Intrusion Detection by Deep Learning with TensorFlow

Navaporn Chockwanich, Vasaka Visoottiviseth

Faculty of Information and Communication Technology, Mahidol University,
999 Phuttamonthon 4 Rd., Salaya, Nakhon Pathom, Thailand
navaporn.chk@student.mahidol.ac.th, vasaka.vis@mahidol.edu

Abstract— **Nowadays intrusion detection systems (IDS) plays an important role in organizations since there are a ton of cyber attacks which affect to security issues: confidential, integrity, availability. Currently, there are many open source tools for intrusion detection but they have different syntax of rules and signatures which cannot be used across different tools. In this paper, we propose an intrusion detection technique by using deep learning model which can classify different types of attacks without human-generated rules or signature mapping. We apply the supervised deep learning technology which are RNN, Stacked RNN, and CNN to classify five popular types of attacks by using Keras on the top of TensorFlow. Our technique requires only the packet header information and does not need any user payload. To verify the performance, we use MAWI dataset which are pcap files and compare our results with Snort IDS. Due to the lack of user payloads, the results show that Snort could not detect the network scan attack via ICMP and UDP. Meanwhile, we prove that RNN, Stacked RNN, and CNN can be used to classify attack for Port scan, Network scan via ICMP, Network scan via UDP, Network scan via TCP, and DoS attack with high accuracy. RNN delivers the highest accuracy.**

Keywords— **Intrusion detection / Deep learning / TensorFlow/ Machine learning / Traffic classification**

I. INTRODUCTION

To protect organizations from cyber attacks, intrusion detection system (IDS) plays an important role. Open source tool for intrusion detection system can perform protocol analysis to detect attacks by using rules and signatures. Well-known open source IDS [1] tools include Snort, Suricata and Bro. However, these open source tools have different syntax of rules and signatures which cannot be used across different tools unless it is a common rule set for using together among various tools. Each open source IDS tool has the different capability to detect the different types of attack. Many researchers studied the detection performance of open source IDS [2]-[3]. They found that each IDS has distinct advantages and disadvantages such as the detection rate of the specific type of attack, false positive, etc.

Currently, deep learning approach is used to solve many problems such as speech and audio recognition, social network filtering, machine translation, identifying suspicious payment transactions, predicting vehicle crash, predicting delays and bottlenecks in production of the complex supply chain and drug design, etc. Deep learning also has been used in cybersecurity area, such as classifying threats and distinguishing anomaly behavior. There are also many open source software libraries for deep learning. In this paper, we select TensorFlow [4], which is one of an open source software library introduced by Google to perform deep learning and deep neural networks.

In this paper we propose the intrusion detection by applying the deep learning technique to detect attack without human-defined rules or signatures. We develop an intrusion detection system by using deep learning with TensorFlow. Moreover, according to McAfee Labs Threats Report in 2016 – 2017 [5]-[8], top list of network attacks include DoS and scan attacks with maximum 35%. Therefore, we select DoS and scan attacks as our target. We evaluate the accuracy and performance of intrusion detection system by comparing with the legacy Snort IDS for five types of attack: DoS, port scan, network scan via UDP, network scan via TCP and network scan via ICMP.

There are three contributions in this paper:

- We use the deep learning technique to detect attack without human-defined rules or signatures.
- We use the deep learning technique to classify attacks without accessing information in the packet payload to avoid a breach of data privacy.
- We compare the detecting time between the deep learning model and Snort IDS.

II. FUNDAMENTAL KNOWLEDGE AND RELATED WORK

This section describes the fundamental knowledge related to this work including IDS, the deep learning mechanism and tools. The related works are also mentioned in this part.

A. Snort IDS

Snort has been the de facto open-source IDS/IPS solution [9]. Due to Snort has light-weight design and flexible deployment options, Snort's userbase rapidly increase in the recent years (up to 400,000 currently) [10].

Our paper considers the performance comparison with the Snort IDS. Snort software tasks and components are performed sequentially packet by packet. With a huge amount

of traffic, buffers will be full, and incoming packets may be dropped. Snort also has a long processing time since it will access not only packet headers but also user payloads before alerting or dropping malicious packets.

As shown in Figure 1, the basic outline of Snort rule consists of two parts which are header and options [11].

