Weighting Technique on Multi-timeline for Machine Learning-based Anomaly Detection System

Kriangkrai Limthong*, Kensuke Fukuda†, Yusheng Ji†, and Shigeki Yamada†
* Computer Engineering Department, Bangkok University, Prathumthani 12120 Thailand.
†National Institute of Informatics, Tokyo 101-8430 Japan.
kriangkrai.l@bu.ac.th, {kensuke,kei,shigeki}@nii.ac.jp

***Abstract*—Anomaly detection is one of the crucial issues of network security. Many techniques have been developed for certain application domains, and recent studies show that machine learning technique contains several advantages to detect anomalies in network traffic. One of the issues applying this technique to real network is to understand how the learning algorithm contains more bias on new traffic than old traffic. In this paper, we investigate the dependency of the time period for learning on the performance of anomaly detection in Internet traffic. For this, we introduce a weighting technique that controls influence of recent and past traffic data in an anomaly detection system. Experimental results show that the weighting technique improves detection performance between 2.7-112% for several learning algorithms, such as multivariate normal distribution, *k*-nearest neighbor, and one-class support vector machine.**

Keywords*—*weighting technique, machine learning, multiple timeline, anomaly detection.

I. Introduction

Because of mobile applications, access and backbone Internet traffic has exponentially grown every year. In addition, available software and tools for novice attackers is easily available [1], so that they can simply use these software to attack and intrude into target networks. These attack tools have been developed very well to imitate normal packets, flows and easily evade existing detection systems. Some attackers have been applied advanced techniques, so conventional detection methods hardly detect the traffic from such malicious software. For these reasons, it makes daily operation difficult for network administrators to monitor unusual incidents or anomalies passing through their own systems.

Many anomalies in current computer networks are more complicated than prior ones. We generally categorize anomalies on computer systems into two groups. The first group is anomalies caused by threats or human intention, such as attacks, viruses, worms, scanning, and spamming. The second group is anomalies caused by accidents, including outages, misconfigurations, and flash crowds. The desirable detection system should discover both types of anomalies with high accuracy and produce low error rate.

In past years, researchers have studied various detection techniques to compete against new malicious software and new types of anomalies in computer networks. A study by Denning [2] classified detection systems into host-based and network-based detection systems. Host-based detection systems are installed locally on many different types of machines, namely servers, workstations, notebook computer, or even mobile devices. Host-based detection systems analyze and pass traffic through a particular host if there are not potentially malicious packets. An example of host-based detection systems is PortSentry [3]. Network-based detection systems are strategically positioned in a network to monitor all traffic and detect any unusual traffic on the hosts of that network. Network-based detection systems need to be a very fast to analyze traffic, because it needs to monitor all packets or flows passing though the network. Examples of network-based detection systems are RealSecure, SecureNet [4], and Snort [5].

Recently, studies on machine learning [6] show promising results in anomaly detection. They have been mainly classified into two techniques: classification and clustering techniques [7]. Classification is a learning technique that requires labeled instances as training data to generate a function for classifying a new instance. Detection systems applying supervised techniques need to label packets or flows which are normal or anomalous. Several learning algorithms based on classification technique have been studied, such as *k*-nearest neighbors [8], support vector machines [9]. Clustering is a learning technique that tries to find hidden structure in data. We could detect anomalies in network traffic on the basis of the assumption that major groups are normal traffic and minor groups are anomalies. Many studies have employed clustering techniques, such as [10] and [11].

As an extension of network anomaly detection with machine learning, we proposed a multi-timeline detection system [12] that use multiple time series of network traffic corresponding to previous days behavior, as input to machine learning algorithm. This technique helps the system to detect anomalies more accurate than those of sigle timeline, as shown in the empirical results [12]. Nevertheless, this multi-timeline technique treats all traffic or timelines equally, but some network environments need a weight on particular timelines, for example, more on recent timelines. Thus, a use of weight among different time series would improve the detection performance.

In this paper, we propose a weighting technique to influence the multi-timeline detection system over recent timelines. We conduct experiments on real network traffic to compare detection performance between weighting and non-weighting technique for several types of attacks with three learning algorithms: the multivariate normal distribution, *k*-nearest neighbor, and one-class support vector machine.

