Performance of Interval-Based Features in Anomaly Detection by Using Machine Learning Approach

Kriangkrai Limthong

***Abstract*—Detecting various anomalies or unusual incidents in computer network traffic is one of the great challenges for both researchers and network administrators. If they had an efficient method that could detect network traffic anomalies quickly and accurately, they would be able to prevent security problems or network congestion caused by such anomalies. Therefore, we conducted a series of experiments to examine which and how interval-based network traffic features affect anomaly detection by using three famous machine learning algorithms: the naïve Bayes, *k*-nearest neighbor, and support vector machine. Our findings would help researchers and network administrators to select effective interval-based features for each particular type of anomaly, and to choose a proper machine learning algorithm for their own network system.**

***Index Terms*—Network traffic, anomaly detection, naïve Bayes, nearest neighbor, support vector machine.**

I. Introduction

One of the crucial responsibilities of administrators is discovering various anomalies and unusual incidents in computer network system. Forms or causes of anomalies can vary considerably, which produce a variety of network problems such as network congestion or even security problems. Examples of network anomalies and unusual incidents are denial of service attacks (DoS), viruses or worms spreading, outages, misconfigurations, and flash crowds. If network administrators had an automatic mechanism that expeditiously detected unknown anomalies or unusual incidents, they would avoid serious consequences caused by such anomalies. Thus, an automatic mechanism detecting unknown anomalies in computer network traffic would be attractive.

According to several studies [1]-[3], we can categorize detection methods into two major groups: signature-based methods and statistical-based methods.

The signature-based methods, such as Snort [4], Suricata, or Bro [5], monitor and compare packets with predetermined attack patterns known as signatures. It is a simple and efficient method to examine network traffic. Although the false positive rate of this technique can also be low, comparing network packets or flows with a large set of signatures is a time consuming task and has limited predictive capabilities. In addition, the signature-based methods cannot detect novel anomalies that are not defined in signatures. It means that administrators have to update the system signatures frequently.

The statistical-based methods, however, can learn behavior of network traffic and possibly detect novel anomalies and unusual incidents. Many researchers have studied on particular techniques, such as the statistical profiling using histograms [6], parametric statistical modeling [7], non-parametric statistical modeling [8], rule-based system [9], clustering-based technique [10], and spectral technique [11]. All these techniques are straightforward, but selecting appropriate parameters and threshold, especially when behavior of network traffic changes, is quite difficult.

The machine learning technique is one of the methods which has high capabilities to automatically recognize complex patterns, and make intelligent decisions on the basis of data [12]. There are two fundamental groups of machine learning algorithms: unsupervised algorithms and supervised algorithms [13].

The unsupervised algorithms are a machine learning technique thats take a set of unlabeled data as input and attempt to cluster data. We could detect anomalies on the basis of the assumption that major groups are normal traffic and minor groups are anomalous traffic [14]. Unfortunately, many cases are not true according to this assumption, such as distributed denial of service attacks (DDoS), viruses or worms spreading, and flash crowds. From these examples, the amount of anomalous traffic is normally larger than those of normal traffic for a certain period. In other cases, outages and misconfigurations for example, although no anomalous packet occurs, an unexpected reduction of network traffic indicates an unusual incident as well.

In contrast to unsupervised algorithms, the supervised algorithms can cover and detect a wide range of network anomalies [15]. The basic assumption of anomaly detection using supervised algorithms is that anomalous traffic is statistically different from normal traffic. Many studies have applied several algorithms based upon this assumption, such as the Bayesian network [16], *k*-nearest neighbor [17], and support vector machine algorithm [18]. Nevertheless, the performance of these algorithms has not been compared.

Many previous studies using machine learning method utilized packet-based or connection-based features, which have a scalability problem when the number of packets or flows increases. However, the interval-based features can possibly solve this problem. For example, suppose we have network traffic including 10 packets for 10 seconds, assume that the processing time for 1 packet is 1 unit then the processing time of packet-based features for this case is 10 units. When the number of packets increases to 1,000 packets for the same 10 seconds, the processing time will

Manuscript received June 15, 2013; revised December 25, 2013. This work was supported by the Faculty Members Development Scholarship Program of Bangkok University, Thailand.