Rule Header							
action	protocol	Source IP	Source port	->	Destination IP	Destination port	(Rule options)

Figure 1. Basic Outline of a Snort rule

Snort repositories provide more than 20,000 rules obtained from the popular Sourcefire and Emerging Threats (E.T.) repositories. In this paper, we use emerging-dos.rules and emerging-scan.rules for Snort to detect DoS and Scan attack.

B. Deep learning

Deep learning was introduced according to the progress of feature learning research, the availability of large scale of labeled data, and hardware [12]. In 2006, G. Hinton initiated the representation learning research with the idea of greedy layer-wise pre-training plus fine-tuning of deep belief networks. This resulted in higher performance than state-of-the-art algorithms on MNIST handwritten digits recognition and document retrieval. Later on, many deep learning algorithms were proposed and successfully applied to many domains.

The typical models include Autoencoder (AE), Deep Belief Network (DBN), Convolutional Neural Network (CNN) [13] and Recurrent Neural Network (RNN) [13]. While AE and DBN are unsupervised learning models, CNN and RNN are supervised learning models. In this paper, we consider using supervised learning models, which are CNN, RNN, and stacked RNN [14]. An introduction of each model is described as below.

1)Convolutional Neural Network (CNN): It is an advanced model used in many fields. Continuous convolution function is an operation that blends two functions occurring on time. To conceptualize this operation as an algorithm, the equation can be explained in the following steps:

1) Flip the signal: it is the ($-\tau$) part of the variable.

2) Shift it: it is given by the t summing factor for $g(\tau)$.

3) Multiply it: it is the product of f and g.

4) Integrate the resulting curve: it is the less intuitive part because each instantaneous value is the result of an integral.

2)Recurrent Neural Network (RNN): It is a sequence model of neural networks. RNN has the property of reusing information already given. RNN is simply a neural network layer, which takes the input, and the previous state as inputs, applies the tanh operation, and outputs the new state.

3)Stacked Recurrent Neural Network (Stacked RNN): It enhances abilities of RNN. Each cell of Stacked RNN can store more information throughout the hidden states between the input and output layer.

C. TensorFlow

TensorFlow is opensource software developed by Google. The main application of TensorFlow is machine learning and Deep Neural Networks based on data flow graphs. TensorFlow can be installed on Ubuntu, macOS, and Windows. A user can run TensorFlow with CPU support or GPU support [15].

D. Keras

Keras [16] is high-level API for neural networks, written in Python and can run on top of TensorFlow and support both CPU and GPU. It focuses on enabling fast experimentation. We will use Keras on top of TensorFlow to develop deep learning models for intrusion detection.

E. Related work

Research work [17] proposed the design of a Hybrid Intrusion Detection System using Snort and Hadoop. The primary purpose of this work is to integrate Snort with Hadoop and auto-generate new Snort's rule for better performance of detection. This work limits the attack type as ICMP attack, Smurf attack, SYN flood attack, UDP attack, and Port scanning. They found that snort rules can be generated by adding options for event filtering in such a way that the alerts will be created if the number of packets from the source exceeds a particular amount. New rules were generated with the output of the analysis done by Hadoop. The new snort rules created were efficient in finding ICMP attack, Smurf attack, SYN flood attack, UDP attack, and Port scanning.

Sasanka Potluri et al. [18] evaluated the performance of the detection mechanism by combining the deep learning techniques with the machine learning techniques. They used Theano deep learning library with MATLAB. They used NSL-KDD DATASET as the input data. There are DoS, Probe, R2L, and U2R. They evaluated the performance of the hybrid deep learning approach in combination with stacked autoencoders, DBN, softmax regression, and SVM. The result shows that the model that gave the best detection accuracy is stacked autoencoders with SVM.

Zhang, et al. [19] created a semi-supervised model by exploiting a small number of labelled data with a large amount of unlabeled data for intrusion detection. At the first stage, information gain-based feature selection is applied to the NSL-KDD datasets to reduce the redundant features. At the second stage, the new training dataset is used to train LapSVM based learning model and the best accuracy 97.8% is obtained.

