on computer vision techniques. Sydney: University of Technology, 2013.

[7] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanović, Flow-based anomaly intrusion detection system using two neural network stages, Computer Science and Information Systems 11(2)(2014) 601-622.

[8] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in : Proc. of 2015 10th International Conference on Malicious and Unwanted Software (MALWARE ’15) (2015) 11-20.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11)(1998) 2278-2324.

[10] University of New Brunswick, NSL-KDD dataset for network–based intrusion detection systems, <https://www.unb.ca/cic/datasets/nsl.html/>, 2018 (accessed 02.03.18).

[11] J. -E.Yan, C. -Y. Yuan, H. -Y. Xu, et al.(2013), Method of detecting IRC botnet based on the multi- features of traffic flow, Journal on communications 34(10)(2013) 49-64.

[12] Z. Y. Tan, Detection of denial-of-service attacks based on computer vision techniques. Sydney: University of Technology, 2013.

[13] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanović, Flow-based anomaly intrusion detection system using two neural network stages, Computer Science and Information Systems 11(2)(2014) 601-622.

[14] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE ’15) (2015) 11-20.

[15] Q. Niyaz, W. Sun, A. Y. Javaid, M. Alam, 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 (2015) 21-26.

[16] S. Tomiyama, Y. Yamaguchi, H. Shimada, T. Ikuse and T. Yagi, Malware detection with deep neural network using process behavior, The 40th IEEE Computer Society International Conference on Computers, Software & Applications. Atlanta, Georgia (2016) 577-582.

[17] T. A Tang, L. Mhamdi, D. McLernon, et al., Deep learning approach for network intrusion detection in software defined networking, In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (2016)26-29. DOI: 10.1109/WINCOM.2016.7777224.

[18] L. Ding, C. Xu, TricorNet: A hybrid temporal convolutional and recurrent network for video action segmentation (2017) arXiv: 1705.07818.

[19] I. Sharafaldin, A. Lashkari, and A. Ghorbani, Toward Generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), (2018) 108-116.

[20] S. Bai, J. Z. Kolter, V. Koltun,, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling (2018) arXiv: 1803.01271.

[21] N. Shone, T. N. Ngoc, V. D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1)(2018) 41-50.

[22] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, Deep learning approach for intelligent intrusion detection system, in IEEE Access 7(2019) 41525-41550.

[23] W. -H. Lin, P. Wang, K. -M. Chao, C. -C. Lo, B. -H. Wu, M. -S. Jhou, Behaviorial-based network flow analyses for anomaly detection in sequential data using temporal convolutional networks, The 16th IEEE International Conference on e-Business Engineering (ICEBE 2019), Shanghai, China (2019) 173-183.

[24] G. Kou, G.-M. Tang, S. Wang, H.-T. Song, Y.Bian, Using deep learning for detecting BotCloud, Journal on Communications 36(11)(2016) 1-7.

[25] X.-G. Han, W. Qu, X.-X. Yao, C.-Y. Guo, F. Zhou. Research on malicious code variants detection based on texture fingerprint. Journal on Communications, 35(8)(2014)125-136.

[26] University of California, Irvine, KDDCUP99 dataset, 1999, <http://kdd.ics.uci.edu/ databases/ kddcup99/kddcup99.html>, 2018 (accessed 19.02.18).

[27] GitHub, Pcap analyzer, available at https://github.com/f18m/large-pcap-analyzer.

[28] S. Zaman, F. Karray, Features selection for intrusion detection systems based on support vector machines, In: Proc. of the 6th IEEE Conference on Consumer Communications and Networking Conference (CCNC'09), 2009.