Kriangkrai Limthong is with the Department of Informatics, Graduate University of Advanced Studies (Sokendai), Japan (e-mail: kriangkrai.l@bu.ac.th).

also rise to 1,000 units. However, assume that the processing time for the interval-based features is 1 unit per 1 second, the processing time of interval-based feature in both cases are equally 10 units.

Another problem of the packet-based or connection-based features is that they cannot detect some incidents such as outages or misconfigurations. Although using packet-based features can distinguish between normal and anomalous packets, it hardly detects an unexpected incident that does not contain anomalous packet. While the interval-based features have been shown that it could discover unusual incidents which do not have anomalous packets [19]. The question remains what interval-based features are suitable for each particular type of anomaly. Therefore, in this study, we examined several interval-based features to find an answer for this question.

In our study, we examined which and how interval-based network traffic features affect the performance of anomaly detection by using three machine learning algorithms. The key contributions of this work comprise the following three comparisons:

1) Comparison of nine interval-based features relating to each particular type of anomaly.
2) Comparison of detection performance for five major types of anomalies selected from a testbed dataset.
3) Comparison of three well-known machine learning algorithms, namely naïve Bayes, *k*-nearest neighbor, and support vector machine.

We explain materials and methods of our experiments in Section II. Next, we show results of each experiment in Section III. Finally, we discuss the results and draw conclusions in Section IV.

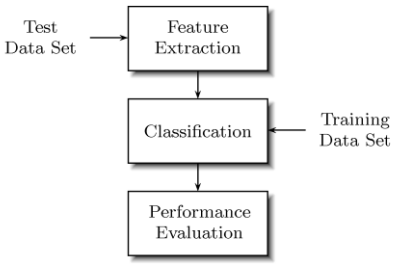

Fig. 1. Procedure for our experiments.

TABLE I: INTERVAL-BASED FEATURES

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|

II. MATERIALS AND METHODS

The diagram in Fig. 1 illustrates the procedure for our experiments. First, the feature extraction step will be explained in subsection II-A. Second, we describe all three learning algorithms applied to the classification step in subsections II-B to II-D. Next, we define metrics used for the performance evaluation step in subsection II-E. Finally, we explain how to collect and prepare training data and test data in subsection II-F.

A. Feature Extraction

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, such as the number of packets, the number of IP addresses, and the number of ports . In Table I, the first column (f#) denotes the abbreviation of each feature, the second column lists the feature name, and the last column describes detail of each feature. Please note that we conducted experiments by using individual features rather than combined features.

The key function of the feature extraction step is extracting interval-based features from network traffic. We separated one-day network traffic data into many piece of time interval. Every single time interval had an index number for distinction. The number of time intervals or index numbers in one-day data depended on the interval value. For example, if we define the interval value as 1 second, the number of time intervals for one day will be 86,400, so the index numbers are from 0 to 86,399. If we change the interval value to 60 seconds, the number of time intervals for one day will be 1,440. In this case, the index numbers are from 0 to 1,439 and so forth. For our experiments, we altered the interval value between 1 and 60 seconds.

Another key point in the feature extraction step is that we plotted the data point of each time interval on each individual feature space. For example, we measured the Packet feature (f1) at the time interval holding index number 0, and then we plotted that data point on the feature space number 0. Similarly, we plotted the Packet feature (f1) data point of the time interval holding index number 1 on the feature space number 1 and so on until the last index number. Obviously, decision making relied on individual time intervals or individual feature spaces.

B. Naïve Bayes Classification

The naïve Bayes algorithm is based on Bayesian theorem [20], which can simplify in Bayes' formula as

$$P(\omega \mid x) = \frac{p(x \mid \omega)P(\omega)}{p(x)}, \quad (1)$$

so we can express Eq. (1) in plain English by saying that

$$posterior = \frac{likelihood \times prior}{evidence}. \quad (2)$$

$P(\omega|x)$ is the *posterior* probability of category ω given that feature value x has been measured; $p(x|\omega)$ is the *likelihood* function of category ω with respect to feature value x; $P(\omega)$ is the *prior* probability of category ω; and the *evidence* or $p(x)$ can be merely viewed as a scale factor that guarantees the posterior probabilities sum to one.

