Real-Time Computer Network Anomaly Detection Using Machine Learning Techniques

Kriangkrai Limthong

***Abstract*—Detecting a variety of anomalies in computer network, especially zero-day attacks, is one of the real challenges for both network operators and researchers. An efficient technique detecting anomalies in real time would enable network operators and administrators to expeditiously prevent serious consequences caused by such anomalies. We propose an alternative technique, which based on a combination of time series and feature spaces, for using machine learning algorithms to automatically detect anomalies in real time. Our experimental results show that the proposed technique can work well for a real network environment, and it is a feasible technique with flexible capabilities to be applied for real-time anomaly detection.**

***Index Terms*—Multivariate normal distribution, nearest neighbor, one-class support vector machine, unsupervised learning.**

I. Introduction

Owing to the explosive growth of Internet traffic, it is quite difficult for network operators to inspect every single packet or flow that passes through their networks. There also has been an exponential increase in sophisticated techniques used by computer attacks to evade existing anomaly detectors [1]. In addition, unusual incidents caused by internal operations, such as outages or misconfigurations, can create abnormal behavior of networks. Anomalies arising from all of these causes adversely affect security, and some of them are responsible for network congestion. Thus, there is a critical need for automatic detection of attacks and unusual incidents in computer networks.

Anomaly detection techniques in the context of computer network can be generally classified into signature-based and statistical-based approaches [2]. Signature-based approaches, however, cannot detect new and previously unidentified attacks, while statistical-based approaches can detect such attacks. Statistical-based approaches are also capable of learning and automatically adapting to specific networks [3]. Meanwhile, machine learning is one of the fields that researchers are currently applying to this domain [4].

Previous studies suggest that machine learning can play a major role in anomaly detection for computer networks [5], [6]. Many machine learning algorithms and techniques, such as *k*-nearest neighbor algorithms [7], neural networks [8], support vector machines [9], and k-means clustering [10], have been applied to detect anomalies. However, most studies are batch processing, which collects a certain amount of data before detecting anomalies, and most of these techniques are therefore not suitable for real-time detection.

To address this problem, we propose a technique on the basis of time series and feature spaces for real-time anomaly detection in computer networks. We also present a series of experiments that we conducted to examine the performance and accuracy of our proposed technique. We acquired real network traffic and a test bed containing various attacks for our experiments. Moreover, we compared the performance of three well-known machine learning algorithms, namely the multivariate normal distribution, *k*-nearest neighbor algorithm, and one-class support vector machine, with the proposed technique.

II. Materials and Methods

A. Our Proposed Technique

The fundamental idea of our proposed technique is depicted in Fig. 1. From one-day network traffic data, we first generate a sequence of data points in successive order at regular time intervals. Second, we extract features from every single time interval and construct a time series of each feature, where *n* is the number of features or the number time series. Next, we create a feature vector for each time interval and map it as a single data point on a corresponded feature space, where one time interval corresponds with only one feature space. Therefore, the number *t* of time intervals or the number of feature spaces depends on duration setting of regular time intervals. Please note that we represent two-dimensional in Fig. 1 rather than high-dimensional feature spaces because it makes visualization more comprehensible. In practice for real environments, we can represent data up to *n*-dimensional features.

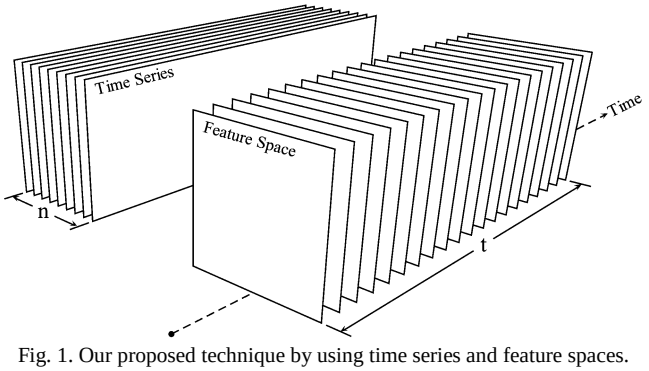

Fig. 1. Our proposed technique by using time series and feature spaces.