Yin, et al. [20] propose a deep learning approach for intrusion detection using recurrent neural networks (RNN-IDS) using NSL-KDD dataset. They also study the performance of the model in binary classification and multiclass classification, and the number of neurons and different learning rate impacts on the performance of the proposed model. Machine learning algorithms such as J48, Naive Bayesian, Random Forest, Multi-layer Perceptron, Support Vector Machine and other are used to train models through the training set by Weka. The

results show that the model has a higher accuracy on the KDDTest+ when there are 80 hidden nodes in the RNN-IDS model; meanwhile the learning rate is 0.5, and the training is performed 80 times.

Vinayakumar [21] evaluated on the essential synthetic ID data set such as KDDCup 99. Comprehensive analysis of various MLP, CNN, CNN-RNN, CNN-LSTM, and CNN-GRU with its topologies, network parameters, and network structures are used to select the optimal network architecture. The models in each experiment are run up to 1000 epochs with learning rate in the range [0.01-05]. CNN and its variant architectures have significantly performed well in comparison to the standard machine learning classifiers. This is mainly due to the reason that CNN can extract high-level feature representations that represent the abstract form of low-level feature sets of network traffic connections.

III. PROPOSED WORK

In this paper, we will develop intrusion detection by using deep learning with TensorFlow to detect five types of popular attacks, which are DoS, port scan, network scan via UDP, network scan via TCP and network scan via ICMP. The proposed system will be evaluated and compared according to the performance matrix.

A. Dataset

We use MAWILab'2017 [22] dataset for our experiments. It consists of a set of labelled anomaly traffic in the MAWI archive packet traces from WIDE Internet backbone, which is a traffic data repository maintained by the MAWI working group of the WIDE project in Japan. This dataset contain data sniffed from sample point-F with 1 Gbps of bandwidth. This dataset contains traces representing 15 minutes daily of traffic captured from a trans-Pacific link between the United States and Japan. Traffics are captured in pcap format by tcpdump command; and IP addresses in the traces are scrambled by a modified version of tcpdpriv to anonymize IP addressed to protect the user privacy. Packet payloads are omitted, since the captured packet size is limited to 96 bytes. The labels are obtained using an advanced graph-based methodology that compares and combines different and independent anomaly detectors. MAWI 2017 dataset contains a lot of normal traffic at random. Then we must separate normal traffic out of attack traffic and balance input data size to ensure that input data include enough attack data for training the model. We use the dataset of January 2017 which has total original file size around 100 GB. We select only pcap files containing attacking traffic which are DoS and Scan. The summary of the dataset for the experiment is as shown in Table 1.

B. Data Pre-processing

In this step, we do pre-processing by performing steps as follows. We split the large pcap file and extract only attacking traffic to a new pcap file to reduce the data amount. According to the label information provided on the MAWILab, the attack labels are marked per data flow. Thus, we have to process the pcap file to Bro [23] and SiLK [24] to convert the packet information to the flow information. The feature we use is based on information that we can gather from Bro's Conn.log and SiLK. Conn.log file is the logging of general information regarding TCP, UDP, and ICMP traffic. We also define additional features by combining features which we can gather from SiLK. The additional features are mean packet length, packets per sec, and byte per sec.

TABLE 1. LIST OF NETWORK ATTACK TYPES WITH LABELS

Attack Class	label	Total file size of attack (Bytes)	Number of attacks	Number of attack packet	Number of attack flows
Port scan	1	1,081,424	113	12,211	12,615
Network scan ICMP	2	3,292,852	146	43,561	35,329
Network scan UDP	3	276,760,828	2,196	3,381,017	2,834,493
Network scan TCP	4	2,508,779,552	4,902	26,602,287	4,904,974
DoS	5	11,742,876	123	119,533	61,315

There are 16 features in total as shown in Table 2. We must convert data about IP address into decimal, and scale all data into [-1,1]. After that, we prepare a set of input data by selecting 12,000 of attacking flow from each attack type. We use 10-fold cross-validation to train and test our models.