In our study, however, we applied the technique called one-class classification to detect anomalies, which is a bit different from original Bayes' formula. In addition, we presumed that the sum of random variables is distributed in accordance with a Gaussian or normal probability density

function, for a sufficiently large number of summing terms. Therefore, Eq. (1) could be rewritten as the univariate Gaussian defined by where the μ denotes the mean or expected value of random variable x, and the parameter σ^2 is equal to the variance of random variable x. From Eq. (3), we estimated the parameters μ and σ of each individual time interval from the training data set.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right], \quad (3)$$

To classify test data, we set the parameter value of the discriminant function between 2σ and 4σ. If a probability of feature value x or $p(x)$ computed from Eq. (3) at a particular time interval is less than the probability of the discriminant value, we classify that time interval as an anomalous interval. On the other hand, if a probability of feature value x or $p(x)$ at a particular time interval is equal to or greater than the probability of the discriminant value, we classify that time interval as a normal interval.

For example, suppose we defined the parameter value of discriminant function equal to 2σ. We measured the Packet feature (f1) to be 48 packets at a particular time interval. If the value of $p(48)$ is less than the value of $p(2\sigma)$ (both values derived from Eq. (3), we classify that time interval as an anomalous interval. However, if the value of $p(48)$ is equal to or greater than the value of $p(2\sigma)$, we classify that interval as a normal interval.

C. K-Nearest Neighbor Classification

The k-nearest neighbor algorithm is one of the fundamental and simple classification methods [21] when reliable parametric estimates of probability densities are unknown or difficult to determine. In general, this technique is based on the Euclidean distance between a test sample and specified training samples. Let x_i be a test sample with f features (x_{i1}, x_{i2},…, x_{if}); n be the total number of input samples (i = 1, 2,…, n) and f be the total number of features (j = 1, 2,…, f). The Euclidean distance between sample x_i and x_t (t = 1, 2,…, n) is defined as

$$d(x_i, x_t) = \sqrt{(x_{i1} - x_{t1})^2 + \ldots + (x_{if} - x_{tf})^2} \quad (4)$$

In our study, however, we tested each individual feature, so the Euclidean distance formula from Eq. (4) can be rewritten as

$$d(x_i, x_t) = \sqrt{(x_{i1} - x_{t1})^2} \quad (5)$$

We varied the distance values for each feature to examine the best distant value for a particular type of anomaly. The range of distance values depended on each feature, the distant value of almost all features was not over 100, and the max distance value was 24,000 for the Byte feature (f2).

To classify test data, we selected common k = 3 as the same other studies using k-nearest neighbor classification. At a particular time interval, if training samples near a test sample are not longer than the distance value, and the number of nearby training samples is less than the k value, we classify that time interval as an anomalous interval. However, if the number of nearby training samples is equal to or more than the k value, we classify that time interval as a normal interval.

For example, suppose that we set the distance value d = 20, and k = 3. We measured the Packet feature (f1) to be 48 at a particular time interval. If we find that the number of training samples, which have a feature value between 28 and 68, is less than 3, we classify that time interval as an anomalous interval. However, if we find that the number of training samples, which have a feature value between 28 and 68, is equal to or more than 3, we classify that time interval as a normal interval.

D. Support Vector Machine Classification

The support vector machines are a relatively new set of algorithms [22] that can map training data into a high-dimensional feature space. As a result, we can construct a separating hyperplane by maximizing the margin or distance from the hyperplane to the nearest training data points.

A decision function for binary support vector machine is presented by the following formula:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i . k(x, x_i) + b \right). \quad (6)$$

where 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 tools [23] with a radial basis function (RBF) kernel of the form

$$k(x, x') = \exp\left(-\gamma \left\| x - x' \right\|\right). \quad (7)$$

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 γ or the gamma value between 10^{-5} and 10^{4} to observe a change for the best detection performance.

To classify test data, we employed 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 decision formula Eq. (6). If the result at a particular time interval is negative class (-1), we classify that time interval as an anomalous interval. While, if the result is positive class (+1), we classify that time interval as a normal interval.