Manuscript received September 20, 2012; revised November 17, 2012. This work was supported in part by the Faculty Members Development Scholarship Program of Bangkok University, Thailand.

Kriangkrai Limthong is with the Department of Computer Engineering, School of Engineering, Bangkok University, Pathumtani 12120, Thailand (e-mail: kriangkrai.l@bu.ac.th). He is also now with the Department of Informatics, Graduate University of Advanced Studies (Sokendai), Chiyoda-ku, Tokyo 101-8430, Japan (e-mail: krngkr@nii.ac.jp).

DOI: 10.7763/JACN.2013.V1.1

TABLE I: CHARACTERISTICS OF SELECTED ATTACKS

Source	No. of SrcAddr	No. of DstAddr	No. of SrcPort	No. of DstPort	No. of Packet	Average Packet Size (Byte)	Duration (sec.)	Average Packet/sec.	% Anomaly
Back									
Week 2 Fri	1	1	1,013	1	43,724	1,292.31	651	67.16	0.75
Week 3 Wed	1	1	999	1	43,535	1,297.29	1,064	40.92	1.23
IpSweep									
Week 3 Wed	1	2,816	1	104	5,657	60.26	132	42.86	0.15
Week 6 Thu	5	1,779	2	105	5,279	67.75	4,575	1.15	5.30
Neptune									
Week 5 Thu	2	1	26,547	1,024	205,457	60	3,143	65.37	3.64
Week 6 Thu	2	1	48,932	1,024	460,780	60	6,376	72.27	7.38
Week 7 Fri	2	1	25,749	1,024	205,600	60	3,126	65.77	3.62
PortSweep									
Week 5 Tue	1	1	1	1,024	1,040	60	1,024	1.02	1.19
Week 5 Thu	1	1	1	1,015	1,031	60	1,015	1.02	1.17
Week 6 Thu	2	2	2	1,024	1,608	60	1,029	1.56	1.19
Smurf									
Week 5 Mon	7,428	1	1	1	1,931,272	1,066	1,868	1,033.87	2.16
Week 5 Thu	7,428	1	1	1	1,932,325	1,066	1,916	1,008.52	2.22
Week 6 Thu	7,428	1	1	1	1,498,073	1,066	1,747	857.51	2.02

We firmly believe that our proposed technique has two noteworthy advantages. The first advantage is that we can transform low-level features into high-level features by applying signal processing or time series techniques after the feature extraction process. High-level features produce higher accuracy of anomaly detection than low-level features. The second advantage is that a single time interval represented by one feature space is both computationally feasible and an appropriate key for real-time anomaly detection because we do not need entire test data for detecting anomalies.

However, a disadvantage of our proposed technique is that it needs more computation time, particularly during the training phase. The computation time at the training phase depends on t, the number of time intervals. Fortunately, for real-time systems, we do require a short time computing during the testing phase rather than during the training phase.

B. Data Sets

Prior to our experiments, we divided the data into two sets: a training set and a test set. The entire network data comprise 55 days of normal traffic from a relatively controlled campus network at the Kasetsart University, Thailand. We used 39 days of normal traffic as the training set for the classifiers, and the remaining 16 days as the test set. We selected five types of attacks from the Lincoln Laboratory at the Massachusetts Institute of Technology [11], and then we combined each type of attack with the test set to create a separate test set for each type of attack. The detail of selected attacks are as follows:

1) *Back* attack, a denial of service attack through port 80 of the Apache web server in which a client requests a URL containing many backslashes.
2) *IpSweep* attack, a surveillance sweep involving either a port sweep or ping on multiple IP addresses.
3) *Neptune* attack, a denial of service attack involving a SYN flood at one or more destination ports.
4) *PortSweep* attack, a surveillance sweep through many ports to determine which services are supported on a single host.
5) *Smurf* attack, an amplified attack using an ICMP echo reply flood.