The following sections provide explanation of our detection system and show experimental results on a campus network

978-1-4673-9354-6/15/$31.00 ©2015 IEEE

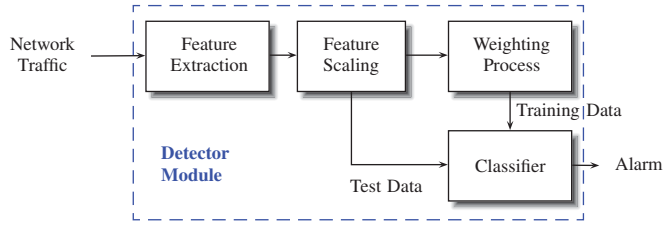

Fig. 1. Block diagram of detector module with weighting process.

traffic. Experimental results also show that the proposed detection system has the capability to detect several types of anomalies without prior knowledge of such attacks.

II. Proposed System Design

This section provides explanation of our weighting technique for applying to the multiple timeline-based anomaly detection system. The original detector module [12] consists of three primary functions, namely feature extraction, feature scaling, and classifier. The weighting process is a newly added module between feature scaling and classifier during learning process as shown in Figure 1.

1) Feature Extraction: The main function of feature extraction is selecting important features of network traffic. Extracted features should directly or indirectly associate with expected anomalies. In our study, we can extract feature of network traffic by using information on packet header and aggregate packets or flows on interval basis. Table I shows traffic features obtained by aggregating information of packet header.

TABLE I. Fundamental features of network traffic by aggregating information of packet header.

$f\#$	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	$\lvert$SrcAddr $-$ DstAddr$\rvert$
f_9	ΔPort	$\lvert$SrcPort $-$ DstPort$\rvert$

2) Feature Scaling: Feature scaling is a process to standardize a wide range of feature values to the same range. This process may be unnecessary for other techniques for anomaly detection in network traffic; however, feature scaling or feature normalization is indispensable for the multi-timeline detection system that relies mainly upon machine learning algorithms. Most learning algorithms will not work properly without feature scaling.

Even though many scaling functions have been proposed [13], the common goal of scaling functions is to independently normalize each feature component to the [0,1] or [-1,1] range. We used following two scaling functions that would be appropriate for fast computation. We normalize feature values by using the max value as

$$x' = \frac{x}{\max(x)}, \tag{1}$$

or by using the range of feature as

$$x' = \frac{x - \mu}{\max(x) - \min(x)}, \tag{2}$$

where x' is the normalized value derived from x an original value, μ is the average value, $\max(x)$ and $\min(x)$ are maximum and minimum values of x respectively. These two scaling functions have the same time complexity as $O(n)$ where n is the number of feature values, so these would take a very short time of computation for real-time systems.

3) Weighting Process: Weighting is our proposed process for the multi-timeline detection module during the learning process. This module guides the learning algorithm to generate a decision function which relies on particular timelines. Consequently, network operators could weight on one or more timelines to bias the decision function of detection system. For example, recent timelines should have a strong influence on the classifier than other older timelines. There are various weighting techniques for different purposes that could be plug into this module, such as by performing a sum, integral, average or even calculus [14]. In our experiments, we adapted a gradual weighting function to recent timelines and we will describe more details of our weighting function later.

4) Classifier: The major role of classifier is distinguishing between normal and anomaly in network traffic. The classifier in detector module is created by an algorithm using training data during learning process. As a result, effectiveness of classifier depends highly upon three factors: learning algorithm, representation of input data, and training set. In detecting process, the classifier receives a test data as an input, then produces a label of test data as an output to alarm system. One of our purposes to multi-timeline learning is enhancing a capability of detector module. In addition, the detector module could be applied in various schemes to detect different types of anomalies in network systems.

In classifier block, conventional detection systems represent input data with sigle timeline as shown in Figure 2. For this representation, we aggregate traffic into five intervals; $\mathbf{x}_{t-4}$, $\mathbf{x}_{t-3}$, $\mathbf{x}_{t-2}$, $\mathbf{x}_{t-1}$, and $\mathbf{x}_t$ occur on a single timeline or on the same day. We intend to classify a test interval whether it is an anomaly instance or not by using the decision function, the test instance is $\mathbf{x}_t$ for example. The decision function is a mathametical function which takes a dataset as input and gives a decision (a test interval is normal or anomalous) as output. The decision function $f(\mathbf{x}_t)$ can use information from the rest of instances, from $\mathbf{x}_{t-4}$ to $\mathbf{x}_{t-1}$ (it may include the test instance $\mathbf{x}_t$ as well) as input. As shown in the figure, we depict the bold line to represent a test connection between the test instance $\mathbf{x}_t$ and the decision function. We also draw the dash lines to represent learning connection between the rest of instances and the decision function. In this figure, we assume that the decision function does not have learning connection with the test instance $\mathbf{x}_t$.