TABLE II: INTERVAL-BASED EVALUATION

Test Result	Actual Result	
	Anomaly	Normal
Anomaly	True Positive (TP)	False Positive (FP)
Normal	False Negative (FN)	True Negative (TN)

E. Performance Evaluation

To evaluate detection performance, we use precision (P), recall (R) [24], and F-measure value (F) [25] on a per-interval basis. All measures can be calculated on the basis of four parameters, namely the true positive (TP) rate (the number of anomalous intervals correctly detected), the false positive (FP) rate (the number of normal intervals wrongly detected as anomalous intervals), the false negative (FN) rate (the number of anomalous intervals not detected), and the true negative (TN) rate (the number of normal intervals

correctly detected). All of the parameters are defined in Table II. The precision, recall, and F-measure value are derived from these parameters by using the following Eqs. 8-10, respectively:

$$precision = \frac{TP}{TP + FP}, \quad (8)$$

$$recall = \frac{TP}{TP + FN}, \quad (9)$$

$$F\text{–}measure = 2 \times \frac{precision \times recall}{precision + recall}. \quad (10)$$

The precision or positive predictive value as Eq. (8) is the percentage of detected intervals that are actually anomalies. The recall or sensitivity value as Eq. (9) is the percentage of the actual anomalous intervals that are detected. Finally, Eq. (10) shows the F-measure value which is the harmonic mean value between precision and recall.

Theoretically, we do need methods or features for reaching a high F-measure value because the F-measure value takes both precision and recall into consideration. From a practical point of view, however, we have to consider costs of both the false positive and false negative rate. After that, we can decide whether precision or recall is the more important factor for detecting anomalies in network traffic.

F. Data Sets

The raw data were collected from an edge router of the Internet service center in Kasetsart University, Thailand. This center is for college students, educators, and researchers to obtain advantageous information for their studies from the Internet. There are about 1,300 users everyday, and the service time is between 8:30 and 24:00 on weekdays. Users cannot modify or install any software in the clients, and administrators provide appropriate software for all ordinary users. In addition, administrators regularly update the virus signatures for anti-virus software installed on all of the clients. At the end of every day, all clients automatically have the installed software and operating system returned back to the initial state. Therefore, we can guarantee that no clients contain any malicious software for attacking. We acquired attack-free network data traces from this center for 55 days in three months.

We divided the attack-free network data traces for a training data set and test data set. We selected 39-day data traces from two months as the training data set for training all of the classifiers. The other 16-day data traces are from another month, and were combined with five groups of anomalies as the test data set. The selected anomalies are from the Lincoln Laboratory at the Massachusetts Institute of Technology [26], [27]. These anomalies were collected and provided for evaluation. We selected five categories of anomalies that had the following characteristics, also as listed in Table III:

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

The 39-day training data contained only normal traffic although these machine learning algorithms can learn from both labeled and unlabeled data. On the other hand, the test data combined both normal traffic and abnormal traffic to evaluate detection performance for every feature and algorithm. According to Table III, each of the Back and IpSweep attack is 32-day test set where from the 16-day data traces were combined with two instances. Moreover, each Neptune, PortSweep, and Smurf attack is 48-day test set where from the 16-day data traces were combined with three instances.

TABLE III: 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

III. Results

To ascertain the best F-measure value of each feature for a different type of anomaly, we varied the time interval value and determined parameters for each algorithm. For all three algorithms, we commonly varied time interval value with 1, 10, 20, 30, 40, 50, and 60 seconds. Moreover, we varied the value of significant parameter for each algorithm: the discriminant value for naïve Bayes, the distance value for *k*-nearest neighbor, and the gamma value for support vector machine.

In our experiment, the classification is interval basis, so each training or test sample represents a single time interval in training or test data for one-day traffic. Along one-day test data, we computed the precision, recall, and F-measure value and then calculated the average of the three values for all test data. After that, we discovered the highest F-measure value of each feature for a different type of anomaly.

A. Experiment 1: Naïve Bayes

In the first experiment, the discriminant value affects the F-measure values on different time interval values. To examine which discriminant value gains the best F-measure value, we varied the discriminant value from 2.0 to 4.0, increasing by 0.1 each time.

The following describes a process to discover the best F-measure value of each feature for a different type of anomaly by using the naïve Bayes algorithm. First, we selected test data containing the Back attacks and tested detection performance on the Packet feature (f1). Then we computed the average value of precision, recall, and F-measure. Second, we varied time interval values and discriminant values, so we can plot average F-measure values on a graph as shown in Fig. 2. The x-axis represents time interval values, the y-axis illustrates discriminant values, and the z-axis shows F-measure.