We listed the essential characteristics of selected attacks in Table I. In the first column, we indicate sources and types of anomalies for each instance. In the next five columns, we show primitive characteristics of each anomaly instance: the number of source addresses, destination addresses, source ports, destination ports, and packets. Next, the average packet size and duration of each anomaly instance are shown in the seventh and eighth columns. Lastly, the average number of anomaly packets per second and percentage of each instance in one day are shown in the last two columns, respectively.

TABLE II: FEATURES OF NETWORK TRAFFIC ON AN INTERVAL BASIS

f#	Feature	Description
f_1	Packet	Number of packets
f_2	Byte	Sum of packet size
f_3	Flow	Number of flows
f_4	SrcAddr	Number of source addresses
f_5	DstAddr	Number of destination addresses
f_6	SrcPort	Number of source ports
f_7	DstPort	Number of destination ports
f_8	ΔAddr	|SrcAddr – DstAddr|
f_9	ΔPort	|SrcPort – DstPort|

C. Feature Extraction and Feature Scaling

We chose the nine features as listed in Table II on account of the distinctive characteristics of selected attacks as listed in Table I. We extracted all nine features during packet aggregation for each interval, and then created a single time series for each feature. On the one hand, we directly derived a feature vector from time series for a corresponded feature space. On the other hand, we applied discrete wavelet transform as a filter bank [12] to remove noise from time series before marking a modified feature vector in a corresponded feature space. In the final step, we compared the detection performance of our technique between using raw features and modified features by applying the discrete wavelet transform to the time series data.

For our feature scaling process, we normalized the wide range of different features into a standard range of 0 to 1. We scaled features according to

$$\hat{x}_{i,j} = \frac{x_{i,j}}{\max_j(x_{i,j})}, \forall_{i=1\cdots f} \wedge \forall_{j=1\cdots m,} \quad (1)$$

where $\hat{x}_{i,j}$ is a scaled feature, $\max_j(x_{i,j})$ is the maximum value of the data in the *i*-th feature, *m* is the number of samples in the training data, and *f* is the number of the feature from feature extraction process.

D. Performance Evaluation

We used F-score [13] as a single measure for evaluating the detection performance of our proposed technique. The F-score is widely used to evaluate the quality of binary classifications, especially when the sizes of two classes are substantially skewed. The F-score, which considers both the precision and recall [14] to compute the score, assigns a value ranging between 0 and 1, where 1 represents a perfect detection and 0 represents a worst detection. We measured the precision, recall, and F-score based on entire intervals. The precision, recall, and F-score are derived by Eqs. 2-4 respectively:

$$precision = \frac{TP}{TP + FP}, \quad (2)$$

$$recall = \frac{TP}{TP + FN}, \quad (3)$$

$$F\text{–}score = 2 \times \frac{precision \times recall}{precision + recall}, \quad (4)$$

where *TP* is the number of true positives (the number of anomalous intervals that were correctly detected), *FP* is the number of false positives (the number of normal intervals incorrectly identified as anomalous intervals), and *FN* is the number of false negatives (the number of anomalous intervals that were not detected). *TP*, *FP*, and *FN* were directly derived from a confusion matrix [14].

E. Learning Algorithms

We employed three standard and well-known algorithms of machine learning: namely the multivariate normal distribution, *k*-nearest neighbor, and one-class support vector machine, to work with our proposed model.

1) *Multivariate Normal Distribution (MND):* The MND is a generalization of the Gaussian or normal probability density function (pdf) in high dimensions [15]. In the *f*-dimensional space, the pdf is given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{f/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \quad (5)$$

where $\boldsymbol{\mu} = E[\mathbf{x}]$ is the vector of mean value, and $\boldsymbol{\Sigma}$ is the $f \times f$ covariance matrix defined as

$$\boldsymbol{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T], \quad (6)$$

where $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

To classify test data, we defined an adaptive threshold

$$\varepsilon = \frac{1}{(2\pi)^{f/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\tfrac{1}{2}\rho^2\right), \quad (7)$$

where ρ is a parameter to get the proportion of maximum probability, where smaller values of ρ produce higher probabilities. We varied ρ between 2 and 4 on a linear scale for selection of the best detection performance. We defined the classify function of test data **x** as

$$f(\mathbf{x}) = \begin{cases} \text{anomaly} & \text{if } p(\mathbf{x}) < \varepsilon \\ \text{normal} & \text{otherwise.} \end{cases} \quad (8)$$