TABLE 2. LIST OF FEATURES

No.	Feature	Definition	Tool
1	sIP	Source IP address	SiLK
2	dIP	Destination IP address	SiLK
3	sPort	Source port	SiLK
4	dPort	Destination port	SiLK
5	proto	IP protocol	SiLK
6	packets	Packet count	SiLK
7	bytes	Byte count	SiLK
8	missed_bytes	Indicates the number of bytes missed in content gaps (packet loss).	Bro
9	Mean packet length	Bytes /packets	-
10	Packets per sec	Packet count / Duration	-
11	Byte per sec	Byte count / Duration	-
12	sTime	Starting time of flow)in sec(	SiLK
13	durat	Duration of flow (in sec)	SiLK
14	eTime	End time of flow)in sec(	SiLK
15	conn_state	State of connection	Bro
16	history	Records the state history of connections.	Bro

C. Dataset Evaluation Criteria

Performance of Intrusion Detection will evaluate based on the following criteria.

- Precision (P): true positive (TP) divided by sum of true positive and false positive (FP).
- Recall (R): TP divided by sum of TP and FN.
- F1-score: Two multiply with P and R, then divide by sum of TP and FN.
- Accuracy: Sum of TP and TN divided by sum of TP, TN, FP, and FN.

- CPU Usage: To measure CPU time, we observe from VM instances dash board of Google cloud.
- Processing Time: the amount of time that use for detecting the attack. For our model, we measure the duration of start time and end time of a testing process. For snort, we measure from the start time and end time of processing the pcap files.

IV. IMPLEMENTATION

This part describes processes to develop an intrusion detection system by using Deep Learning. As in Figure 2, firstly we have to preprocess the MAWI dataset by using Bro and SiLK to convert the packet to the flow and extract features. Then, we scale all data into [-1,1]. After that, we develop an intrusion detection system by using deep learning. Finally, we find the accuracy of intrusion detection system based on deep learning comparing with Snort to evaluate the detection efficiency and performance for DoS and Scan attack. Note that, in the performance comparison we used snort version 2.9.11.1 GRE (Build 268) with the ruleset from EmergingThreats Open optimized for Snort-2.9.0-enhanced [25].

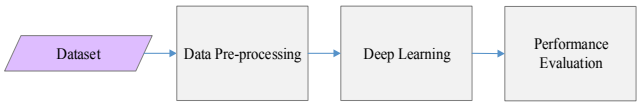


Figure 2. Proposed work's architecture

A. Environment settings

To implement deep learning, firstly, we install TensorFlow on Google Cloud with Ubuntu operating system and implement TensorFlow through Anaconda with GPU support. The detail of machine specifications is as shown in Table 3.

TABLE 3. MACHINE SPECIFICATIONS

Factors	Setting
Machine type	n1-standard-4 (4 vCPUs, 15 GB memory)
GPUs	1 x NVIDIA Tesla K80
Boot disk and local disks	400 GB (Standard persistent disk)
OS	Ubuntu 16.04
TensorFlow version	1.10.0
Python version	3.6.5
Jupyther notebook	5.6.0
Keras	2.2.2

Moreover, we define the parameters for each model as shown in Table 4. The meaning of each parameter is described as follow:

1)Epoch: It is the number of times that learning algorithm will work through all the training dataset. The example of the number of epochs set to 10, 100, 500, and larger [26].

2)Batch*:* It is a hyperparameter that defines the number of samples to work before update the model parameters as same as a for-loop [26].

3)Activation function*:* It is a node added to the output end of neural network, it is also known a transfer function [27].

4)Loss function: Loss functions map a set of parameter values for the network onto a scalar value that indicates how well those parameters accomplish the task the network is intended to do [28].

5)The Optimizer: A base class provides methods to compute gradients for a loss and apply gradients to variables. A collection of subclasses implements classic optimization algorithms such as GradientDescent and AdamOptimizer [29].

TABLE 4. PARAMETERS SETTINGS

Model	Parameter	Value
RNN	Epoch	100
	Batch	512
	Activation function	softmax
	Loss function	categorical_crossentropy
	Optimizer	Adam
Staked RNN	Epoch	100
	Batch	512
	Activation function	softmax
	Loss function	categorical_crossentropy
	Optimizer	Adam
CNN	Epoch	100
	Batch	512
	Activation function	Relu, softmax
	Loss function	categorical_crossentropy
	optimizer	Adam

B. Deep learning model

We implement deep learning by using Keras on the top of Tensorflow to build deep learning model. We selected supervise deep learning including RNN, stacked RNN, and CNN. The detail for each model is mentioned as follow:

1) Recurrent Neural Network: A simple RNN. There are three layers. First, simple RNN layer which has output shape (None,80) and 6,560 parameters (weights) in this layer. Second, dense layer has output shape (None,6) and 486 parameters. Lastly, activation layer has output shape (None,6).