The multi-timeline representation as shown in Figure 3 is different from the single timeline. Suppose we intend to classify the test instance $\mathbf{x}_t^p$ on the present timeline p and have other two timelines, $p-2$ and $p-1$ with the same length. We also depict intervals on timeline $p-2$ with $\mathbf{x}_{t-2}^{p-2}$, $\mathbf{x}_{t-1}^{p-2}$,and $\mathbf{x}_t^{p-2}$, and so forth on others timelines. The algorithm uses

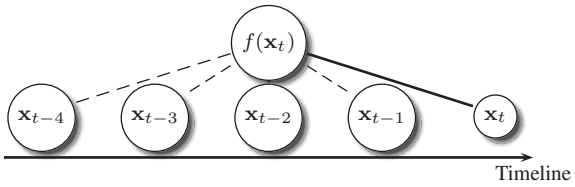

Fig. 2. Single timeline representation of network traffic.

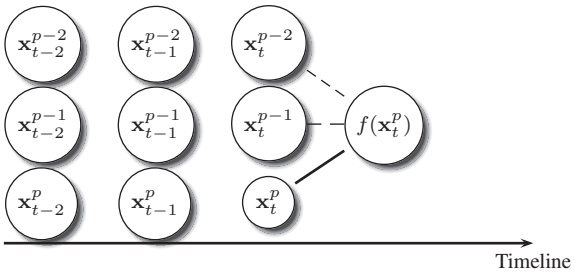

Fig. 3. Multi-timeline representation of network traffic.

multiple timeline from the past as input at the same interval of test instance $\mathbf{x}_t^p$. The learning algorithm also generates the decision function $f(\mathbf{x}_t^p)$ by learning from the same interval as the test instance. In this figure, for the test instance $\mathbf{x}_t^p$, we show a detecting connection has been represented by the bold line, and learning connection by dash lines. The algorithm generates the decision function for $\mathbf{x}_t^p$ by learning from prior intervals from timelines $p-2$ and $p-1$, which are represented by $\mathbf{x}_t^{p-2}$ and $\mathbf{x}_t^{p-1}$, respectively.

The previous study [12] shows that the multi-timeline system outperforms single timeline system in most cases. However, this technique treats all input timelines equally, we add the weighting technique in the multi timelines. To validate the effectiveness of the weight, we conduct experiments on multi-timeline system to compare detection performance between weighting and non-weighting system.

III. Materials and Methods

In this section, we describe data sets, all setting variables for detector module, learning algorithms employed in our experiments, and the evaluation metric.

A. Data Sets

The entire network data comprise 55 days of normal traffic from a relatively controlled campus network.We divided the data into two sets; training and testing data, 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 [15], and then we combined each type of attack into the test set to create separate test set for each type of attack.

The description 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 also measured the volume of normal traffic, which is the aggregated traffic volume of all packets from/to the computer center, and anomaly traffic in both datasets. The minimum, maximum, and average volume of normal traffic are approximately 496 bit/sec., 394 kbit/sec., and 13 kbit/sec. respectively. The average volume of Back, Ipsweep, Neptune, PortSweep, and Smurf are approximately 560 kbit/sec., 11 kbit/sec., 32 kbit/sec., 576 kbit/sec., and 8 Mbit/sec.

B. Feature Extraction and Feature Scaling

We focused on nine interval-based features of network traffic listed in Table I. The interval-based features are characteristics of network traffic that occur in a particular time interval, the number of packets, the number of IP addresses, and the number of ports occur, for example. In Table I, the first column (f#) denotes the abbreviation of each feature; the second column lists the feature name; and the last column shows the description of each feature.

The main function of the feature extraction process is extracting interval-based features from network traffic. We divided one-day network traffic data into each individual time interval. Every time interval had an index number to distinguish it. The number of time intervals or index numbers in a day depended on the interval value selected. For example, if we set the interval value at 1 second, the number of time intervals in a day was 86,400, so the index numbers were from 0 to 86,399. If we changed the interval value to 60 seconds, the number of time intervals in a day was 1,440. In our experiments, we set the interval value at 10 seconds.