2) *k-Nearest Neighbor (KNN):* The KNN is an instance-based learning for classifying data point based on closest learning examples in the *f*-dimensional space [16]. In our experiment, the nearest neighbors of data are defined by the standard Euclidean distance. More precisely, let test instance **x** comprising *f* features be described by the feature vector $(x_1, x_2, \cdots, x_f)$, where x_i denotes the value of the *i*-th feature of data **x**. The Euclidean distance between two instances **x** and **y** is defined by

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{f}(x_i - y_i)^2}. \quad (9)$$

To classify test data, we constantly specified the parameter $k = 3$, and defined the classify function of test data **x** as

$$f(\mathbf{x}) = \begin{cases} \text{anomaly} & \text{if amount of training data nearest} \\ & \text{to } \mathbf{x} \text{ is less than } k \text{ in the distance } D \\ \text{normal} & \text{otherwise.} \end{cases} \quad (10)$$

Thanks to the feature scaling step, we can vary a constant value *D* on a logarithmic scale between 10^{-6} and 10^{0} for selection of the best detection performance.

3) *One-Class Support Vector Machine (OSVM):* The OSVM introduced by B. Schölkopf et al. [17] is a variation of the standard support vector machines (SVM) algorithm. The main ideal is that the OSVM maps unlabeled input data into a high dimensional space via an appropriate kernel function, and then attempts to find hyperplanes that separate input data with maximum margin. According to [18], the decision function

$$h(\mathbf{x}) = \text{sign}((\boldsymbol{\omega} \cdot \Phi(\mathbf{x})) - \rho) \quad (11)$$

will be positive for most examples $\mathbf{x}_i$ contained in the training set or negative for the opposite. Therefore, we defined the classify function of test data **x** as

$$f(\mathbf{x}) = \begin{cases} \text{anomaly} & \text{if } h(\mathbf{x}) = -1 \\ \text{normal} & \text{if } h(\mathbf{x}) = +1. \end{cases} \quad (12)$$

In our experiments, we used the LIBSVM [19] tool with a radial basis function (RBF) as an appropriate kernel. We used the standard parameters of this tool for all experiments with the OSVM algorithm; however, we varied the *nu* and *gamma* parameters on a logarithmic scale between 10^{-3} and 10^{0} for selection of the best detection performance.

III. Preliminary Results

We performed our experiments with two groups of features: raw features and modified features. The raw features were directly extracted from the computer traffic while the modified features were the result of using a discrete wavelet transform to denoise the raw feature.

Fig. 2 demonstrates three different time series for a raw feature and for modified features at first and second levels of denoising, where the x-axis represents time between 8:00 and 24:00, and the y-axis represents the number of packets, one of the features that we used. The top graph in Fig. 2 shows the packet feature (f_1) as a raw feature time series. The middle graph in Fig. 2 shows the packet feature as the first level of feature modification, after a discrete wavelet transform was applied to remove noise from the raw feature. The bottom graph in Fig. 2 shows the packet feature as the second level of feature modification, after we applied a discrete wavelet transform to the first level of feature modification. We performed iterative denoising to arrive at eleven levels of modified features.

To examine the performance of our technique using raw features, we first trained classifiers and detected each type of attack using the individual raw feature from f_1 to f_9 as listed in Table II. We then compared detection performance between the MND, KNN, and OSVM algorithms. Fig. 3 shows the F-score results for the individual raw features using the three different learning algorithms. For the next step, we combined high effective features and conducted experiments to compare the performance between using raw and modified features. According to the results in Fig. 3, we manually selected the effective features for feature combination and used the same combination for both raw and modified features. We then examined the performance for a combination of raw features and combinations of modified features as shown in Table III.