From Fig. 2, we spotted the best F-measure value for Back attacks by using the Packet feature (f1). Next, we switched from the Packet feature (f1) to the Byte feature (f2) and performed it again from the first step until all nine features had been used. After testing all nine features, we changed the type of anomalies from Back attacks to the IpSweep attacks, and performed the experiment again from the first step. We carried out all tasks like this for all five types of anomalies by using all nine features. Finally, we compared the maximum value of precision, recall, and F-measure value for each anomaly sorted by features (f1-f9) as shown in Fig. 3. The x-axis indicates all nine features and y-axis shows the value of precision, recall, and F-measure.

B. Experiment 2: K-Nearest Neighbor

In the second experiment, the distance value affects the F-measure values on different time interval values. To determine which distance value earns the highest F-measure value, we commonly varied the distance value from 1 up to 100, increasing by 1 each time, except the Byte feature (f2), which we varied the distance value up to 24,000. We constantly specified the number of *k* is equal to 3 for all experiments using *k*-nearest neighbor.

We conducted the experiment like a naïve Bayes experiment to define the highest F-measure value of each feature for different types of anomalies. After varying time interval values and distance values on the Back attacks by using the Packet feature (f1), we plotted F-measure value on a graph as shown in Fig. 4. The x-axis represents time interval values, y-axis illustrates distance values, and z-axis shows F-measure values. We can notice the different shapes of graphs between using naïve Bayes and *k*-nearest neighbor.

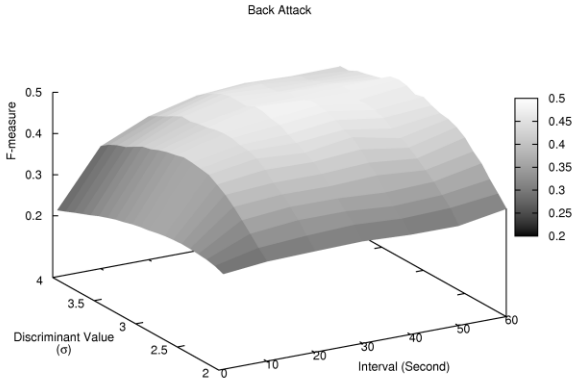

Fig. 2. F-measure of Packet (f1) using naïve Bayes.

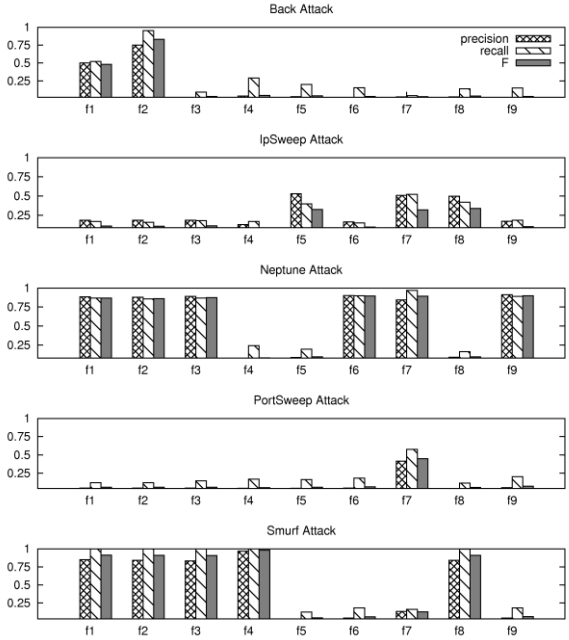

Fig. 3. Performance comparison using naïve Bayes.

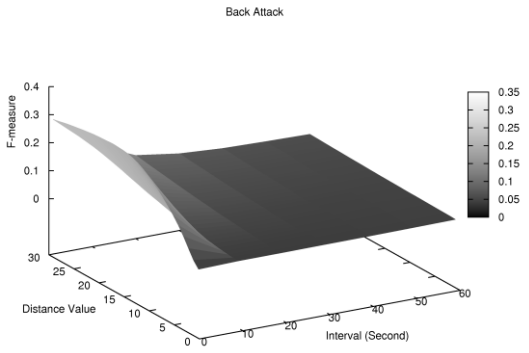

Fig. 4. F-measure of Packet (f1) using *k*-nearest neighbor.