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

$$x'_{i,j} = \frac{x_{i,j}}{\max_j(x_{i,j})}, \forall_{i=1\ldots f} \wedge \forall_{j=1\ldots m}, \tag{3}$$

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

C. Weighting Technique

This process will guide the learning algorithm to generate a decision function which relies on particular timelines. Consequently, network operators could weight on one or more timelines to bias the decision function of detection system. Figure 4 shows the weighting process in our detector module during the learning process.

In our implementation, we conduct experiments on the detector module both with and without the weighting process. There are two parameters for weighting process, namely weight length and weight value; the weight length is the number of weighted training days and weight value is the

maximum number of replication. We also made an assumption that recent data contains more useful information than older ones. Figure 4 shows an example of weighting process; m is the number of training days, and setting weight length to 6 and weight value to 3. As a result, we add a weight from day m to $m-5$ where weight 3 added to m and $m-1$, weight 2 added to $m-2$ and $m-3$, weight 1 added to $m-4$ and $m-5$ sequentially. For weight value, we replicate training samples as the number of weight value. For example, we replicate samples 3 times for weight value 3 and so on. We call this technique as linear gradual weighting, used in a study by [16]. We could modify by reducing the weight value exponentially called exponential gradual weighting. In our experiments, however, we apply only linear gradual weighting technique because it is faster than exponential weighting.

D. Learning Algorithms

We employed three well-known algorithms of machine learning [17]: namely the multivariate normal distribution, *k*-nearest neighbor, and one-class support vector machine, to work with our proposed system.

1) Multivariate Normal Distribution (MND): The MND is a generalization of the Gaussian or normal probability density function (pdf) in high dimensions [18]. In the f-dimensional space, the pdf is given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{f/2}\left|\mathbf{\Sigma}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{4}$$

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

$$\mathbf{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T], \tag{5}$$

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

To classify test data, we defined an adaptive threshold

$$\varepsilon = \frac{1}{(2\pi)^{f/2}\left|\mathbf{\Sigma}\right|^{1/2}} \exp\left(-\frac{1}{2}\rho^2\right), \tag{6}$$

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 define the classify function of test data $\mathbf{x}$ as

$$f(\mathbf{x}) = \begin{cases} \text{anomaly} & \text{if } p(\mathbf{x}) < \varepsilon, \\ \text{normal} & \text{otherwise.} \end{cases} \tag{7}$$

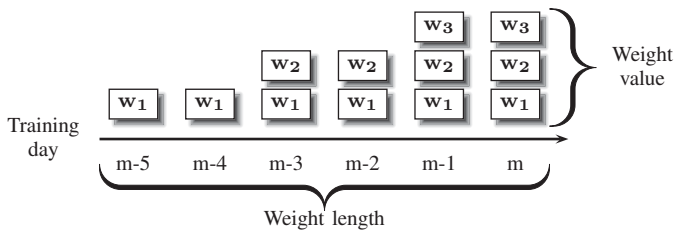

Fig. 4. An example of weighting process with weight length = 6 and weight value = 3.

2) k-Nearest Neighbor (KNN): KNN is an instance-based learning for classifying data point based on closest learning examples in the f-dimensional space [19]. In our experiment, the nearest neighbors of data are defined by the standard Euclidean distance. More precisely, let test instance $\mathbf{x}$ comprising f features be described by the feature vector $(x_1, x_2, ..., x_f)$, where x_i denotes the value of the i-th feature of data $\mathbf{x}$. The Euclidean distance between two instances $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{f}(x_i - y_i)^2}. \tag{8}$$

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

$$f(\mathbf{x}) = \begin{cases} \text{anomaly} & \text{if amount of training data nearest} \\ & \text{to } \mathbf{x} \text{ is less than } k \text{ in the distance } D, \\ \text{normal} & \text{otherwise.} \end{cases} \tag{9}$$

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: The support vector machin maps training data into a high-dimensional feature space. Therefore, we construct a separated hyperplane by maximizing the margin or distance from the hyperplane to the nearest training data points.

In the input space, a binary support vector machine decision function is presented by the following formula:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i . k(x, x_i) + b\right). \tag{10}$$

Here x is the feature vector; α and y are the weights of the support vectors; having y as a positive or negative class mark (+1 or -1) and b is the bias; function $k()$ is the kernel function. Training vectors for which $\alpha_i \neq 0$ are called support vectors.

In our study, we use the Libsvm [20] tools with a radial basis function (RBF) kernel of the form

$$k(x, x') = \exp(-\gamma \left\| x - x' \right\|). \tag{11}$$

Each support vector thus becomes the center of a RBF, and γ determines the area of influence that the support vector has over the data space. We varied the γ value between 10^{-5} and 10^{4} to observe a change for the best accuracy.