2) Stacked Recurrent Neural Network: there are four layers. First simple RNN layer has output shape (None, 16, 50) and 2,600 parameters in this layer. Second, simple RNN layer has output shape (None, 50) and 5,050 parameters in this layer. Third, dense layer has output shape (None,6) and 306 parameters. Lastly, activation layer has output shape (None,6).

3) Convolutional Neural Network: There are 14 layers. For 1-10 layer, there is output shape (None, 1, 1, 14) and 210 parameters in each layer. In flatten layer, there is output shape (None, 14) and 15,000 parameters. In dense_3 layer has output shape (None, 14) and 15,000 parameters. For dropout layer, there is output shape (None, 1000). Lastly, dense_4 layer has output shape (None,6) and 6,006 parameters.

V. EXPERIMENT RESULTS AND DISCUSSION

To validate the model, we used 10-fold cross validation with 60,000 samples in total: 54,000 samples for training and

6,000 samples for testing. We also use the same dataset for Snort to evaluate the performance. We enabled rules related to TCP, UDP, ICMP in emerging-dos.rules and emerging-scan.rules of Snort's rules. Then, we send pcap files to Snort to assess the detection performance comparing to RNN, Stacked RNN, and CNN models. The results according to the evaluation criteria are as shown in Table 5. We found that for the port scan and DoS attacks, all model except Snort provide 1.00 for precision, recall, and f1-score. For Network scan via ICMP, all three models provide 1.00 for precision, recall, and f1-score but Snort could not detect at all. For a network scan via UDP and TCP, stacked RNN gives the highest for all precision, recall, and f1-score.

TABLE 5. PRECISION, RECALL, AND F1-SCORE OF EACH ATTACK TYPE

Result	Model	Attack Type				
		Port scan	Network scan ICMP	Network scan UDP	Network scan TCP	DoS
Precision	Snort	1.00	0.00	0.00	1.00	1.00
	RNN	1.00	1.00	0.99	1.00	1.00
	Stacked RNN	1.00	1.00	1.00	1.00	1.00
	CNN	1.00	1.00	0.99	0.99	1.00
Recall	Snort	0.0023	0.00	0.00	0.48	0.0004
	RNN	1.00	1.00	1.00	0.99	1.00
	Stacked RNN	1.00	1.00	1.00	1.00	1.00
	CNN	1.00	1.00	1.00	0.99	0.99
f1-score	Snort	0.0046	0.00	0.00	0.65	0.0075
	RNN	1.00	1.00	1.00	0.99	1.00
	Stacked RNN	1.00	1.00	1.00	1.00	1.00
	CNN	1.00	1.00	0.99	0.99	1.00

In term of accuracy, we found that accuracy of Snort, RNN, stacked RNN, CNN are 0.4716, 0.9976, 0.9975, 0.9956, respectively.

When comparing the CPU usage and processing time, the results in Table 6 show that Snort reached the highest 2.69% of CPU usage and spent 14.95 sec for processing time to detect attacks from pcap files.

For the processing time in each deep learning model, we calculate by combining time in pre-processing step with training time and testing time. For our experiment, in pre-processing step consume 4 minutes to convert the packet to the 60,000 flows and extract features from each flow.

As a result, for training time, RNN required 46 minutes 6 seconds, while Stacked RNN and CNN required 53 minutes 37 seconds and 38 minutes, respectively.

For testing time, RNN required 24.10 seconds, while Stacked RNN and CNN required 24.18 and 24.48 seconds, respectively.