We spotted the highest F-measure value of the Back attacks by using the Packet feature (f1) from Fig. 4. Next, we went through a process the same as that in the naïve

Bayes experiment for all five types of anomalies and all nine features. Finally, we compared the maximum value of precision, recall, and F-measure for individual anomalies sorted by features (f1-f9) as shown in Fig. 5.

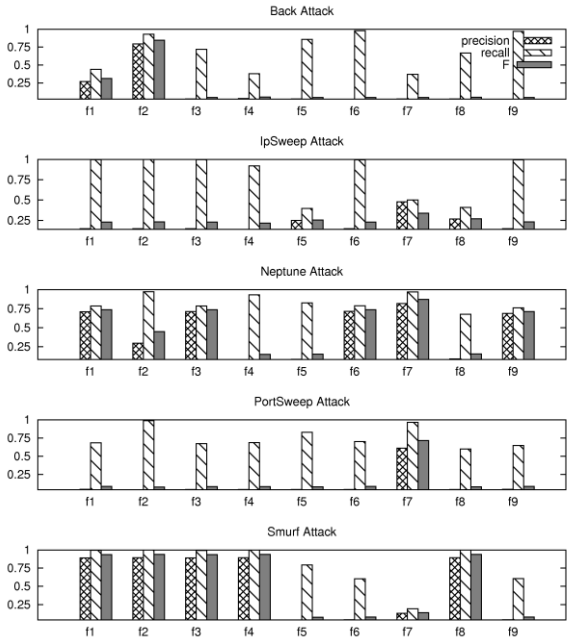

Fig. 5. Performance comparison using *k*-nearest neighbor.

C. Experiment 3: Support Vector Machine

The final experiment was conducted on the assumption that the gamma affects the F-measure values on different time interval values. Therefore, we varied the gamma value between 10^{-5} and 10^{4} for different time interval values to ascertain the best gamma value.

We performed the experiment in the same way as for both previous algorithms to discover the finest F-measure value of each feature for a different type of anomaly. First, we varied time interval values and gamma values on the Back attacks by using the Packet feature (f1), and then we plotted F-measure values on a graph as shown in Fig. 6. The x-axis represents time interval values, y-axis illustrates gamma values, and z-axis shows F-measure values.

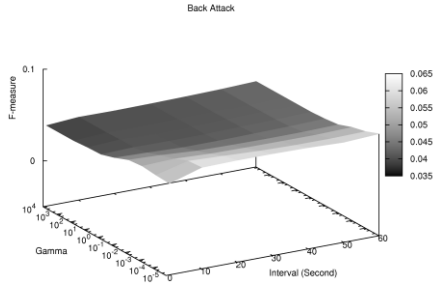

Fig. 6. F-measure of Packet (f1) using support vector machine.

As shown in Fig. 6, we discovered the finest F-measure value of Back attacks by using the Packet feature (f1). Next, we went through the same process as in the two previous experiments for all five types of anomalies by using all nine features. Eventually, we compared the maximum value of precision, recall, and F-measure value for each anomaly sorted by features (f1-f9) as shown in Fig. 7. The x-axis shows all nine features and y-axis indicates the value of precision, recall, and F-measure.

For all three experiments, we list the top three features of highest prediction performance in Table V, which is primarily ordered by the naïve Bayes algorithm. This table indicates the interval values (Intvl.) of the best detection performance where the unit of this column is second. Moreover, the table separately presents precise values of precision (P), recall (R), and F-measure (F) for each learning algorithm.

IV. Discussion

Our results suggest that selecting the proper interval-based network traffic feature for particular types of anomalies is the key to a great success in anomaly detection by using machine learning approach. By using naïve Bayes and *k*-nearest neighbor algorithms, we can use the Packet (f1) or Byte feature (f2) to detect anomalies contained in a large number of packets, such as the Back, Neptune, and Smurf attack. In addition, the Flow feature (f3) can detect only a huge number of connections or flows, such as the Neptune and Smurf attack. The SrcAddr (f4), DstAddr (f5) and ΔAddr features (f8) efficiently detect anomalies in which source addresses and destination addresses been varied, such as the IpSweep and Smurf attack. The DstPort feature (f7) effectively detects anomalies that have changed the destination ports, especially in PortSweep attack. However, it is different from the SrcPort (f6) and ΔPort features (f9), both of these features can detect only substantial varying in source ports like a Neptune attack, because source port numbers are always changed by source computers.