To classify testing data, we used the Svm-Predict function from the Libsvm to determine an unknown vector sample x, which belongs to the positive or negative class. It returns +1 or -1 as the result of classification and provides to y the result of the sum from the SVM decision formula Eq. 10. If the result at a particular time interval is -1, we classify that time interval as an anomalous interval. While, if the result is +1, we classify that time interval as a normal interval.

E. Performance Metric

We used F-score [21] 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,

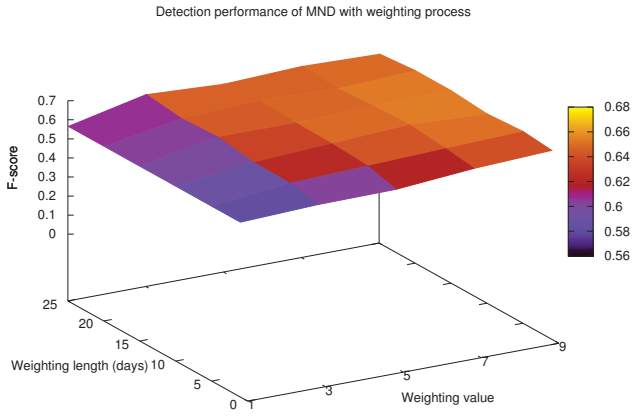

Fig. 5. Detection performance of multi-timeline module with weighting process by using MND.

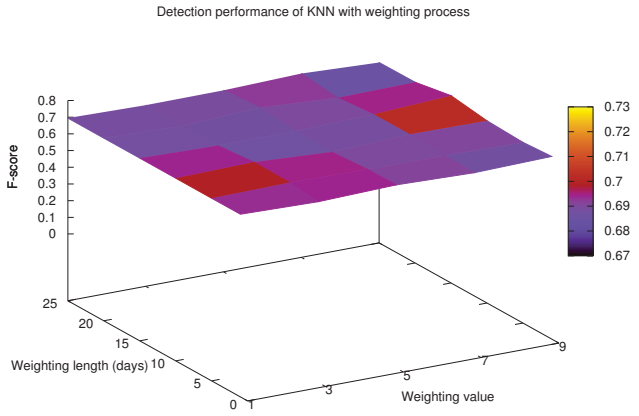

Fig. 6. Detection performance of multi-timeline module with weighting process by using KNN.

especially when the sizes of two classes are substantially skewed. The F-score, which considers both the precision and recall [22] to compute the score, assigns a value ranging between 0 and 1, where 1 represents the perfect detection and 0 represents the worst detection. We measured the precision, recall, and F-score based on entire intervals. The precision, recall, and F-score are derived by Eqs. 12-14 respectively:

$$precision = \frac{TP}{TP + FP}, \tag{12}$$

$$recall = \frac{TP}{TP + FN}, \tag{13}$$

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall}, \tag{14}$$

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

Fig. 7. Detection performance of multi-timeline module with weighting process by using OSVM.

from a confusion matrix [22]. The acceptable F-score in our experiments is 0.5.

IV. Results

We conduct the experiment to explore detection performance of both the multi-timeline detector module with and without weighting process. Network operators intend to give some recent traffic more weight on recent traffic behavior. We use a weighting process on the multi-timeline detector module and comparing detection performance to those without the weighting process.

We perform experiments with a linear gradual weighting technique. First, we set the weight length and weight value as 1 for weighting process. Second, we select the best feature from the first experiment for each type of anomaly by using first learning algorithm, MND. After that we measure F-score for every types of anomalies and compute the average performance of multi-timeline detector module for MND. Third, we alter the weight length from 1 day to 5, 10, 15, 20, and 25 days, then average detection performance for all weight length. Next, we change the weight value from 1 to 3, 5, 7, and 9, then follow the procedure from the first to third step and compute the average detection performance for all value. Finally, we switch the learning algorithm from MND to KNN and OSVM respectively, and plot all computed average values on three-dimensional graphs to compare trends in detection performance between these learning algorithms.

Performance results from this experiment on weighted timeline are shown in Figures 5-7 for MND, KNN, and OSVM, respectively. The x-axis represents the weight values from 1 to 9, and the y-axis indicates the weight length during the learning process. The z-axis shows the F-score that contains a value between 0 and 1, where 0 represents the worst and 1 represents the best detection performance.