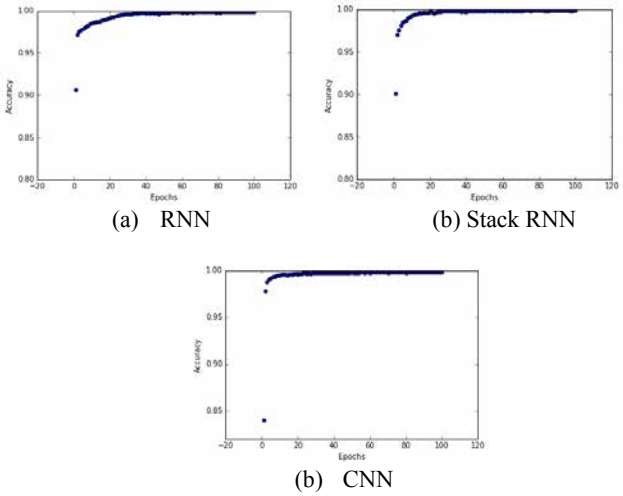
In term of the CPU usage in the training process, Stacked RNN also had the highest CPU usage of 78%, while RNN and CNN had 74% and 67.2% of CPU usage, respectively. The training time and CPU usage of these three deep learning models is high since we do not have the pre-defined rules as in Snort.

TABLE 6. CPU USAGE AND PROCESSING TIME OF EACH ATTACK TYPE

Model	CPU usage		Processing time
Snort	2.69%		14.95 sec
RNN	Training:	74%	46 min 6 sec
	Testing:	0.10%	24.10 sec
Stacked RNN	Training:	78%	53 min 37 sec
	Testing:	0.13%	24.18 sec
CNN	Training:	67.2%	38 min
	Testing:	0.11%	24.48 sec

Moreover, in term of CPU usage in the testing process, RNN required only 0.10%, while Stacked RNN and CNN required only 0.13% and 0.11%, respectively.

Figure 3 (a) – (c) illustrates the accuracy per epochs of RNN, Stacked RNN, and CNN models. By observing the accuracy score in each epoch, we found that CNN started with 0.84 at first epoch then reached 0.98 – 0.99, while stacked RNN and RNN model is usually in the range of 0.90 - 0.99.



(a) RNN (b) Stack RNN

(b) CNN

Figure 3. Accuracy per Epochs of (a) RNN, (b) Stacked RNN, and (c) CNN

VI. CONCLUSIONS

In this paper, we proposed a method to detect network attacks by using deep learning technique instead of using pre-defined rules and signatures of IDS. Through our experiments with five types of attacks, we proved that three supervised-learning models, which are RNN, Stacked RNN, and CNN can be used to classify attacks without information from packet payload. We compare the detection performance of these three models with Snort.

The evaluation results show that all models except Snort provide 1.00 for precision, recall, and f1-score for the port scan. However, for Network scan via ICMP, all three model provide 1.00 for precision, recall, and f1-score but Snort could not detect at all due to the lack of data payload. For DoS

attacks, both Stacked RNN and RNN provide 1.00 for all value of precision, recall, and f1-score, but Snort provides the lowest score for recall and f1-score. Moreover, Stacked RNN gives the highest for all value of precision, recall, and f1-score for network scan via UDP and network scan via TCP. In term of accuracy, RNN yields the best accuracy score of 0.9976.

Snort has the fastest processing time for detecting attacks than all three deep learning models due to the pcap files we used for testing are lack of payload, then Snort analysed only the packet header. However, Snort used the highest CPU usage with 2.69% and provided the lowest accuracy of 0.4716.

In this paper, through our experiments, we confirm that all three deep learning models can detect popular network attacks even we did not access the user payload. Our proposed method was validated in offline manner, however in the real environment we need to do in real time. Therefore, for the future work, we plan to compare and analysis relative approaches and improve our model to detect attacks in real-time manner.

ACKNOWLEDGEMENT

This research project was supported by Faculty of Information and Communication Technology, Mahidol University.

REFERENCES

[1] A. Nayyar, "The Best Open Source Network Intrusion Detection Tools," Open Source For You, [Online], July 12 2018. Available: https://opensourceforu.com/2017/04/best-open-source-network-intrusion-detection-tools/.

[2] K. Thongkanchorn, S. Ngamsuriyaroj and V. Visoottiviseth, "Evaluation studies of three intrusion detection systems under various attacks and rule sets," in 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013), Xi'an, 2013, pp. 1-4.

[3] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to Snort system," in Future Generation Computer Systems, vol. 80, 2018, pp. 157–170.