In contrast to both the naïve Bayes and *k*-nearest neighbor algorithms, we found that the performance of support vector machine contains a high false positive rate. Although recall values of all features were high when the support vector machine was used, the high false positive rate caused the low values of F-measure. The main reason for the high false positive values is that the support vector machine is sensitive to noisy data, of which the network traffic contains many. Moreover, we discovered that the gamma value is not strong enough to affect the F-measure values on different time interval values.

Consideration in time complexity is a secondary issue to how effective each algorithm will be. Table IV shows the time complexity of training phase and test phase for each algorithm. In this table, m is the number of training days, n is the number of time intervals for one-day long, and f is the number of features, we performed our experiment by using individual features so $f = 1$, and s is the number of support vectors generated by the Svm-Train function from the Libsvm, which depend on patterns of training data. If the training data closely gather together, the number of support vectors will be low; however, if the training data disperse, the number of support vectors will reach a higher value.

Although we can apply more than one feature to machine learning algorithms, the number of features affects the processing time of both training and test phases. For the

naïve Bayes, the time complexity of the training phase relies on *m*, *n* and *f*, while the test phase relies on *n* and *f*. These indicate when we combine more than one feature, the processing time for training and test phases linearly increases. For *k*-nearest neighbor, although the time complexity of the training phase is a constant, the test phase depends on *m*, *n*, and *f*. This indicates that only the processing time of the test phase is linearly grown when we increase the number of features. For the support vector machine, the time complexity of the training phase relies on m^2, *n* and f^2 while the test phase relies on *n*, *f*, and *s*. That means when we add a number of features, the processing time of the training phase increases exponentially, while the processing time of the test phase increases linearly.

TABLE IV: TIME COMPLEXITY

Classifier	Training Phase	Test Phase
Naïve Bayes	*O(mnf)*	*O(nf)*
k-Nearest Neighbor	*O(1)*	*O(mnf)*
Support Vector Machine	$O(m^2nf^2)$	*O(nfs)*

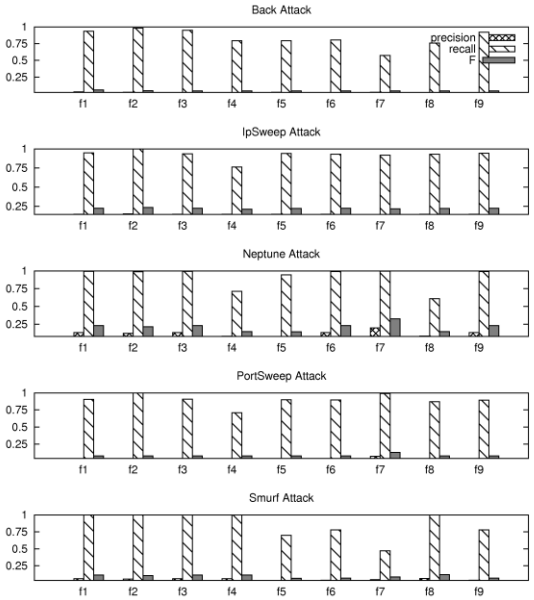

Fig. 7. Performance comparison using support vector machine.