Our experimental results indicates the advantage of weight process in most cases. Results of MND (Figure 5) show that the multi-timeline detector module with the weighting process produces 3.2-16.6% improvement than the module without the weighted process. The case of KNN (Figure 6) indicates

that adding the weighting process randomly produces a small effect between 2.7-6.4% improvement. Employing OSVM in Figure 7, however, the trend in detection performance is quite different from those by using MND and KNN. The detection performance with OSVM is improved linearly for increase of the weight values and weight lengths. The improvement of performances by using OSVM are between 3.5-112%. From these three figures, OSVM is a suitable algorithm for the multi-timeline detector module with the linear gradual weighting technique.

V. Discussion and Conclusion

Empirical results from our experiment reveal that the detection performance of multi-timeline system with weighting relies on two main factors. One factor is dependency of two parameters in the weighted process. The weighting technique in this experiment is timeline replication, so both the number of training data and the number of replication as a weighting value mainly affect the F-score. Another factor is the learning algorithm employed to the detection module. Our results indicate that the weighting technique strengthens the role of OSVM. The weighting process with MND has slightly improved for increase of the number of training data and weighting value. The results from KNN show random and small change when we added the weighting technique to the multi-timeline detection module.

In summary, we proposed a weighting technique over the multi-timeline detection system so that we can apply any machine learning algorithms or use any traffic feature to detect network anomalies. We conducted two experiments to examine capabilities of the multi-timeline, detection performance over different volumes of background traffic and weighting process. Experimental results strongly confirm that the multi-timeline detector module with weighting process outperform those without weighting process, especially for OSVM.

For our future work, we intend to apply the multi-timeline detection system to a real network environment. Although, our experimental results show that the multi-timeline detection system with some learning algorithms detected several types of anomalies with promising performance, there are many factors in network traffic that might adversely affect detection performance of the system. Moreover, to fulfill an essential requirement of anomaly detection in real time, we intend to develop an automatic inspector who provide full details of anomalies after it have been detected by the multi-timeline detection system.

Acknowledgment

We gratefully acknowledge the funding from the Faculty Members Development Scholarship Program of Bangkok University, Thailand. 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] H. F. Lipson, "Tracking and tracing cyber-attacks: Technical challenges and global policy issues," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Special Report CMU/SEI-2002-SR-009, November 2002.

[2] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. 13, no. 2, pp. 222–232, Feb. 1987.

[3] B. Toxen, *Real World Linux Security*, 2nd ed. Prentice Hall Professional Technical Reference, 2002.

[4] P. Spirakis, S. Katsikas, D. Gritzalis, F. Allegre, J. Darzentas, C. Gigante, D. Karagiannis, P. Kess, H. Putkonen, and T. Spyrou, "Securenet: A network oriented intrusion prevention and detection intelligent system," in *Proceedings of the 10th International Conference on Information Security, IFIP SEC94*, The Netherlands, May 1994.

[5] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238.

[6] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[8] S. Manocha and M. A. Girolami, "An empirical analysis of the probabilistic k-nearest neighbour classifier," *Pattern Recogn. Lett.*, vol. 28, no. 13, pp. 1818–1824, Oct. 2007.

[9] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of svm and ann for intrusion detection," *Comput. Oper. Res.*, vol. 32, no. 10, pp. 2617–2634, Oct. 2005.

[10] 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, no. 7, pp. 802–810, May 2006.

[11] A. Kind, M. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *Network and Service Management, IEEE Transactions on*, vol. 6, no. 2, pp. 110 –121, june 2009.

[12] K. Limthong, K. Fukuda, Y. Ji, and S. Yamada, "Unsupervised learning model for real-time anomaly detection in computer networks," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 8, pp. 2084–2094, 2014.

[13] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563 – 582, 2001, image/Video Indexing and Retrieval.

[14] J. Grossman, M. Grossman, and R. Katz, *The first systems of weighted differential and integral calculus*. Archimedes Foundation, 1980.

[15] 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 *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, vol. 2, 2000, pp. 12 –26 vol.2.

[16] S. Amasaki and C. Lokan, "The effects of gradual weighting on duration-based moving windows for software effort estimation," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Mnnist, J. Mnch, and M. Raatikainen, Eds. Springer International Publishing, 2014, vol. 8892, pp. 63–77.

[17] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Dec. 2007.

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

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

[20] 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.

[21] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

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