Table III indicates F-score values of MND, KNN, and OSVM algorithms for each type of attack. The features that were used for each type of attack are indicated in the Feature column. The f_{raw} column shows F-score values for raw features and f_{mod} column shows the highest F-score values among the eleven levels of modified features. The bold text is used in Table III to highlight the F-score values for modified features that are higher than the corresponding F-score values for raw features.

IV. Discussion

Our results suggest that the proposed technique produce fine performance for many types of attacks. Even if our technique depends upon selecting the features to detect each particular type of anomaly, the results of experiments on real computer traffic show the proposed technique is capable of detecting anomalies in real time. Our proposed technique does not require the entire test data, and it can detect anomalies that occur during each time interval.

Our results also suggest that in many cases, the performance of feature combination was inferior than the performance of single feature. In addition, the comparison between using raw features and modified features strongly suggests that feature selection has an major effect on performance of our technique. In some cases, we were able to improve performance by using modified features instead of using raw features.

We realize that an inherent limitation of our technique is giving anomaly details. Although the proposed technique can indicate specific time interval during anomalies occur, it does not provide details related to such anomalies. Therefore, we need a second technique to provide more details about the anomalies after they have been identified. However, adding another technique could take more time for computation. There is thus a trade-off between taking more time and providing details about the anomalies.

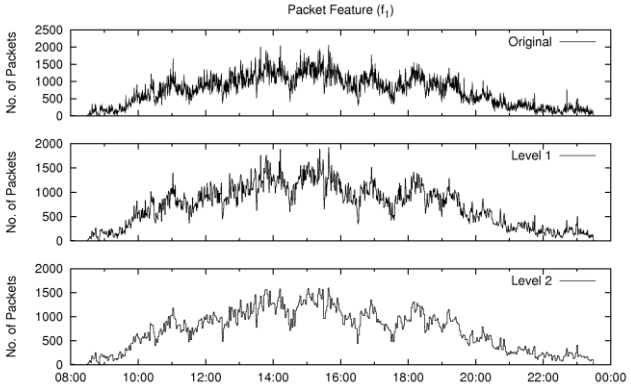

Fig. 2. Original packet feature (top), first level of modified features (middle), and second level of modified feature (bottom) by using wavelet transform.

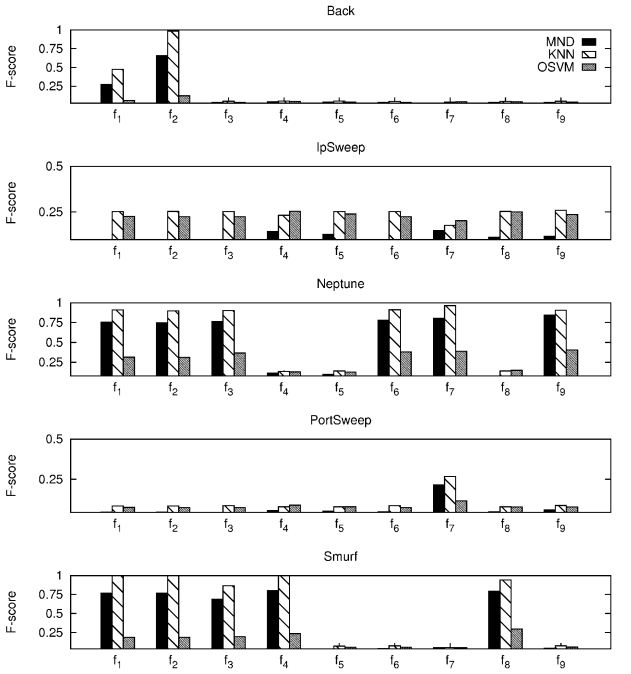

Fig. 3. Performance comparison between individual raw features (f_1-f_9) using the MND, KNN, and OSVM algorithms for each type of attack.

V. Conclusion

The ultimate goal of our research is to develop a highly flexible technique to automatically detect a variety of computer network anomalies in real time. We have proposed a feasible technique that has various flexible capabilities for performing this difficult task. We conducted experiments with real network traffic and compared the detection performance of three well-know machine learning algorithms using nine features. We also tried to improve the quality of the features by using discrete wavelet transform to remove noise from the raw features.