[4] *TensorFlow*. [Online]. Available: https://www.tensorflow.org/.

[5] "McAfee Labs Threats Report." [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2016.pdf

[6] "McAfee Labs Threats Report: April 2017 | McAfee." [Online]. Available: https://www.mcafee.com/enterprise/en-us/security-awareness/threats-report-mar-2017.html

[7] "McAfee Labs Threats Report: June 2017 | McAfee." [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2017.pdf

[8] "McAfee Labs Threats Report: September 2017 | McAfee." [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-sept-2017.pdf

[9] George, K., "SANS Institute InfoSec Reading Room." [Online], 2015, Available:https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772

[10] F. Siemons. (2018, Jan. 18) *Open Source IDS: Snort or Suricata?*, InfoSec Resources [Online]. Available: https://resources.infosecinstitute.com/open-source-ids-snort-suricata/#gref

[11] Anon. *Snort* [Online]. Available: https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/000/116/original/Snort_rule_infographic.pdf?AWSAccessKeyId=AKIAIXACIED2SPMSC7GA&Expires=1511594325&Signature=KsbSffWa4JPKl68TyInchf79rWg%3D

[12] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," in The Journal of Finance and Data Science, vol. 2, no. 4, 2016, pp. 265–278.

[13] R. Bonnin, *Building Machine Learning Projects with TensorFlow*, Birmingham: Packt Publishing, 2016.

[14] W. Luo, W. Liu and S. Gao, "A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 341-349.

[15] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems [Online]. Available: https://www.tensorflow.org/.

[16] Keras.io. (2018). *Keras Documentation* [Online]. Available at: https://keras.io/

[17] P. P.g and D. E. D, "Design of a Hybrid Intrusion Detection System using Snort and Hadoop," in International Journal of Computer Applications, vol. 73, no. 10, 2013, pp. 5–10.

[18] S. Potluri, N. F. Henry and C. Diedrich, "Evaluation of hybrid deep learning techniques for ensuring security in networked control systems," in 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, 2017, pp. 1-8.

[19] X. Zhang, P. Zhu, J. Tian, and J. Zhang, "An effective semi-supervised model for intrusion detection using feature selection based LapSVM," in 2017 International Conference on Computer, Information and Telecommunication Systems (CITS), 2017, pp. 283–286.

[20] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, 2017, pp. 21954-21961.

[21] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1222- 1228.

[22] Fontugne, R. (2018). *MAWILab - Data set – 2017* [Online]. Available: http://www.fukuda-lab.org/mawilab/v1.1/2017.html.

[23] Anon. (2018, Nov. 15). *Bro Script Index* [Online]. Available: https://www.bro.org/sphinx/script-reference/scripts.html

[24] The CERT Network Situational Awareness Team (CERT NetSA). (2006). *SiLK,* [Online]. Available: https://tools.netsa.cert.org/silk/index.html

[25] Index of /open/snort-2.9.0. [Online]. Available: https://rules.emergingthreats.net/open/snort-2.9.0/.

[26] J. Brownlee. (2018, Apr. 24). *What is the Difference Between a Batch and an Epoch in a Neural Network?, Machine Learning Mastery* [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[27] S. Sharma. (2017, Sep. 6). *Activation Functions: Neural Networks – Towards Data Science* [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[28] Anon. (2015). *Theanets 0.7.3 documentation* [Online]. Available: https://theanets.readthedocs.io/en/stable/api/losses.html

[29] Tensorflow. (2018, Nov. 2). *tensorflow/docs* [Online]. Available: https://www.tensorflow.org/api_guides/python/train

Navaporn Chockwanich is a master's degree student in the Faculty of Information and Communication Technology, Mahidol University. She received her bachelor's degree in computer network at Mahidol University, Thailand, in 2014. She was an internship student at Nara Institute of Science and Technology in Japan in 2012. Her research interests are network security, anomaly detection and data analysis.

Vasaka Visoottiviseth is an assistant professor at Mahidol University, Thailand. She received her Ph.D. degree in computer engineering from Nara Institute of Science and Technology in 2003, M.E. and B.E. degree from Tokyo University of Agriculture and Technology in 1999 and 1997, respectively. Her current research interests are mobile and wireless computing, Internet traffic measurement, and network security.