TABLE V: TOP 3 FEATURES OF DETECTION PERFORMANCE

Anomaly Type	f#	Naïve Bayes				*k*-Nearest Neighbor				Support Vector Machine			
		Intvl.	P	R	F	Intvl.	P	R	F	Intvl.	P	R	F
Back	f2	10	0.75	0.95	0.83	1	0.79	0.93	0.84	1	0.03	0.98	0.05
	f1	40	0.50	0.52	0.48	1	0.27	0.44	0.31	20	0.03	0.93	0.06
	f4	40	0.04	0.29	0.05	30	0.03	0.38	0.05	60	0.02	0.79	0.04
IpSweep	f8	60	0.50	0.42	0.34	1	0.27	0.41	0.27	10	0.14	0.93	0.22
	f5	60	0.53	0.40	0.33	1	0.25	0.40	0.25	10	0.14	0.94	0.22
	f7	10	0.51	0.52	0.32	10	0.48	0.50	0.34	10	0.14	0.92	0.22
Neptune	f9	30	0.91	0.89	0.90	1	0.69	0.76	0.71	1	0.13	0.99	0.23
	f6	30	0.90	0.90	0.90	1	0.71	0.79	0.74	1	0.13	0.99	0.23
	f7	10	0.84	0.97	0.89	10	0.82	0.97	0.87	1	0.20	1.00	0.33
PortSweep	f7	50	0.41	0.58	0.45	60	0.61	0.97	0.71	60	0.07	0.99	0.13
	f9	60	0.05	0.20	0.07	10	0.04	0.65	0.08	10	0.04	0.90	0.08
	f6	60	0.05	0.18	0.06	10	0.04	0.70	0.08	10	0.04	0.90	0.07
Smurf	f4	1	0.97	1.00	0.98	1	0.89	1.00	0.94	1	0.06	1.00	0.11
	f1	10	0.85	1.00	0.91	1	0.89	0.99	0.94	1	0.06	1.00	0.11
	f8	1	0.84	0.99	0.91	1	0.89	1.00	0.94	1	0.06	1.00	0.12

In summary, our work revealed the effective interval-based network traffic features for five major types of network traffic anomalies. In many cases, the results from the naïve Bayes and *k*-nearest neighbor algorithm were virtually the same. Nearly all F-measure values of naïve Bayes were higher than those of the *k*-nearest neighbor algorithm. However, almost all recall values of *k*-nearest neighbor were better than those of naïve Bayes. For the support vector machine, F-measure values for all features were the lowest of the three algorithms, but recall values were high for almost all features. Our experimental results not only reveale interval-based features which enable administrators to detect current anomalies more accurately and quickly but these can be used for detecting a novel anomaly in real time as well.

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] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1-15:58, July 2009.
[2] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448-3470, 2007.
[3] J. McHugh, "Intrusion and intrusion detection," *International Journal of Information Security*, vol. 1, pp. 14-35, 2001.
[4] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proc. the 13th USENIX conference on System administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229-238.
[5] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, pp. 2435-2463, December 1999.
[6] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Data Min. Knowl. Discov.*, vol. 8, pp. 275-300, May 2004.
[7] R. Gwadera, M. J. Atallah, and W. Szpankowski, "Reliable detection of episodes in event sequences," *Knowl. Inf. Syst.*, vol. 7, pp. 415-437, May 2005.
[8] C. Chow, "Parzen-window network intrusion detectors," in *Proc. the 16th International Conference on Pattern Recognition (ICPR'02)*, vol. 4, ser. ICPR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 385-388.
[9] D. Barbará, J. Couto, S. Jajodia, and N. Wu, "Adam: a testbed for exploring the use of data mining in intrusion detection," *SIGMOD Rec.*, vol. 30, pp. 15-24, December 2001.

[10] K. Sequeira and M. Zaki, "Admit: anomaly-based data mining for intrusions," in *Proc. the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 386-395.
[11] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191-2204, Aug. 2003.
[12] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proc. the 15th Annual Computer Security Applications Conference*, ser. ACSAC '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 371-377.
[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, Oct. 2007.
[14] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proc. the Twenty-Eighth Australasian Conference on Computer Science - Volume 38*, ser. ACSC '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 333-342.
[15] P. Laskov, P. Düssel, C. Schäfer, 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. Springer Berlin / Heidelberg, 2005, vol. 3617, pp. 50-57.
[16] D. Barbará, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators," in *Proc. the First SIAM Conference on Data Mining*, Apr. 2001.
[17] L. Kuang, "Dnids: A dependable network intrusion detection system using the csi-knn algorithm," 2007.
[18] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, pp. 507-521, October 2007.
[19] K. Limthong, P. Watanapongse, and F. Kensuke, "A wavelet-based anomaly detection for outbound network traffic," in *Proc. 8th Asia-Pacific Symposium on Information and Telecommunication Technologies*, June 2010.
[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*, 4th ed. Academic Press, 2008.
[21] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.
[22] 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.
[23] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1-27:27, May 2011.
[24] 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.
[25] C. J. V. Rijsbergen, *Information Retrieval*. Newton, MA, USA: Butterworth-Heinemann, 1979.
[26] R. Lippmann, D. Fried, I. Graf *et al.*, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," vol. 2, pp. 12-26, 2000.
[27] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579-595, 2000.

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.