The results show that our proposed technique performs well in task of anomaly detection and has a good possibility for applying in real-time system. Nevertheless, indicating

time consumption of our technique during the training and test phase are need. Another challenge for our future research is to refine the selection of features for detecting particular anomalies.

TABLE III: PERFORMANCE COMPARISON BETWEEN RAW AND MODIFIED FEATURE COMBINATIONS

Attack	Feature	MND		KNN		OSVM	
		f_{raw}	f_{mod}	f_{raw}	f_{mod}	f_{raw}	f_{mod}
Back	f_{1-2}	0.3116	**0.3162**	0.9846	0.9803	0.1498	0.1459
IpSweep	f_{1-9}	0.3172	0.3077	0.2533	**0.2687**	0.2479	**0.3160**
Neptune	f_{1-3}, f_{6-7}, f_9	0.5124	0.4431	0.9534	**0.9683**	0.4197	0.4191
PortSweep	f_7	0.2150	**0.2741**	0.2675	**0.2992**	0.1154	0.1043
Smurf	f_{1-4}, f_8	0.2436	0.2113	1.0000	0.9946	0.2386	0.2310

ACKNOWLEDGMENT

The authors would like to thank all of the anonymous reviewers for their excellent suggestions that have greatly improved the quality of this paper.

REFERENCES

[1] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Computers & Security*, vol. 24, no. 1, pp. 31–43, 2005.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[3] P. Garca-Teodoro, J. Daz-Verdejo, G. Maci-Fernndez, and E. Vzquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 12, pp. 18–28, 2009.

[4] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings of the 15th Annual Computer Security Applications Conference*, ser. ACSAC '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 371–377.

[5] S. Jiang, X. Song, H. Wang, J.-J. Han, and Q.-H. Li, "A clustering-based method for unsupervised intrusion detections," *Pattern Recogn. Lett.*, vol. 27, pp. 802–810, May 2006.

[6] P. Laskov, P. Dssel, C. Schfer, and K. Rieck, "Learning intrusion detection: Supervised or unsupervised?" in *Image Analysis and Processing ICIAP 2005*, ser. Lecture Notes in Computer Science, F. Roli and S. Vitulano, Eds., vol. 3617. Springer Berlin / Heidelberg, 2005, pp. 50–57.

[7] Y. Liao and V. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.

[8] S.-J. Han and S.-B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 3, pp. 559–570, June 2005.

[9] R. Zhang, S. Zhang, S. Muthuraman, and J. Jiang, "One class support vector machine for anomaly detection in the communication network performance data," in *Proc. the 5th conference on Applied electromagnetics, wireless and optical communications.* Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 31–37.

[10] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *Proceedings of Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 4. GI/ITG-Workshop MMBnet 2007*, Hamburg, Germany, Sep. 2007.

[11] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," in *Proc. DARPA Information Survivability Conference and Exposition*, vol. 2, 2000, pp. 12–26.

[12] M. Vetterli and C. Herley, "Wavelets and filter banks: theory and design," *IEEE Transactions on Signal Processing*, vol. 40, no. 9, pp. 2207–2232, September 1992.

[13] C. J. V. Rijsbergen, *Information Retrieval*, Newton, MA, USA: Butterworth-Heinemann, 1979.

[14] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proc. the 23rd international conference on Machine learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240.

[15] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*, 4th ed. Academic Press, 2008.

[16] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[17] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[18] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 181 –201, mar 2001.

[19] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Kriangkrai Limthong received a B.Eng (2nd Honors) degree in Computer Engineering from Sripatum University; and a M.Eng degree in Computer Engineering from Kasetsart University, Thailand. He worked as a Systems Engineer at Advanced Info Service PLC. and Thailand Post Co., Ltd. for several years. He is currently pursuing the Ph.D. degree in the Department of Informatics, Graduate University of Advanced Studies (Sokendai), Japan. He has also been a lecturer in the Department of Computer Engineering, School of Engineering, Bangkok University, Thailand, since 2009. His research interests are network traffic measurement, computer security, signal processing techniques and machine learning methods.